

LNBI 5541

Serafim Batzoglou (Ed.)

# Research in Computational Molecular Biology

13th Annual International Conference, RECOMB 2009  
Tucson, AZ, USA, May 2009  
Proceedings

 Springer

Lecture Notes in Bioinformatics

5541

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

Editorial Board: A. Apostolico S. Brunak M. Gelfand  
T. Lengauer S. Miyano G. Myers M.-F. Sagot D. Sankoff  
R. Shamir T. Speed M. Vingron W. Wong

Subseries of Lecture Notes in Computer Science

Serafim Batzoglou (Ed.)

# Research in Computational Molecular Biology

13th Annual International Conference, RECOMB 2009  
Tucson, AZ, USA, May 18-21, 2009  
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 Editor**

Serafim Batzoglou

Computer Science Department

James H. Clark Center, 318 Campus Drive, RM S266

Stanford, CA 94305-5428, USA

E-mail: [serafim@cs.stanford.edu](mailto:serafim@cs.stanford.edu)

Library of Congress Control Number: Applied for

CR Subject Classification (1998): J.3, I.3.5, F.2.2, F.2, G.2.1

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743

ISBN-10 3-642-02007-0 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-02007-0 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

[springer.com](http://springer.com)

© Springer-Verlag Berlin Heidelberg 2009

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12677146 06/3180 5 4 3 2 1 0

# Preface

This volume contains the papers presented at RECOMB 2009: the 13th Annual International Conference on Research in Computational Molecular Biology held in Tucson, Arizona, USA, during May 18-21, 2009. The RECOMB conference series was started in 1997 by Sorin Istrail, Pavel Pevzner, and Michael Waterman. RECOMB 2009 was hosted by the University of Arizona, organized by a committee chaired by John Kececioglu, and took place at The Westin La Paloma Resort and Spa in Tucson, Arizona.

This year, 37 papers were accepted for presentation out of 166 submissions. The papers presented were selected by the Program Committee (PC) assisted by a number of external reviewers. Each paper was reviewed by three members of the PC, or by external reviewers acting as sub-reviewers to members of the PC. Following the initial reviews, there was an extensive Web-based discussion over a period of two weeks, leading to the final decisions. The RECOMB conference series is closely associated with the *Journal of Computational Biology*, which traditionally publishes special issues devoted to presenting full versions of selected conference papers.

RECOMB 2009 invited several distinguished speakers as keynotes and for a special session on “Personalized Genomics”. Invited speakers included Carlos D. Bustamante (Cornell University), Rade Drmanac (Complete Genomics), Mark Gerstein (Yale University), Eran Halperin (Navigenics), Michael Hammer (University of Arizona), Joanna Mountain (23andMe), Stephen Quake (Stanford University), Mostafa Ronaghi (Illumina), Pardis Sabeti (Harvard University), and Michael Snyder (Yale University).

RECOMB 2009 was only possible through the dedication and hard work of many individuals and organizations. Special thanks go to the PC and external reviewers for helping to form the conference program, and the organizers, chaired by John Kececioglu, for hosting the conference and providing the administrative, logistic, and financial support. Special thanks also go to our sponsors. The conference was overseen by the RECOMB Steering Committee. We thank Marina Sirota for help with editing the proceedings volume. Finally, we thank all the authors who contributed papers and posters, as well as the attendees of the conference for their enthusiastic participation.

March 2009

Serafim Batzoglou

# Conference Organization

## Program Committee

Tatsuya Akutsu	Kyoto University, Japan
Serafim Batzoglou (Program Chair)	Stanford University, USA
Gill Bejerano	Stanford University, USA
Bonnie Berger	Massachusetts Institute of Technology, USA
Michael Brent	Washington University, USA
Mike Brudno	University of Toronto, Canada
Jeremy Buhler	Washington University, USA
Atul Butte	Stanford University, USA
Rhiju Das	Stanford University, USA
Colin Dewey	University of Wisconsin, USA
Eleazar Eskin	University of California Los Angeles, USA
Nir Friedman	Hebrew University, Israel
James Galagan	Broad Institute, USA
Eran Halperin	UC Berkeley, USA
Alexander Hartemink	Duke University, USA
Des Higgins	University College Dublin, Ireland
Trey Ideker	University of California San Diego, USA
Sorin Istrail	Brown University, USA
Tao Jiang	University of California Riverside, USA
Simon Kasif	Boston University, USA
John Kececioglu (Conference Chair)	University of Arizona, USA
Manolis Kellis	Massachusetts Institute of Technology, USA
Jens Lagergren	Stockholm University, Sweden
Thomas Lengauer	Max Planck Institut-Informatik-für, Germany
Satoru Miyano	Tokyo University, Japan
William Noble	University of Washington, USA
Pavel Pevzner	University of California San Diego, USA
Ron Pinter	Technion, Israel
Aviv Regev	Massachusetts Institute of Technology, USA
Knut Reinert	Freie Universität Berlin, Germany
David Sankoff	University of Ottawa, Canada
Russell Schwartz	Carnegie Mellon University, USA
Eran Segal	Weizmann Institute, Israel
Roded Sharan	Tel Aviv University, Israel

## VIII Organization

Adam Siepel	Cornell University, USA
Mona Singh	Princeton University, USA
Peter Stadler	Universität Leipzig, Germany
Jens Stoye	Universität Bielefeld, Germany
Josh Stuart	University of California Santa Cruz, USA
Fengzhu Sun	University of Southern California, USA
Olga Troyanskaya	Princeton University, USA
Martin Vingron	Max Planck Institute, Germany
Tandy Warnow	University of Texas Austin, USA
Eric Xing	Carnegie Mellon University, USA
Zohar Yakhini	Agilent Laboratories

## Steering Committee

Serafim Batzoglou	Stanford University, USA
Sorin Istrail	RECOMB Vice-Chair, Brown University, USA
Thomas Lengauer	Max Planck Institute, Germany
Michal Linial	Hebrew University, Israel
Pavel A. Pevzner	RECOMB General Chair, University of California San Diego, USA
Terence P. Speed	University of California Berkeley, USA

## Local Organization at the University of Arizona

John Kececioglu (Conference Chair)	University of Arizona
Maura Grohan	BIO5 Institute and University of Arizona
Daphne Gillman	BIO5 Institute and University of Arizona
Deborah Daun	BIO5 Institute and University of Arizona
Steve Dix	BIO5 Institute and University of Arizona

## Previous RECOMB Meetings

Dates	Hosting Institution	Program Chair	Conference Chair
January 20-23, 1997 Santa Fe, NM, USA	Sandia National Lab	Michael Waterman	Sorin Istrail
March 22-25, 1998 New York, NY, USA	Mt. Sinai School of Medicine	Pavel Pevzner	Gary Benson
April 22-25, 1999 Lyon, France	INRIA	Sorin Istrail	Mireille Regnier
April 8-11, 2000 Tokyo, Japan	University of Tokyo	Ron Shamir	Satoru Miyano
April 22-25, 2001 Montreal, Canada	Universite de Montreal	Thomas Lengauer	David Sankoff
April 18-21, 2002 Washington, DC, USA	Celera	Gene Myers	Sridhar Hannenhalli
April 10-13, 2003 Berlin, Germany	German Federal Ministry for Education and Research	Webb Miller	Martin Vingron
March 27-31, 2004 San Diego, USA	UC San Diego	Dan Gusfield	Philip E. Bourne
May 14-18, 2005 Boston, MA, USA	Broad Institute of MIT and Harvard	Satoru Miyano	Jill P. Mesirov and Simon Kasif
April 2-5, 2006 Venice, Italy	University of Padova	Alberto Apostolico	Concettina Guerra
April 21-25, 2007 San Francisco, CA	QB3	Terry Speed	Sandrine Dudoit
March 30 - April 2 2008 Singapore, Singapore	National University of Singapore	Martin Vingron	Limsoon Wong

## External Reviewers

Alber, Frank	Candeias, Rogerio
Alekseyev, Max	Chen, Rong
Andreotti, Sandro	Chen, Xiaoyu
Anton, Brian	Chen, Yang-ho
Arvestad, Lars	Cho, Sungje
Aydin, Zafer	Chor, Benny
Berry, Vincent	Chuang, Han-Yu
Bandyopadhyay, Sourav	Chung, Ho-Ryun
Bar-Joseph, Ziv	Cohen-Gihon, Inbar
Bauer, Markus	Cowen, Lenore
Baumbach, Jan	Csuros, Miklos
Bercovici, Sivan	Dalca, Adrian
Bernhart, Stephan	Diehl, Adam
Bielow, Chris	Do, Chuong
Blom, Jochen	Dudley, Joel
Bordewich, Magnus	Dunbrack, Roland
Brejova, Bronislava	Elidan, Gal
Brown, Randall	El-Hay, Tal
Bruckner, Sharon	Eran, Ally

Ernst, Jason	Lee, Ki Young
Fernndez-Baca, David	Lee, KiYoung
Fujita, Andre	Li, Yong
Gat-Viks, Irit	Lilien, Ryan
Gerlach, Wolfgang	Lin, Michael
Goff, Loyal	Lonardi, Stefano
GrApl, Clemens	Ma, Xiaotu
Gusfield, Dan	Martins, Andre
Haas, Stefan	McIlwain, Sean
Habib, Naomi	Medvedev, Paul
Haim, Wolfson, Efrat Mashiach and	Meshi, Ofer
Han, Buhm	MilaniÄ, Martin
Hannum, Gregory	Misra, Navodit
Hayashida, Morihiro	Mitrofanova, Antonina
Haynes, Brian	Morgan, Alex
Heinig, Matthias	Morozov, Alexandre
Hildebrandt, Andreas	Moses, Alan
Hoffman, Michael	Mosig, Axel
Hudek, Alexander	Mil, Mathias
Hue, Martial	Nakhleh, Luay
Husemann, Peter	Navon, Roy
Huson, Daniel	Ng, Julio
Imoto, Seiya	Novershtern, Noa
Inbar, Yuval	Obozinski, Guillaume
Jaimovich, Ariel	Paik, David
Jeong, Euna	Pardo, Matteo
Kaell, Lukas	Pasaniuc, Bogdan
Kamisetty, Hetunandan	Perrier, Eric
Kamisetty, Hndetunandan	Phillips, Michael
Kang, Hyun Min	Pincus, Zach
Kato, Yuki	Pop, Mihai
Kelley, Ryan	Qi, Yanjun
Kheradpour, Pouya	Raphael, Ben
Kim, Seyoung	Rasmussen, Matthew
Kimmel, Gad	Ray, Pradipta
Kirkpatrick, Bonnie	Reynolds, Sheila
Klau, Gunnar	RingnÄr, Markus
Kojima, Kaname	Salari, Rahele
Kolaczyk, Eric	Sankararaman, Sriram
Kosiol, Carolin	Satulovsky, Javier
Kriete, Andres	Schelhorn, Sven-Eric
Kuang, Rui	Schuster, Meromit
Kuttykrishnan, Sooraj	Sealfon, Rachel
Lasserre, Julia	Segal, Eran
Lee, Byoungkoo	Sennblad, Bengt

- Shen-Orr, Shai  
Shimamura, Teppei  
Shlomi, Tomer  
Shmoish, Michael  
Shringarpure, Suyash  
Sievers, Fabian  
Sjöstrand, Joel  
Sohn, Kyung-Ah  
Song, Le  
Sridhar, Srinath  
Srivas, Rohith  
Steel, Mike  
Steffen, Martin  
Steinfeld, Israel  
Steinhoff, Christine  
Subramanian, Ayshwarya  
Suthram, Silpa  
Taliwal, Vikas  
Tamada, Yoshinori  
Tamura, Takeyuki  
Wittler, Roland  
Wu, Xuebing  
Xia, Charles Li  
Xu, Wei  
Yamaguchi, Rui  
Yanovsky, Vladimir  
Ye, Chun  
Yosef, Nir  
Yuan, Yuan  
Zaitlen, Noah  
Zerck, Alexandra  
Zhang, Kui  
Zheng, Chunfang

## Table of Contents

Searching Protein 3-D Structures in Linear Time .....	1
<i>Tetsuo Shibuya</i>	
Optimization-Based Peptide Mass Fingerprinting for Protein Mixture Identification .....	16
<i>Zengyou He, Chao Yang, Can Yang, Robert Z. Qi,     Jason Po-Ming Tam, and Weichuan Yu</i>	
Boosting Protein Threading Accuracy .....	31
<i>Jian Peng and Jinbo Xu</i>	
New Perspectives on Gene Family Evolution: Losses in Reconciliation and a Link with Supertrees .....	46
<i>Cedric Chauve and Nadia El-Mabrouk</i>	
A Probabilistic Graphical Model for Ab Initio Folding .....	59
<i>Feng Zhao, Jian Peng, Joe DeBartolo, Karl F. Freed,     Tobin R. Sosnick, and Jinbo Xu</i>	
Topology-Free Querying of Protein Interaction Networks .....	74
<i>Sharon Bruckner, Falk Hüffner, Richard M. Karp, Ron Shamir, and     Roded Sharan</i>	
Cross Species Expression Analysis of Innate Immune Response .....	90
<i>Yong Lu, Roni Rosenfeld, Gerard J. Nau, and Ziv Bar-Joseph</i>	
Haplotype Inference in Complex Pedigrees .....	108
<i>Bonnie Kirkpatrick, Javier Rosa, Eran Halperin, and     Richard M. Karp</i>	
Storage and Retrieval of Individual Genomes .....	121
<i>Veli Mäkinen, Gonzalo Navarro, Jouni Sirén, and Niko Välimäki</i>	
An Online Approach for Mining Collective Behaviors from Molecular Dynamics Simulations .....	138
<i>Arvind Ramanathan, Pratul K. Agarwal, Maria Kurnikova, and     Christopher J. Langmead</i>	
Parameter Synthesis in Nonlinear Dynamical Systems: Application to Systems Biology .....	155
<i>Alexandre Donzé, Gilles Clermont, Axel Legay, and     Christopher J. Langmead</i>	

Spatial Clustering of Multivariate Genomic and Epigenomic Information . . . . .	170
<i>Rami Jaschek and Amos Tanay</i>	
How Many Bootstrap Replicates Are Necessary? . . . . .	184
<i>Nicholas D. Pattengale, Masoud Alipour, Olaf R.P. Bininda-Emonds, Bernard M.E. Moret, and Alexandros Stamatakis</i>	
A Robust Bayesian Two-Sample Test for Detecting Intervals of Differential Gene Expression in Microarray Time Series . . . . .	201
<i>Oliver Stegle, Katherine Denby, David L. Wild, Zoubin Ghahramani, and Karsten M. Borgwardt</i>	
Incorporating Nucleosomes into Thermodynamic Models of Transcription Regulation . . . . .	217
<i>Tali Raveh-Sadka, Michal Levo, and Eran Segal</i>	
Combinatorial Algorithms for Structural Variation Detection in High Throughput Sequenced Genomes . . . . .	218
<i>Fereydoun Hormozdiari, Can Alkan, Evan E. Eichler, and S. Cenk Sahinalp</i>	
Optimizing PCR Assays for DNA Based Cancer Diagnostics . . . . .	220
<i>Ali Bashir, Qing Lu, Dennis Carson, Benjamin Raphael, Yu-Tsuong Liu, and Vineet Bafna</i>	
The Multi-State Perfect Phylogeny Problem with Missing and Removable Data: Solutions via Integer-Programming and Chordal Graph Theory . . . . .	236
<i>Dan Gusfield</i>	
COE: A General Approach for Efficient Genome-Wide Two-Locus Epistasis Test in Disease Association Study . . . . .	253
<i>Xiang Zhang, Feng Pan, Yuying Xie, Fei Zou, and Wei Wang</i>	
Overlapping Pools for High Throughput Targeted Resequencing . . . . .	270
<i>Snehit Prabhu and Itsik Pe'er</i>	
Deep Sequencing of a Genetically Heterogeneous Sample: Local Haplotype Reconstruction and Read Error Correction . . . . .	271
<i>Osvaldo Zagordi, Lukas Geyrhofer, Volker Roth, and Niko Beerenwinkel</i>	
Lifting Prediction to Alignment of RNA Pseudoknots . . . . .	285
<i>Mathias Möhl, Sebastian Will, and Rolf Backofen</i>	
Detection of Locally Over-Represented GO Terms in Protein-Protein Interaction Networks . . . . .	302
<i>Mathieu Lavallée-Adam, Benoit Coulombe, and Mathieu Blanchette</i>	

Protein Fragment Swapping: A Method for Asymmetric, Selective Site-Directed Recombination .....	321
<i>Wei Zheng, Karl E. Griswold, and Chris Bailey-Kellogg</i>	
Simultaneous Alignment and Folding of Protein Sequences .....	339
<i>Jérôme Waldspühl, Charles W. O'Donnell, Sebastian Will, Srinivas Devadas, Rolf Backofen, and Bonnie Berger</i>	
Shared Peptides in Mass Spectrometry Based Protein Quantification ...	356
<i>Banu Dost, Nuno Bandeira, Xiangqian Li, Zhouxin Shen, Steve Briggs, and Vineet Bafna</i>	
Evaluating Between-Pathway Models with Expression Data .....	372
<i>Benjamin J. Hescott, Mack D.M. Leiserson, Lenore J. Cowen, and Donna K. Slonim</i>	
Sorting Signed Permutations by Inversions in $O(n \log n)$ Time .....	386
<i>Krister M. Swenson, Vaibhav Rajan, Yu Lin, and Bernard M.E. Moret</i>	
Finding Biologically Accurate Clusterings in Hierarchical Tree Decompositions Using the Variation of Information .....	400
<i>Saket Navlakha, James White, Niranjan Nagarajan, Mihai Pop, and Carl Kingsford</i>	
Identification and Frequency Estimation of Inversion Polymorphisms from Haplotype Data .....	418
<i>Suzanne S. Sindi and Benjamin J. Raphael</i>	
On the Relationship between DNA Periodicity and Local Chromatin Structure .....	434
<i>Sheila M. Reynolds, Jeff A. Bilmes, and William Stafford Noble</i>	
Phylogenies without Branch Bounds: Contracting the Short, Pruning the Deep (Extended Abstract) .....	451
<i>Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch</i>	
Detecting the Presence and Absence of Causal Relationships between Expression of Yeast Genes with Very Few Samples .....	466
<i>Eun Yong Kang, Ilya Shpitser, Chun Ye, and Eleazar Eskin</i>	
An Adaptive and Memory Efficient Algorithm for Genotype Imputation .....	482
<i>Hyun Min Kang, Noah A. Zaitlen, Buhm Han, and Eleazar Eskin</i>	
A Statistical Framework for the Functional Analysis of Metagenomes ...	496
<i>Itai Sharon, Amrita Pati, Victor M. Markowitz, and Ron Y. Pinter</i>	

XVI      Table of Contents

Learning Models for Aligning Protein Sequences with Predicted Secondary Structure.....	512
<i>Eagu Kim, Travis Wheeler, and John Kececioglu</i>	
<b>Author Index .....</b>	<b>533</b>

# Searching Protein 3-D Structures in Linear Time

Tetsuo Shibuya

Human Genome Center, Institute of Medical Science, University of Tokyo  
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan  
[tshibuya@hgc.jp](mailto:tshibuya@hgc.jp)

**Abstract.** Finding similar structures from 3-D structure databases of proteins is becoming more and more important issue in the post-genomic molecular biology. To compare 3-D structures of two molecules, biologists mostly use the RMSD (root mean square deviation) as the similarity measure. We propose new theoretically and practically fast algorithms for the fundamental problem of finding all the substructures of structures in a structure database of chain molecules (such as proteins), whose RMSDs to the query are within a given constant threshold. We first propose a breakthrough linear-expected-time algorithm for the problem, while the previous best-known time complexity was  $O(N \log m)$ , where  $N$  is the database size and  $m$  is the query size. For the expected time analysis, we propose to use the random-walk model (or the ideal chain model) as the model of average protein structures. We furthermore propose a series of preprocessing algorithms that enable faster queries. We checked the performance of our linear-expected-time algorithm through computational experiments over the whole PDB database. According to the experiments, our algorithm is 3.6 to 28 times faster than previously known algorithms for ordinary queries. Moreover, the experimental results support the validity of our theoretical analyses.

## 1 Introduction

3-D structure database searching of molecules, especially proteins, plays a very important role in molecular biology [28][10]. For example, if we have proteins whose structures are known but their functions are unknown, we may be able to predict their functions by searching for similar structures whose functions are known, as structurally similar proteins tend to have similar functions. Moreover, more and more protein structures are solved today with the aid of state-of-the-art technologies such as nuclear magnetic resonance (NMR) techniques, as seen in the rapid growth of the PDB database [3]. Thus, faster searching techniques are seriously needed for the molecular structure databases.

A protein is a chain of amino acids. Thus, its structure can be represented by a sequence of 3-D coordinates, each of which corresponds to the position of a specified atom (the  $C_\alpha$  atom is usually used) of each amino acid. Such molecules are called chain molecules. There are also many other important chain molecules in living cells, such as DNAs, RNAs and glycans. The RMSD (root mean square

deviation) [1, 7, 12, 13, 16, 19] is the fundamental measure to determine the geometric similarity between two same-length sequences of 3-D coordinates. It has been widely used not only for molecular structure comparison, but also for various problems in various fields, such as computer vision and robotics. It is defined as the square root of the minimum value of the average squared distance between each pair of corresponding atoms, over all the possible rotations and translations (see section 2.2). In this paper, we consider one of the most fundamental RMSD-related problems as follows.

**Problem.** Given a structure database  $\mathcal{D}$  of chain molecules and a query structure  $\mathbf{Q}$ , find all the substructures of the structures in  $\mathcal{D}$  whose RMSDs to  $\mathbf{Q}$  are at most a given fixed threshold  $c$ , without considering any insertions or deletions.

In general,  $c$  should be set to a fixed constant proportional to the distance between two adjacent atoms of the chain molecules. In the case of proteins, the distance between two adjacent  $C_\alpha$  atoms is around 3.8 Å, while two protein structures are said to be similar to each other if their RMSD is smaller than 1 or 2 Å.

**Our results.** The best-known worst-case/expected time complexity of the problem was  $O(N \log m)$  [16, 19], where  $N$  is the database size (*i.e.*, the sum of the lengths of all the structures in the database) and  $m$  is the query size. We propose the first linear-expected-time (*i.e.*,  $O(N)$ ) algorithm. To analyze the expected time of the algorithm, we give an assumption that the structures in the database follow a model called the random-walk model (see section 2.4). We also propose several preprocessing algorithms that enable faster queries. We first propose an  $O(N \log N)$ -time and  $O(N)$ -space preprocessing algorithm that enables  $O(m + N/\sqrt{m})$ -expected-time query, for queries of a fixed length. We next extend it to an  $O(N \log^2 N)$ -time and  $O(N \log N)$ -space preprocessing algorithm that enables the same  $O(m + N/\sqrt{m})$ -expected-time query, for queries of arbitrary lengths. We also propose an  $O(N \log N)$ -time and  $O(N)$ -space preprocessing algorithm that enables  $O(\frac{N}{\sqrt{m}} + m \log(N/m))$ -expected-time query, for queries of arbitrary lengths.

We also examine the performance of our algorithms by computational experiments on the whole PDB database. Our linear-time algorithm is much faster than any of previous algorithms, *i.e.*, 3.6 to 28 times faster to search for substructures whose RMSDs are at most 1 Å. Moreover, no inconsistency is observed between the theoretical results and the experimental results, which means our random-walk assumption is very reasonable for analyses of algorithms for protein structure database search.

**The organization of this paper.** In section 2, we describe the basic definitions and related previous work as preliminaries. In section 3, we propose an  $O(N\sqrt{m})$  algorithm for the problem above, where  $N$  is the database size and  $m$  is the length of the query. We next improve it to obtain the linear-time algorithm in section 4. In section 5, we further extend our algorithms for faster queries after preprocessing. In section 6, we examine the performance of our algorithms against the PDB database. In section 7, we conclude our results.

## 2 Preliminaries

### 2.1 Notations and Definitions

A chain molecule is represented like  $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$ , where  $\mathbf{s}_i$  denotes the 3-D coordinates of the  $i$ -th atom. The length  $n$  of  $\mathbf{S}$  is denoted by  $|\mathbf{S}|$ . A structure  $\mathbf{S}[i..j] = \{\mathbf{s}_i, \mathbf{s}_{i+1}, \dots, \mathbf{s}_j\}$  ( $1 \leq i \leq j \leq n$ ) is called a substructure of  $\mathbf{S}$ .  $R \cdot \mathbf{S}$  denotes the structure  $\mathbf{S}$  rotated by the rotation matrix  $R$ , *i.e.*,  $R \cdot \mathbf{S} = \{R\mathbf{s}_1, R\mathbf{s}_2, \dots, R\mathbf{s}_n\}$ .  $\mathbf{v}^t$  denotes the transpose of the vector  $\mathbf{v}$  and  $A^T$  denotes the transpose of the matrix  $A$ .  $\text{trace}(A)$  denotes the trace of the matrix  $A$ .  $|\mathbf{v}|$  denotes the norm of the vector  $\mathbf{v}$ .  $\mathbf{0}$  denotes the zero vector.  $\langle x \rangle$  denotes the expected value of  $x$ .  $\text{var}(x)$  denotes the variance of  $x$ .  $\text{Prob}(X)$  denotes the probability of  $X$ .

In the rest of this paper, we consider that the target database  $\mathcal{D}$  consists of one long structure  $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N\}$ , and we let  $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_m\}$  denote the query structure, where  $m$  is supposed to be smaller than  $N$ . Our problem is to find all the positions  $i$  such that the RMSD (see section 2.2 for its definition) between  $\mathbf{P}[i..i+m-1]$  and  $\mathbf{Q}$  is at most a given fixed threshold  $c$ . An ordinary database may contain more than one structure, but the problem against such databases can be reduced to the problem against databases with only one structure, by concatenating all the database structures into one structure and ignoring substructures that cross over the boundaries of concatenated structures.

### 2.2 RMSD: The Root Mean Square Deviation

The RMSD (root mean square deviation) [1,7,12,13,16,19] between two 3-D coordinate sequences  $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$  and  $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n\}$  is defined as the minimum value of  $E_{R,\mathbf{v}}(\mathbf{S}, \mathbf{T}) = \sqrt{\frac{1}{n} \sum_{i=1}^n |\mathbf{s}_i - (R \cdot \mathbf{t}_i + \mathbf{v})|^2}$  over all the possible rotation matrices  $R$  and translation vectors  $\mathbf{v}$ . Let  $\text{RMSD}(\mathbf{S}, \mathbf{T})$  denote the minimum value, and let  $\hat{R}(\mathbf{S}, \mathbf{T})$  and  $\hat{\mathbf{v}}(\mathbf{S}, \mathbf{T})$  denote the rotation matrix and the translation vector that minimizes  $E_{R,\mathbf{v}}(\mathbf{S}, \mathbf{T})$ .  $\text{RMSD}(\mathbf{S}, \mathbf{T})$ ,  $\hat{R}(\mathbf{S}, \mathbf{T})$  and  $\hat{\mathbf{v}}(\mathbf{S}, \mathbf{T})$  can be computed in  $O(n)$  time as follows.

If the rotation matrix  $R$  is fixed,  $E_{R,\mathbf{v}}(\mathbf{S}, \mathbf{T})$  is known to be minimized when the centroid (center of mass) of  $R \cdot \mathbf{T}$  is translated to the centroid of  $\mathbf{S}$  by the translation vector  $\mathbf{v}$ , regardless of what the rotation matrix  $R$  is. It means that  $\hat{\mathbf{v}}(\mathbf{S}, \mathbf{T})$  can be computed in linear time if we are given  $\hat{R}(\mathbf{S}, \mathbf{T})$ . Moreover, it also means that the problem of computing the RMSD can be reduced to a problem of finding  $R$  (*i.e.*,  $\hat{R}(\mathbf{S}, \mathbf{T})$ ) that minimizes  $E'_R(\mathbf{S}, \mathbf{T}) = \sum_{i=1}^n |\mathbf{s}_i - R \cdot \mathbf{t}_i|^2$ , by translating both  $\mathbf{S}$  and  $\mathbf{T}$  so that both of their centroids are moved to the origin of the coordinates, which can be done in linear time.

After translating both structures so that both of their centroids are moved to the origin, we can compute  $\hat{R}(\mathbf{S}, \mathbf{T})$  in linear time as follows [1,12,13].<sup>1</sup> Let  $J = \sum_{i=1}^n \mathbf{s}_i \cdot \mathbf{t}_i^t$ . Clearly,  $J$  can be computed in  $O(n)$  time. Then  $E'_R(\mathbf{S}, \mathbf{T})$  can

<sup>1</sup> Here,  $\mathbf{S}$  and  $\mathbf{T}$  are the translated structures which are different from the original  $\mathbf{S}$  and  $\mathbf{T}$  in the original problem (that optimizes both rotation and translation).

be described as  $\sum_{i=1}^n (\mathbf{s}_i^t \mathbf{s}_i + \mathbf{t}_i^t \mathbf{t}_i) - 2 \cdot \text{trace}(R \cdot J)$ , and  $\text{trace}(R \cdot J)$  is maximized when  $R = VU^T$ , where  $UV$  is the singular value decomposition (SVD) of  $J$ . Thus  $\hat{R}(\mathbf{S}, \mathbf{T})$  can be obtained from  $J$  in constant time, as  $J$  is a  $3 \times 3$  matrix and the SVD can be computed in  $O(d^3)$  time for a  $d \times d$  matrix [1]. Note that there are degenerate cases where  $\det(VU^T) = -1$ , which means that  $VU^T$  is a reflection matrix. See [17, 12] for the details of the degenerate cases. Finally, we can compute the RMSD in linear time once we have obtained  $\hat{R}(\mathbf{S}, \mathbf{T})$ . In total, we can compute the RMSD in  $O(n)$  time.

### 2.3 Previous Best-Known Searching Algorithms

According to the previous section, we can compute the RMSD between any substructure  $\mathbf{P}[i..i+m-1]$  and the query  $\mathbf{Q}$  in  $O(m)$  time. Consequently, we can solve our problem in  $O(Nm)$  time by checking the RMSDs between the query and all the  $O(N)$  substructures of length  $m$  in the database.

Schwartz and Sharir [16] proposed a more sophisticated approach for the problem that solves it in  $O(N \log N)$  time, based on the convolution technique using the FFT (fast Fourier transform) [5]. Shibuya [19] also proposed a different algorithm with the same time complexity, also based on the convolution technique. These algorithms are not faster than the naive algorithm when  $N \gg m$ . But this time bound can be easily improved to  $O(N \log m)$  as follows. Break  $\mathbf{P}$  into  $O(N/m)$  substructures of length  $m + O(m)$  each of which overlaps with its adjacent fragment with overlap length  $m - 1$ . Then our problem can be solved in  $O(N \log m)$  time by applying the above  $O(N \log N)$ -time algorithm against each fragment. The expected time complexity of these algorithms are all the same as their worst-case time complexity, and no algorithm with better expected time complexity is known. But note that the above FFT-based  $O(N \log m)$ -time algorithm is not practically faster than the naive  $O(Nm)$ -time algorithm in case  $m$  is not large enough, and it is rarely used in practice.

For the problem, a linear-size indexing data structure called the geometric suffix tree [17, 18] is known to enable practically faster query than the above algorithms. But its worst-case query time complexity is still  $O(Nm)$ , while we need  $O(N^2)$  time to construct the data structure. In fact, there have been no known indexing algorithms whose theoretical query time complexity is smaller than the above  $O(N \log m)$  bound.

### 2.4 The Random-Walk Model for Chain Molecule Structures

The *random-walk model* for chain molecule structures is a simple but useful model for analyzing their behavior [4, 6, 9, 15]. The model is also called the *freely-jointed chain model* or the *ideal chain model*. In the model, we assume that the structure of a chain molecule is constructed as a result of a random walk in 3-D space. It is useful in various analyses in molecular physics, as it reflects properties of structures of real chain molecules very well [4].

Consider a chain molecule  $\mathbf{S} = \{\mathbf{s}_0, \mathbf{s}_2, \dots, \mathbf{s}_n\}$  of length  $n+1$ , in which the distance between two adjacent atoms is fixed to some constant  $\ell$ . Note that

the length between two adjacent  $C_\alpha$  atoms in a protein structure is constantly  $3.8\text{\AA}$ , as mentioned in section 1. In the random-walk model, a bond between two adjacent atoms, *i.e.*,  $\mathbf{b}_i = \mathbf{s}_{i+1} - \mathbf{s}_i$ , is considered as a random vector that satisfies  $|\mathbf{b}_i| = \ell$ , and  $\mathbf{b}_i$  is independent from  $\mathbf{b}_j$  for any  $i$  and  $j$  ( $i \neq j$ ). If  $n$  is large enough, the distribution of the end-to-end vector  $\mathbf{s}_n - \mathbf{s}_0$  is known to converge to the Gaussian distribution in 3-D space, in which  $\langle \mathbf{s}_n - \mathbf{s}_0 \rangle = \mathbf{0}$  and  $\langle |\mathbf{s}_n - \mathbf{s}_0|^2 \rangle = n \cdot \ell^2$ . In the distribution, the probability (or probability density) that  $\mathbf{s}_n - \mathbf{s}_0$  is located at some position  $(x, y, z)$  is  $W_{n,\ell}(x, y, z) dx dy dz = (\frac{3}{2\pi n \ell^2})^{\frac{3}{2}} e^{-3(x^2+y^2+z^2)/2n\ell^2} dx dy dz$ .

In our theoretical analysis in later sections, we will give an assumption that the structures in the target database follow the random-walk model. Our experimental results on the PDB database in section 6 show high consistency with our theoretical analyses based on the model.

### 3 An $O(N\sqrt{m})$ Algorithm

In this section, we propose an algorithm that will be the basis for our algorithms in later sections. The algorithm given in section 3.2 achieves  $O(N\sqrt{m})$  time under the random-walk assumption, by filtering out most of the dissimilar substructures (*i.e.*, substructures with RMSDs larger than the given threshold  $c$ ) before computing the actual RMSDs, based on a measure described in section 3.1. Its time complexity will be analyzed in section 3.3.

#### 3.1 An Efficiently-Computable Lower Bound for the RMSD

We first propose a nontrivial, but easily-computable lower bound for the RMSD between any two structures with the same length. Let  $\mathbf{U}^{left}$  denote  $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{\lfloor k/2 \rfloor}\}$  and  $\mathbf{U}^{right}$  denote  $\{\mathbf{u}_{\lfloor k/2 \rfloor + 1}, \mathbf{u}_{\lfloor k/2 \rfloor + 2}, \dots, \mathbf{u}_{2 \cdot \lfloor k/2 \rfloor}\}$  for a structure  $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$ . Let  $G(\mathbf{U})$  denote the centroid (center of mass) of the structure  $\mathbf{U}$ , *i.e.*,  $G(\mathbf{U}) = \frac{1}{k} \sum_{i=1}^k \mathbf{u}_i$ . Let  $F(\mathbf{U})$  denote  $|G(\mathbf{U}^{left}) - G(\mathbf{U}^{right})|/2$ , which means the half of the distance between the centroids of  $\mathbf{U}^{left}$  and  $\mathbf{U}^{right}$ . For any two structures  $\mathbf{S}$  and  $\mathbf{T}$  with the same length  $n$ , we define  $D(\mathbf{S}, \mathbf{T})$  as  $|F(\mathbf{S}) - F(\mathbf{T})|$  if  $n$  is an even integer. If  $n$  is an odd integer, we define  $D(\mathbf{S}, \mathbf{T})$  as  $\sqrt{\frac{n-1}{n}} |F(\mathbf{S}) - F(\mathbf{T})|$ . From now on, we prove that  $D(\mathbf{S}, \mathbf{T})$  is a lower bound of the RMSD between  $\mathbf{S}$  and  $\mathbf{T}$ .

Let  $\mathbf{s}'_i = \mathbf{s}_i - G(\mathbf{S})$ , and  $\mathbf{t}'_i = \mathbf{t}_i - G(\mathbf{T})$ . In case  $n$  is an even integer, we prove that  $D(\mathbf{S}, \mathbf{T})$  is always smaller than or equal to  $RMSD(\mathbf{S}, \mathbf{T})$ , as follows:

$$\begin{aligned} RMSD(\mathbf{S}, \mathbf{T}) &= \sqrt{\frac{1}{n} \sum_{i=1}^n |\mathbf{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \mathbf{t}'_i|^2} \geq \frac{1}{n} \sum_{i=1}^n |\mathbf{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \mathbf{t}'_i| \\ &\geq \frac{1}{n} \left| \sum_{i=1}^{n/2} \{\mathbf{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \mathbf{t}'_i\} \right| + \frac{1}{n} \left| \sum_{i=n/2+1}^n \{\mathbf{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \mathbf{t}'_i\} \right| \end{aligned}$$

$$\begin{aligned}
&= |G(\mathbf{S}^{left}) - G(\mathbf{S}) - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \{G(\mathbf{T}^{left}) - G(\mathbf{T})\}|/2 \\
&\quad + |G(\mathbf{S}^{right}) - G(\mathbf{S}) - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \{G(\mathbf{T}^{right}) - G(\mathbf{T})\}|/2 \\
&= |G(\mathbf{S}^{left}) - G(\mathbf{S}^{right}) - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \{G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})\}|/2 \\
&\geq |(|G(\mathbf{S}^{left}) - G(\mathbf{S}^{right})| - |G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})|)|/2 \\
&= D(\mathbf{S}, \mathbf{T}). \tag{1}
\end{aligned}$$

Note that we used the fact that  $G(\mathbf{S}) = \{G(\mathbf{S}^{left}) + G(\mathbf{S}^{right})\}/2$  and  $G(\mathbf{T}) = \{G(\mathbf{T}^{left}) + G(\mathbf{T}^{right})\}/2$  in the above.

In case  $n$  is an odd integer, we prove the same as follows:

$$\begin{aligned}
RMSD(\mathbf{S}, \mathbf{T}) &\geq \sqrt{\frac{n-1}{n}} RMSD(\mathbf{S}^-, \mathbf{T}^-) \\
&\geq \sqrt{\frac{n-1}{n}} D(\mathbf{S}^-, \mathbf{T}^-) = D(\mathbf{S}, \mathbf{T}), \tag{2}
\end{aligned}$$

where  $\mathbf{S}^-$  denotes  $\{s_1, s_2, \dots, s_{n-1}\}$ , and  $\mathbf{T}^-$  denotes  $\{t_1, t_2, \dots, t_{n-1}\}$ .

If the above lower bound can be computed very efficiently, we may solve our problem (presented in section II) more efficiently by filtering out hopelessly dissimilar substructures before computing the actual RMSD value. In fact, we can compute the above lower bound  $D(\mathbf{P}[i..i+m-1], \mathbf{Q})$  for all the positions  $i$  in linear time, as follows. For any  $m$ , we can compute  $G(\mathbf{P}[i..i+\lfloor m/2 \rfloor - 1])$  for all the positions  $i$  such that  $1 \leq i \leq N - \lfloor m/2 \rfloor + 1$  in  $O(N)$  time, as  $G(\mathbf{P}[i..i+\lfloor m/2 \rfloor - 1]) = G(\mathbf{P}[i-1..i+\lfloor m/2 \rfloor - 2]) - \frac{1}{\lfloor m/2 \rfloor} (\mathbf{p}_{i-1} - \mathbf{p}_{i+\lfloor m/2 \rfloor - 1})$ . It means that  $F(\mathbf{P}[i..i+m-1], \mathbf{Q})$  and consequently  $D(\mathbf{P}[i..i+m-1], \mathbf{Q})$  can be computed for all the positions  $i$  such that  $1 \leq i \leq N - m + 1$ , also in  $O(N)$  time.

### 3.2 The Algorithm

Our basic algorithm is simple. It uses the above lower bound to filter out some (hopefully most) of the substructures in the database before the time-consuming RMSD computation, as follows.

#### Algorithm 1

- 1 Compute  $D_i = D(\mathbf{P}[i..i+m-1], \mathbf{Q})$   
for all  $i$  such that  $1 \leq i \leq N - m + 1$ .
- 2 for (all  $i$  such that  $1 \leq i \leq N - m + 1$ ) {  
3     if ( $D_i \leq c$ ) {  
4         if ( $RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c$ )  
5             { output("P[i..i+m-1] is similar to the query Q.") }  
6     }  
7 }

The above algorithm is valid, *i.e.*, it enumerates all the positions of the substructures whose RMSDs to the query are at most  $c$ , because  $D_i = D(\mathbf{P}[i..i+m-1], \mathbf{Q})$  is always smaller than or equal to  $RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q})$ . Let the number of times of the RMSD computation in line 4 be  $N' (\leq N)$ . Then, the time complexity of the above algorithm is  $O(N + N'm)$ , as the line 1 of the algorithm requires only  $O(N)$  time according to the discussion in section 3.1. In the next section, we will prove that  $\langle N' \rangle$  is in  $O(N/\sqrt{m})$ , if the structures in the database follow the random-walk model.

### 3.3 Computational Time Analysis

Consider a structure  $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{2n}\}$  of length  $2n$  that follows the random-walk model. In this section, we let the distance between two adjacent atoms ( $\ell$  in section 2.4) be 1 without loss of generality, *i.e.*, we consider the distance between two adjacent atoms as the unit of distance. Then  $\mathbf{s}_i$  can be represented as  $\mathbf{s}_1 + \sum_{j=1}^{i-1} \mathbf{b}_j$ , where  $\mathbf{b}_i$  is an independent random vector that satisfies  $|\mathbf{b}_i| = 1$ . Let  $H(\mathbf{S}) = G(\mathbf{S}^{left}) - G(\mathbf{S}^{right})$ . Notice that  $F(\mathbf{S}) = |H(\mathbf{S})/2|$ . Then the following equation holds:

$$\begin{aligned} H(\mathbf{S}) &= \frac{1}{n} \sum_{i=1}^n (\mathbf{s}_1 + \sum_{j=1}^{i-1} \mathbf{b}_j) - \frac{1}{n} \sum_{i=n+1}^{2n} (\mathbf{s}_1 + \sum_{j=1}^{i-1} \mathbf{b}_j) \\ &= -\left\{ \sum_{i=1}^n \frac{i}{n} \cdot \mathbf{b}_i + \sum_{i=n+1}^{2n-1} \frac{2n-i}{n} \cdot \mathbf{b}_i \right\}. [1mm] \end{aligned} \quad (3)$$

Let  $\mathbf{b}'_i$  denote  $\frac{i}{n} \cdot \mathbf{b}_i$  if  $i \leq n$  and  $\frac{2n-i}{n} \cdot \mathbf{b}_i$  if  $i > n$ . Then  $H(\mathbf{S})$  can be described as  $\sum_{i=1}^{2n} \mathbf{b}'_i$ . Let  $z_i$  denote the  $z$  coordinate of  $\mathbf{b}_i$  and  $z'_i$  denote the  $z$  coordinate of  $\mathbf{b}'_i$ . It is easy to see that  $\langle z_i \rangle = 0$  and  $var(z_i) = 1/3$ , as  $\mathbf{b}_i$  is a random vector that satisfies  $|\mathbf{b}_i| = 1$ . Let  $M_n = \sum_{i=1}^{2n} \langle |z'_i| \rangle^{2+\delta} / \sqrt{\sum_{i=1}^{2n} var(z'_i)}^{2+\delta}$ , where  $\delta$  is some positive constant. According to Lyapunov's central limit theorem [14], the distribution of  $\sum_i^{2n} z'_i$  converges to the Gaussian distribution, if  $M_n$  converges to 0 as  $n$  grows up to infinity for some  $\delta$  such that  $\delta > 0$ . It can be proved as follows:

$$\begin{aligned} M_n &= \sum_{i=1}^{2n} \langle |z'_i| \rangle^{2+\delta} / \sqrt{\sum_{i=1}^{2n} \langle |z'_i| \rangle^2}^{2+\delta} \leq \sum_{i=1}^{2n} \langle |z'_i| \rangle^2 / \sqrt{\sum_{i=1}^{2n} \langle |z'_i| \rangle^2}^{2+\delta} \\ &= \left\{ \sum_{i=1}^{2n} \langle |z'_i| \rangle^2 \right\}^{-\delta/2} = \left\{ \frac{1}{3n^2} \cdot \left\{ \sum_{i=1}^n i^2 + \sum_{i=n+1}^{2n-1} (2n-i)^2 \right\} \right\}^{-\delta/2} \\ &= \left\{ \frac{2}{9}n + \frac{1}{9n} \right\}^{-\delta/2} \rightarrow 0 \quad (n \rightarrow \infty). [2mm] \end{aligned} \quad (4)$$

Hence, we conclude that  $\sum_i^{2n} z'_i$  converges to the Gaussian distribution. It also means that  $H(\mathbf{S})$  converges to the Gaussian distribution in 3-D space if  $n$

grows up to infinity, as the same discussion can be done for the other two axes ( $x$  and  $y$ ). The variance of  $H(\mathbf{S})$  is:

$$\begin{aligned} \text{var}(H(\mathbf{S})) &= \langle |H(\mathbf{S})|^2 \rangle - \langle |H(\mathbf{S})| \rangle^2 = \langle |H(\mathbf{S})|^2 \rangle \\ &= \left\langle \left| \sum_{i=1}^n \frac{i}{n} \cdot \mathbf{b}_i + \sum_{i=n+1}^{2n-1} \frac{2n-i}{n} \cdot \mathbf{b}_i \right|^2 \right\rangle \\ &= \frac{1}{n^2} \left\{ \sum_{i=1}^n i^2 + \sum_{i=n+1}^{2n-1} (2n-i)^2 \right\} = \frac{2}{3}n + \frac{1}{3n} \approx \frac{2}{3}n, [1mm] \quad (5) \end{aligned}$$

as  $\langle \mathbf{b}_i \cdot \mathbf{b}_j \rangle = 0$  if  $i \neq j$ . Moreover, it is easy to see that  $\langle H(\mathbf{S}) \rangle = \mathbf{0}$ . Thus the distribution of  $H(\mathbf{S})$  is the same as the distribution of random walks of length  $2n/3$ . Hence the probability distribution of  $H(\mathbf{S})$  is  $Z_n(x, y, z) dx dy dz = (\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9(x^2+y^2+z^2)/4n} dx dy dz$ . Consequently, the probability (or probability density) that  $|H(\mathbf{S})| = r$  is  $Z_n(r) dr = 4\pi r^2 (\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9r^2/4n} dr$ . Integrating  $Z_n(r) dr$ , we obtain  $\text{Prob}(x \leq |H(\mathbf{S})| \leq y) = \int_{r=x}^y Z_n(r) dr$ .  $Z_n(r)$  takes the maximum value at  $r_{max} = \frac{2}{3}\sqrt{n}$  and  $Z_n(r_{max}) = 6e^{-1}/\sqrt{\pi n}$ . Thus  $\text{Prob}(x \leq |H(\mathbf{S})| \leq y)$  is at most  $(y-x) \cdot Z_n(r_{max}) = 6e^{-1}(y-x)/\sqrt{\pi n}$  for any  $x$  and  $y$  ( $x < y$ ).

Therefore, for any structure  $\mathbf{T}$  such that  $|\mathbf{T}| = |\mathbf{S}| = 2n$ , the probability  $\text{Prob}(|D(\mathbf{S}, \mathbf{T})| \leq c) = \text{Prob}(F(\mathbf{T}) - c \leq F(\mathbf{S}) \leq F(\mathbf{T}) + c) = \text{Prob}(2 \cdot F(\mathbf{T}) - 2c \leq |H(\mathbf{S})| \leq 2 \cdot F(\mathbf{T}) + 2c)$  is at most  $4 \cdot c \cdot Z_n(r_{max}) = 24c \cdot e^{-1}/\sqrt{\pi n}$ , which is in  $O(1/\sqrt{n})$  as  $c$  is a fixed constant. Notice that there is no assumption on the structure  $T$  in this analysis.

Consequently, as  $\sqrt{\frac{m-1}{m}} \approx 1$ , the probability  $\text{Prob}(D_i \leq c)$  in the line 3 of the algorithm in section 3.2 is in  $O(1/\sqrt{m})$  if  $\mathbf{P}$  follows the random-walk model, no matter what the query structure  $\mathbf{Q}$  is. It means that  $\langle N' \rangle$  is in  $O(N/\sqrt{m})$ . Therefore, we conclude that the expected time complexity of the algorithm is  $O(N + \langle N' \rangle \cdot m) = O(N\sqrt{m})$ , under the assumption that the structures in the database follow the random-walk model.<sup>2</sup> Note that the worst-case time complexity of the algorithm is still  $O(Nm)$  as  $N'$  can be in  $O(N)$  at worst, but it should be rare under the random-walk assumption.

## 4 The Linear-Time Algorithm

We next improve the algorithm 1 to obtain better expected time complexity. From the definition of the RMSD, we can deduce that

$$\begin{aligned} \text{RMSD}(\mathbf{S}, \mathbf{T}) &\geq [2mm]\sqrt{t} \cdot \{( \text{RMSD}(\mathbf{S}^{left}, \mathbf{T}^{left}))^2 + \\ &\quad ( \text{RMSD}(\mathbf{S}^{right}, \mathbf{T}^{right}))^2 \}^{1/2} \\ &\geq \sqrt{t} \cdot \{ (D(\mathbf{S}^{left}, \mathbf{T}^{left}))^2 + (D(\mathbf{S}^{right}, \mathbf{T}^{right}))^2 \}^{1/2}, \quad (6) \end{aligned}$$

<sup>2</sup> The same discussion can be done in case the query structures, instead of the database structures, follow the random-walk model.

where  $t = |\mathbf{S}^{\text{left}}|/|\mathbf{S}| = |\mathbf{S}^{\text{right}}|/|\mathbf{S}| \approx 1/2$ . Let  $D^{\text{left}}(\mathbf{S}, \mathbf{T}) = \sqrt{t} \cdot D(\mathbf{S}^{\text{left}}, \mathbf{T}^{\text{left}})$  and  $D^{\text{right}}(\mathbf{S}, \mathbf{T}) = \sqrt{t} \cdot D(\mathbf{S}^{\text{right}}, \mathbf{T}^{\text{right}})$ . The expression (6) can also be used as a lower bound of the RMSD for another valid filtering algorithm, as follows:

### Algorithm 2

```

1   For all  $i$  such that  $1 \leq i \leq N - m + 1$ , compute
    
$$D'_i = \{(D^{\text{left}}(\mathbf{P}[i..i+m-1], \mathbf{Q}))^2 +$$

           
$$(D^{\text{right}}(\mathbf{P}[i..i+m-1], \mathbf{Q}))^2\}^{1/2}.$$

2   for (all  $i$  such that  $1 \leq i \leq N - m + 1$ ) {
3       if ( $D'_i \leq c$ ) {
4           if ( $\text{RMSD}(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c$ )
5               { output("P[i..i+m-1] is similar to the query Q.") }
6       }
7   }
```

The only difference from the Algorithm 1 is the lower bound  $D'_i$  used in the line 1. Note that the time complexity of the line 1 is still  $O(N)$ .

If  $D'_i \leq c$  in line 3, both  $D^{\text{left}}(\mathbf{P}[i..i+m-1], \mathbf{Q})$  and  $D^{\text{right}}(\mathbf{P}[i..i+m-1], \mathbf{Q})$  must also be at most  $c$ . According to the discussion in section 3.3, the two probabilities  $\text{Prob}(D^{\text{left}}(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c)$  and  $\text{Prob}(D^{\text{right}}(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c)$  are independent and both in  $O(1/\sqrt{m})$ , under the assumption that  $\mathbf{P}$  follows the random-walk model. Thus,  $\text{Prob}(D'_i \leq c)$  in line 3 must be in  $O((1/\sqrt{m})^2) = O(1/m)$ . Therefore the expected number of RMSD computations in line 4 should be in only  $O(N/m)$ , and consequently the expected time complexity spent in the lines 4–6 of the above algorithm is  $O(N)$ . Thus, the total expected time complexity of the algorithm 2 is  $O(N)$  under the random-walk assumption. Note that the worst-case time complexity is still  $O(Nm)$ , but it should be very rare under the random-walk assumption.

## 5 Faster Queries after Preprocessing

### 5.1 Preprocessing for Queries of a Fixed Length

From now on, we further improve the query time complexity by allowing preprocessing on the database structure  $\mathbf{P}$ . First, we consider the case where each query has the same length  $m$ . According to section 3.1, we can compute  $F(\mathbf{P}[i..i+w-1])$  for all  $i$  in  $O(N)$  time for a fixed value of  $w$ . Let  $L_w$  be the sorted list of  $i$  according to the value of  $F(\mathbf{P}[i..i+w-1])$ , which can be obtained in  $O(N \log N)$  time. By doing a binary search on  $L_w$ , we can find all the  $i$  such that  $x \leq F(\mathbf{P}[i..i+w-1]) \leq y$  in  $O(\log N + occ)$  time for any  $x$  and  $y$ , where  $occ$  is the number of the outputs. Hence, we can list all the  $i$  such that  $D(\mathbf{S}, \mathbf{P}[i..i+w-1]) \leq c$  in  $O(\log N + occ)$  time for any structure  $\mathbf{S}$  of length  $w$  by utilizing  $L_w$ , where  $c$  is some constant and  $occ$  is the number of outputs, as  $F(\mathbf{S}) - c \leq F(\mathbf{P}[i..i+w-1]) \leq F(\mathbf{S}) + c$  if  $D(\mathbf{S}, \mathbf{P}[i..i+w-1]) \leq c$ . Our preprocessing algorithm in this section computes  $F(\mathbf{P}[i..i+m'-1])$  for all  $i$  and

sorts them to obtain  $L_{m'}$ , where  $m' = \lfloor m/3 \rfloor$ , which can be done in  $O(N \log N)$  time in total.

Consider yet another lower bound for the RMSD, as follows:<sup>3</sup>

$$\begin{aligned}
D''_i &= \sqrt{\frac{m'}{m}} \cdot \left\{ \sum_{j=1}^3 (D(\mathbf{P}[i + (j-1) \cdot m'..i + j \cdot m' - 1], \right. \\
&\quad \left. \mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m'])^2 \right\}^{1/2} \\
&\leq \sqrt{\frac{m'}{m}} \cdot \left\{ \sum_{j=1}^3 (RMSD(\mathbf{P}[i + (j-1) \cdot m'..i + j \cdot m' - 1], \right. \\
&\quad \left. \mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m'])^2 \right\}^{1/2} \\
&\leq RMSD(\mathbf{P}[i..i + m - 1], \mathbf{Q}). \tag{7}
\end{aligned}$$

Notice that  $D(\mathbf{P}[i + (j-1) \cdot m'..i + j \cdot m' - 1], \mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m']) \leq c\sqrt{\frac{m}{m'}}$  for any  $j$ , if  $D''_i \leq c$ . Let  $X_j$  be the set of all positions  $i$  such that  $D(\mathbf{P}[i + (j-1) \cdot m'..i + j \cdot m' - 1], \mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m']) \leq c\sqrt{\frac{m}{m'}}$  (for  $1 \leq j \leq 3$ ). By using  $L_{m'}$ , we can find all  $i \in X_j$  in  $O(\log N + |X_j|)$  time for any of  $j = 1, 2, 3$  after we have computed  $F(\mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m'])$ . Note that  $F(\mathbf{Q}[1 + (j-1) \cdot m'..j \cdot m'])$  for all of  $j = 1, 2, 3$  can be computed in  $O(m)$  time. Let  $X$  be the set of common integers of the three sets  $X_1$ ,  $X_2$ , and  $X_3$ .  $X$  can be obtained in expected  $O(|X_1| + |X_2| + |X_3|)$  time by using a hashing technique. Notice that the positions  $i$  such that  $RMSD(\mathbf{P}[i..i + m - 1], \mathbf{Q}) \leq c$  must be included in  $X$ . For substructures at the positions  $i \in X$ , we finally have to compute the RMSD to check whether the actual RMSD is at most  $c$ , if  $D''_i \leq c$ . It can be done in at most  $O(m \cdot |X|)$  time.

$\langle |X_j| \rangle$  is in  $O(N/\sqrt{m})$  under the assumption that  $\mathbf{P}$  follows the random-walk model. Moreover, as the structures  $\mathbf{P}[i + (j-1) \cdot m'..i + j \cdot m' - 1]$  with different  $j$  are independent random walks,  $\langle |X| \rangle$  is estimated as  $O(N/(\sqrt{m})^3) = O(N/m^{1.5})$ . Thus the total expected query time complexity utilizing  $L_{m'}$  is  $O(m + N/\sqrt{m} + m \cdot N/m^{1.5} + \log N) = O(m + N/\sqrt{m})$ .

## 5.2 Preprocessing for Queries of Arbitrary Lengths

We next consider queries of arbitrary lengths. For such queries, consider computing  $L_w$  for all  $w$  such that  $w$  is a power of 2, *i.e.*, representable as  $2^d$  for some integer  $d$ . They can be obtained in  $O(N \log^2 N)$  time, as the number of different  $w$  is in  $O(\log N)$ . Let  $m'$  be the largest power of 2 such that  $3m' \leq m$ . Then the inequality (7) also holds for this case. The only difference is that  $m'$  is some power of 2 that satisfies  $m/6 < m' \leq m/3$ , while  $m' = \lfloor m/3 \rfloor$  in the previous section. Thus, according to the same discussion as in the previous section, we obtain the same query time complexity, *i.e.*,  $O(m + N/\sqrt{m})$ . A problem is that

<sup>3</sup> We can consider another linear-expected time algorithm based on the Algorithms 1 and 2 by replacing  $D_i$  or  $D'_i$  with  $D''_i$ . We will examine the performance of the algorithm (denoted as A3 in section 6) through computational experiments in section 6.

the algorithm requires  $O(N \log N)$  space to store all the  $L_w$ , which might be undesired for huge databases.

### 5.3 Preprocessing with Linear Space for Queries of Arbitrary Lengths

In this section, we propose another preprocessing algorithm that uses only  $O(N)$  space for queries of arbitrary lengths. Consider dividing  $\mathbf{P}$  into substructures of length  $2^d$  for each  $d$  such that  $1 \leq d \leq \log_2 N$ . By doing so, we get substructures  $\mathbf{P}^{k,d} = \mathbf{P}[(k-1) \cdot 2^d + 1..k \cdot 2^d]$  ( $1 \leq k \leq N/2^d$ ) for each  $d$ . There are only  $O(N)$  number of substructures denoted as  $\mathbf{P}^{k,d}$ , even if we enumerate all the possible  $k$  and  $d$ .  $G(\mathbf{P}^{k,1})$  (see section 3 for its definition) can be computed in constant time for each  $k$ . Moreover,  $G(\mathbf{P}^{k,d}) = \{G(\mathbf{P}^{2k-1,d-1}) + G(\mathbf{P}^{2k,d-1})\}/2$ . Thus, we can compute  $G(\mathbf{P}^{k,d})$  for all the possible  $k$  and  $d$  such that  $1 \leq k \leq N/2^d$  and  $1 \leq d \leq \log_2 N$  in  $O(N)$  time by dynamic programming. Consequently, all of the  $F(\mathbf{P}^{k,d})$  values can also be computed in  $O(N)$  time. For each  $d$  ( $1 \leq d \leq \log_2 N$ ), let  $K_d$  be the sorted list of integers  $k$  ( $1 \leq k \leq N/2^d$ ) according to the  $F(\mathbf{P}^{k,d})$  values.  $K_d$  can be computed in  $O((N \log N)/2^d)$  time. Our preprocessing algorithm in this section computes all these  $F(\mathbf{P}^{k,d})$  and  $K_d$  for all the  $k$  and  $d$ , which can be done in  $O(N \log N)$  time in total.

By doing a binary search on  $K_d$ , we can find all the  $k$  such that  $x \leq F(\mathbf{P}^{k,d}) \leq y$  for any  $x$ ,  $y$ , and  $d$ , in  $O(\log(N/2^d) + occ)$  time, where  $occ$  is the number of the outputs. Hence, if we are given any structure  $\mathbf{S}$  of length  $2^d$  and the value of  $F(\mathbf{S})$ , we can list all the  $k$  such that  $D(\mathbf{S}, \mathbf{P}^{k,d}) \leq c$  in  $O(\log(N/2^d) + occ)$  time, as  $F(\mathbf{S}) - c \leq F(\mathbf{P}^{k,d}) \leq F(\mathbf{S}) + c$  iff  $D(\mathbf{S}, \mathbf{P}^{k,d}) \leq c$ . Let  $d_Q$  be the largest  $d$  such that, for any  $i$ , there exists some  $k$  such that  $\mathbf{P}^{k,d}$ ,  $\mathbf{P}^{k+1,d}$  and  $\mathbf{P}^{k+2,d}$  are substructures of  $\mathbf{P}[i..i+m-1]$ . Explicitly,  $d_Q = \lfloor \log_2(m+1) \rfloor - 2$ . Let  $w_Q = |\mathbf{P}^{k,d_Q}| = 2^{d_Q}$ . Notice that  $w_Q > m/8$ . Let  $I_p^Q$  be a set of integers whose remainder is  $p-1$  when divided by  $w_Q$  ( $1 \leq p \leq w_Q$ ), i.e., integers representable as  $p+j \cdot w_Q + 1$  with some integer  $j$ .

Now consider comparing the query  $\mathbf{Q}$  and substructures  $\mathbf{P}[i..i+m-1]$  such that  $i \in I_p^Q$ . There are  $O(N/w_Q) = O(N/m)$  such substructures. Let  $k_Q = \lceil (i-1)/w_Q \rceil + 1$ . Then,  $\mathbf{P}^{k_Q,d_Q}$ ,  $\mathbf{P}^{k_Q+1,d_Q}$ , and  $\mathbf{P}^{k_Q+2,d_Q}$  are substructures of  $\mathbf{P}[i..i+m-1]$ . Let  $\mathbf{P}_{Q,i,j} = \mathbf{P}^{k_Q+j-1,d_Q}$  for  $j = 1, 2$ , and  $3$ . Let  $\mathbf{Q}_{p,j} = \mathbf{Q}[p+(j-1) \cdot w_Q..p+j \cdot w_Q-1]$  for  $j = 1, 2$ , and  $3$ , and let  $D_i^* = \frac{1}{2\sqrt{2}} \{ \sum_{j=1}^3 (D(\mathbf{P}_{Q,i,j}, \mathbf{Q}_{p,j}))^2 \}^{1/2}$ . Then,  $D_i^*$  is also a lower bound of  $RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q})$ , as shown in the following inequality:

$$D_i^* \leq \frac{1}{2\sqrt{2}} \{ \sum_{j=1}^3 (D(\mathbf{P}_{Q,i,j}, \mathbf{Q}_{p,j}))^2 \}^{1/2} < RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q}). \quad (8)$$

Notice that  $D(\mathbf{P}_{Q,i,j}, \mathbf{Q}_{p,j}) < 2\sqrt{2}c$  for any  $j$ , if  $D_i^* < c$ . Given the value of  $F(\mathbf{Q}_{p,j})$ , we can list all  $i \in I_p^Q$  such that  $D(\mathbf{P}_{Q,i,j}, \mathbf{Q}_{p,j}) < 2\sqrt{2}c$  in  $O(\log(N/m) + occ)$  time for any  $j$ , by a binary search on the list  $K_{d_Q}$ , where  $occ$  is the number of the  $i$  to be listed. Let the list be  $Y_j$  ( $j = 1, 2, 3$ ). Note that  $F(\mathbf{Q}_{p,j})$  for all  $j$  and  $p$  ( $1 \leq j \leq 3, 1 \leq p \leq w_Q$ ) can be computed in  $O(m)$  time in total.

Then the same discussion as in section 5.1 can be done. According to the discussions in section 3.3,  $\langle |Y_j| \rangle$  is in  $O((N/m)/\sqrt{m}) = O(N/m^{1.5})$ , under the random-walk assumption on  $\mathbf{P}$ . The set of the start positions  $i \in I_p^{\mathbf{Q}}$  of similar (*i.e.*, the corresponding RMSD is at most  $c$ ) substructures must be included in all of the three lists:  $Y_1$ ,  $Y_2$  and  $Y_3$ . Thus, the next thing to do is to choose the common positions from the three lists. By using a hashing technique, it can be done in  $O(|Y_1| + |Y_2| + |Y_3|)$  time, which is  $O(N/m^{1.5})$  under the random-walk assumption. Let  $Y$  denote the list of the positions commonly listed in  $Y_1$ ,  $Y_2$  and  $Y_3$ . As  $\mathbf{P}_{\mathbf{Q},i,1}$ ,  $\mathbf{P}_{\mathbf{Q},i,2}$  and  $\mathbf{P}_{\mathbf{Q},i,3}$  are independent random walks,  $\langle |Y| \rangle$  is estimated to be in  $O((N/m)/(\sqrt{m})^3) = O(N/m^{2.5})$ . For substructures at the positions  $i \in Y$ , we finally have to compute the RMSD to check whether the actual RMSD is at most  $c$ , if  $D_i^* < c$ . It takes at most  $O(m \cdot \langle |Y| \rangle) = O(N/m^{1.5})$  expected time under the random-walk assumption. Thus the total computational time to enumerate all the positions  $i$  of similar substructures such that  $i \in I_p^{\mathbf{Q}}$  is  $O(N/m^{1.5} + \log(N/m))$ .

To enumerate all the positions of similar structures, we execute the above for all  $p$  ( $1 \leq p \leq w_{\mathbf{Q}}$ ). Thus the total expected query time complexity is  $O(m + w_{\mathbf{Q}} \cdot (N/m^{1.5} + \log(N/m))) = O(N/\sqrt{m} + m \log(N/m))$  under the random-walk assumption.

## 6 Computational Experiments on the PDB Database

We did computational experiments using the whole PDB database [3] of the date September 5th, 2008. Note that more detailed results will be given in the full version of this paper. The database contains 52,821 entries, which include 244,719 chains of proteins. The total number of amino acids of all the chains is 38,267,694. We used the  $C_{\alpha}$  coordinates as the representative coordinates of each amino acid. We did experiments of searching queries of 10 different lengths. In each experiment, we selected 100 substructures of each specified length randomly from the whole database, as sample queries. We used 1 CPU of 1200MHz UltraSPARC III Cu on a SunFire 15K super computer for each experiment.

Table 1 shows the results. Each column shows the result of the queries of each specified length. The ‘#Substructures’ row shows the number of substructures of each specified length in the PDB database. The ‘#Hits’ row shows the average number of hits, *i.e.*, the average number of substructures whose RMSDs to queries are at most 1Å, among the results of the 100 random queries of each specified length. The ‘A1’, ‘A2’, ‘A3’, ‘Naive’, and ‘FFT’ rows show the average computation time for the 5 algorithms: A1: the  $O(N\sqrt{m})$ -time algorithm proposed in section 3, A2: the linear-time algorithm proposed in section 4, A3: another linear-time algorithm that uses the lower bound  $D''_i$  proposed in section 5.1, Naive: the previously known standard  $O(Nm)$ -time algorithm, and FFT: the previously known  $O(N \log N)$ -time algorithm based on the FFT (introduced in section 2.3). The algorithms A1,

**Table 1.** Experimental results over the PDB database for various-length queries

Query length	20	40	60	80	100
#Substructures	33,722,208	29,299,006	25,273,633	21,692,707	18,634,620
#Hits	15,093.64	38.07	27.36	32.90	28.61
A1 (sec)	119.87	98.94	98.94	92.43	85.80
A2 (sec)	117.39	58.86	44.01	36.41	36.25
A3 (sec)	151.52	74.56	33.63	25.54	20.46
Naive (sec)	423.52	447.01	450.39	442.13	428.06
FFT (sec)	551.94	531.92	505.52	463.01	425.77

Query length	120	140	160	180	200
#Substructures	16,134,096	14,084,515	12,362,509	10,884,548	9,559,056
#Hits	27.26	27.71	16.01	17.70	23.21
A1 (sec)	75.58	71.22	59.47	63.48	59.98
A2 (sec)	32.84	30.39	27.27	23.12	25.71
A3 (sec)	17.30	15.85	14.25	12.78	12.91
Naive (sec)	415.24	395.54	378.87	361.43	342.50
FFT (sec)	399.83	367.76	330.57	307.89	293.03

A2 and A3 are all the same algorithms, except for the lower bounds used in them.

In the experiments, we achieved 3.6 to 28 times speed-up against any of the previous algorithms for any-length queries, if we choose to use the lower bound  $D''_i$  when the query is longer than 40, and choose  $D'_i$  otherwise. Moreover, the experiments show that our linear-expected-time algorithms run actually in linear time on the PDB database, *i.e.*, the algorithm is not influenced by the difference of query lengths.<sup>4</sup> It means our random-walk assumption is very reasonable for analyses of protein structure databases.

## 7 Concluding Remarks

We proposed the first linear-expected-time algorithm for searching similar substructures from structure databases, based on the RMSD measure. Moreover, we proposed several preprocessing algorithms that enable theoretically even faster queries. The performance of our algorithms is examined by computational experiments on the whole PDB database.

As for the future work, it would be very interesting to apply our techniques against protein alignment problems that consider insertions and deletions, though it is known to be theoretically much more difficult. Another challenging task would be to design a deterministically linear-time algorithm for our problem. It is also very interesting to extend our techniques against similar problems

<sup>4</sup> Notice that the numbers of substructures of some specified length in the database decreases as the length increases.

in higher dimensions. Moreover, our technique should be applicable to many problems in other research fields such as robotics and computer vision.

## Acknowledgement

The author would like to thank Jesper Jansson, Gregory Kucherov, and Ku-nihiko Sadakane for invaluable discussions. This work was partially supported by the Grant-in-Aid for Young Scientist (B) No. 20700264 from the Ministry of Education, Culture, Sports, Science and Technology of Japan. The author used the super computer system of the Human Genome Center, Institute of Medical Science, University of Tokyo.

## References

1. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Anal. Machine Intell.* 9, 698–700 (1987)
2. Aung, Z., Tan, K.-L.: Rapid retrieval of protein structures from databases. *Drug Discovery Today* 12, 732–739 (2007)
3. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. *Nucl. Acids Res.* 28, 235–242 (2000)
4. Boyd, R.H., Phillips, P.J.: The Science of Polymer Molecules: An Introduction Concerning the Synthesis. In: *Structure and Properties of the Individual Molecules That Constitute Polymeric Materials*. Cambridge University Press, Cambridge (1996)
5. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* 19, 297–301 (1965)
6. de Gennes, P.-G.: *Scaling Concepts in Polymer Physics*. Cornell University Press (1979)
7. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications* 9, 272–290 (1997)
8. Eidhammer, I., Jonassen, I., Taylor, W.R.: Structure comparison and structure patterns. *J. Comput. Biol.* 7(5), 685–716 (2000)
9. Flory, P.J.: *Statistical Mechanics of Chain Molecules*. Interscience, New York (1969)
10. Gerstein, M.: Integrative database analysis in structural genomics. *Nat. Struct. Biol.*, 960–963 (2000)
11. Golub, G.H., Van Loan, C.F.: *Matrix Computation*, 3rd edn. John Hopkins University Press (1996)
12. Kabsch, W.: A solution for the best rotation to relate two sets of vectors. *Acta Cryst. A*32, 922–923 (1976)
13. Kabsch, W.: A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Cryst. A*34, 827–828 (1978)
14. Kallenberg, O.: *Foundations of Modern Probability*. Springer, Heidelberg (1997)
15. Kramers, H.A.: The behavior of macromolecules in inhomogeneous flow. *J. Chem. Phys.* 14(7), 415–424 (1946)

16. Schwartz, J.T., Sharir, M.: Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *Intl. J. of Robotics Res.* 6, 29–44 (1987)
17. Shibuya, T.: Geometric suffix tree: a new index structure for protein 3-D structures. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 84–93. Springer, Heidelberg (2006)
18. Shibuya, T.: Prefix-shuffled geometric suffix tree. In: Ziviani, N., Baeza-Yates, R. (eds.) SPIRE 2007. LNCS, vol. 4726, pp. 300–309. Springer, Heidelberg (2007)
19. Shibuya, T.: Efficient substructure RMSD query algorithms. *J. Comput. Biol.* 14(9), 1201–1207 (2007)

# Optimization-Based Peptide Mass Fingerprinting for Protein Mixture Identification

Zengyou He<sup>1</sup>, Chao Yang<sup>1</sup>, Can Yang<sup>1</sup>, Robert Z. Qi<sup>2</sup>, Jason Po-Ming Tam<sup>2</sup>, and Weichuan Yu<sup>1</sup>

<sup>1</sup> Laboratory for Bioinformatics and Computational Biology,

Department of Electronic and Computer Engineering

<sup>2</sup> Department of Biochemistry

The Hong Kong University of Science and Technology, Hong Kong, China

{ezyhe,yorkey,eeyang,qirz,bctpm,eeyu}@ust.hk

**Abstract.** In current proteome research, the most widely used method for protein mixture identification is probably peptide sequencing. Peptide sequencing is based on tandem Mass Spectrometry (MS/MS) data. The disadvantage is that MS/MS data only sequences a limited number of peptides and leaves many more peptides uncovered.

Peptide Mass Fingerprinting (PMF) has been widely used to identify single purified proteins from single-stage MS data. Unfortunately, this technique is less accurate than the peptide sequencing method and can not handle protein mixtures, which hampers the widespread use of PMF.

In this paper, we tackle the problem of protein mixture identification from an optimization point of view. We show that some simple heuristics can find good solutions to the optimization problem. As a result, we obtain much better identification results than previous methods. Through a comprehensive simulation study, we identify a set of limiting factors that hinder the performance of PMF-based protein mixture identification. We argue that it is feasible to remove these limitations and PMF can be a powerful tool in the analysis of protein mixtures, especially in the identification of low-abundance proteins which are less likely to be sequenced by MS/MS scanning.

**Availability:** The source codes, data and supplementary documents are available at <http://bioinformatics.ust.hk/PMFMixture.rar>

## 1 Introduction

The identification of proteins expressed in a cell or tissue is an explicit goal of proteomics. Among existing protein identification strategies, peptide mass fingerprinting (PMF) has been widely used to identify single purified proteins since 1993 [1]. Popular methods include MASCOT [2], MS-Fit [3] and ProFound [4].

However, PMF fell out of favor in the emerging shotgun proteomics [5], which combines protein digestion and tandem MS (MS/MS) based peptide sequencing to perform peptide-centric identification of complex protein mixtures. The major

reason is that PMF cannot distinguish different peptides with identical mass, while the peptide sequencing method can identify peptides with the help of collision-induced dissociation and MS/MS scanning.

We also observe that it is still impossible for the peptide sequencing method to perform tandem mass spectrometry scanning on every single ion, leading to an incomplete identification of peptides from complex mixtures. So far, there is no good method yet available to identify the peptides uncovered by the peptide sequencing method.

We believe that PMF can serve as a supplement to the peptide sequencing method in analyzing complex protein mixtures, especially in the analysis of low-abundance proteins, whose peptide digestion results are less likely to be covered by the peptide sequencing method. Thus, our motivation is to extend the use of PMF from single protein identification to the identification of protein mixtures.

Some efforts have been taken in this regard. A *subtraction* strategy is proposed in [6]: The masses matching the protein identified in the first round are removed prior to the second round of searching. This procedure repeats a number of rounds until enough proteins are identified. The same strategy is also adopted in [7,8]. Though the *subtraction* approach can be used to identify component proteins from simple mixtures, it still suffers from the problem of random matching and has poor performance when the mixture is complex and noisy. The approach in [9] relies on a randomized decoy database and high mass accuracy capability of mass spectrometry. This method is still a traditional PMF method in which each single protein is ranked separately.

The main objective of this paper is to study the potentials and pitfalls of PMF in protein mixture identification. More concisely, we are interested in answering the following two questions:

1. Is it possible to achieve an acceptable identification accuracy in protein mixtures if we only use PMF on single-stage MS data? The answer to this question is *yes*. We first show that protein mixture identification is an optimization problem. While obtaining the optimal solution is difficult, we can employ some heuristic searching methods to find the local optima. As a result, the identification accuracy is significantly better than that of traditional PMF and *subtraction* strategy.
2. What are the limiting factors when applying PMF? Through a comprehensive simulation study, we show that the performance of PMF is mainly affected by the mass accuracy of mass spectrometer, the number of component proteins in the mixture, the sequence coverage of each protein and the noise level in MS data. With further improvements in MS instrumentation, protein and peptide separation techniques, and computational data analysis tools, it is possible to overcome these limitations.

The rest of the paper is organized as follows: Section 2 describes the problem formulation and introduces two new PMF algorithms. Section 3 shows the potential of our new methods and investigates some major limiting factors through experimental studies. Section 4 concludes this paper.

## 2 Methods

### 2.1 PMF for Single Protein Identification

PMF based single protein identification consists of the following steps:

1. Protein purification: 2D gel-based separation produces purified protein samples.
2. Protein digestion: Protease (such as trypsin) digests each protein into peptide mixtures and MS records the masses of resulting peptides.
3. Protein identification: PMF ranks each protein in the database according to its match quality with the MS spectrum and reports the best ones.

Most existing PMF algorithms require a user-specified mass tolerance threshold as input to define the peak matching relationship. The underlying assumption is that one experimental peak corresponds to a theoretical peak if their distance is not larger than the mass tolerance threshold. Suppose we have a set of peaks  $Z = (z_1, z_2, \dots, z_l)$  and a database of proteins  $D = (X_1, X_2, \dots, X_g)$ , where  $g$  denotes the size of the database. The quality of matching between a protein  $X_i \in D$  and the experimental peak set  $Z$  is measured by a scoring function  $S^{(L)}(Z, X_i, \sigma)$ , where  $\sigma$  is the mass tolerance threshold. When  $\sigma$  is fixed, we may use  $S^{(L)}(Z, X_i)$  instead of  $S^{(L)}(Z, X_i, \sigma)$ .

Assuming that the ground-truth protein has the highest score, the problem of single protein identification is an optimization problem:

$$\hat{X} = \arg \max_{X_i \in D} S^{(L)}(Z, X_i), \quad (1)$$

where  $\hat{X}$  is the protein with the highest score, i.e.,  $\hat{X}$  best “explains”  $Z$ .

### 2.2 PMF for Protein Mixture Identification

In the context of protein mixtures, we have the same input:  $Z = (z_1, z_2, \dots, z_n)$ . Here our objective is to find a set of proteins  $\hat{Y}$  that best “explains”  $Z$ :

$$\hat{Y} = \arg \max_{Y \subseteq D} S^{(M)}(Z, Y), \quad (2)$$

where  $Y$  denotes a subset of proteins and  $S^{(M)}(Z, Y)$  is a scoring function. Note that  $S^{(M)}(\cdot, \cdot)$  is different from  $S^{(L)}(\cdot, \cdot)$  as  $Y$  may have multiple proteins. The definition of  $S^{(M)}(\cdot, \cdot)$  will be discussed in the next section.

Obviously, single protein identification is a special case of protein mixture identification when an additional constraint  $|Y| = 1$  is provided.

### 2.3 Scoring Function for Protein Mixture Identification

**Generic Scoring Function.** To define  $S^{(M)}(Z, Y)$ , we have two choices:

1. Virtual single protein approach: We consider  $Y$  as a “virtual” single protein  $V$ , yielding the following relationship:

$$S^{(M)}(Z, Y) = S^{(L)}(Z, V). \quad (3)$$

2. Peak partition approach: We distribute peaks in  $Z$  to different proteins in  $Y$  explicitly. If we assume that each peak can only be assigned to one protein, then we need to partition the peaks of  $Z$  into disjoint subsets. Without loss of generality, we assume that  $Y$  consists of  $k$  proteins  $X_{u_1}, X_{u_2}, \dots, X_{u_k}$  ( $1 \leq u_j \leq g$ ) and divide  $Z$  into  $k$  disjoint subsets  $Z_1, Z_2, \dots, Z_k$ . Then the score is calculated as:

$$S^{(M)}(Z, Y) = \sum_{j=1}^k S^{(L)}(Z_j, X_{u_j}). \quad (4)$$

When the single protein identification method is applied directly to protein mixtures, the scoring function is actually  $S^{(L)}(Z, X_{u_j})$  in which protein  $X_{u_j}$  is used to match/explain all the peaks in  $Z$ . Clearly, this scheme will suffer from serious random matching.

The *subtraction* strategy [6] can be regarded as a special instance of peak partition approach in which peaks are divided in a greedy manner. Suppose the peak subsets  $Z_0, Z_1, Z_2, \dots, Z_k$  are generated by the *subtraction* strategy sequentially ( $Z_0$  is an empty set), then the score at each step  $j$  ( $1 \leq j \leq k$ ) is calculated as:

$$S^{(L)}(Z - \bigcup_{t=0}^{j-1} Z_t, X_{u_j}). \quad (5)$$

Although the *subtraction* strategy partitions the peaks into different groups, it evaluates each protein using a much larger subset of  $Z$  instead of its corresponding peak subset. Obviously, this strategy still suffers from the problem of random matching.

Here we use the virtual single protein approach to define the scoring function. This approach has the following advantages:

- ★ It is simple to understand and easy to implement.
- ★ The calculation of score can be very efficient in the optimization process if the single protein scoring function is properly selected.

**Materialized Scoring Function.** Some scoring functions such as Mascot and ProFound are commonly used. In our implementation, we choose the scoring function in [10] due to the following reasons:

1. It has comparable performance to Mascot and ProFound.
2. Its score can be calculated incrementally.

The empirical results in [10] support the first point and we will discuss the second point in the next section.

The basic idea of [10] is to consider a good match as an unlikely event. If one protein  $X_i$  matches  $r_i$  peaks in  $Z$ , the algorithm computes *a priori* random probability that these  $r_i$  matches occur. The score is taken as the negative logarithm of that probability. A high score value reflects an unlikely event, and hence a high degree of good matching.

There are different ways to compute the *a priori* random probability. One strategy is to apply the binomial distribution, i.e., the probability that a protein  $X_i$  has  $r_i$  random matched peaks in  $Z$  is given by:

$$\Pr(|M_Z(X_i)| = r_i) = C_l^{r_i} p_i^{r_i} (1 - p_i)^{l - r_i}, \quad (6)$$

where  $M_Z(X_i)$  denotes a subset of  $Z$  whose peaks match protein  $X_i$ ,  $l$  is the number of observed peaks in  $Z$ ,  $C_l^{r_i}$  is the binomial coefficient and  $p_i$  is the probability for at least one match between a peak from  $Z$  and one of the  $n_i$  peptide masses of protein  $X_i$ .

The value of  $p_i$  is calculated as [10]:

$$p_i = 1 - (1 - 2\sigma/\Delta)^{n_i}, \quad (7)$$

where  $\Delta$  is the acquisition mass range (i.e. the difference between the maximum and minimum mass values of  $Z$ ) and  $\sigma$  is the mass tolerance threshold.

The interpretation of  $p_i$  is straightforward: If we assume the probability of random match for an observed peak is  $2\sigma/\Delta$ , then the probability for a random miss is  $1 - 2\sigma/\Delta$ . If we draw  $n_i$  random peptides, then the probability of missing this observed peak in all  $n_i$  trials is  $(1 - 2\sigma/\Delta)^{n_i}$ . Therefore, the probability for at least one match is  $1 - (1 - 2\sigma/\Delta)^{n_i}$ .

Then, the scoring function is defined as the negative natural logarithm of  $\Pr(|M_Z(X_i)| = r_i)$ :

$$S^{(L)}(Z, X_i) = -\ln C_l^{r_i} - r_i \ln p_i - (l - r_i) \ln(1 - p_i). \quad (8)$$

Suppose  $Y$  consists of  $k$  proteins  $X_{s_1}, X_{s_2}, \dots, X_{s_k}$ , we can consider  $Y$  as a virtual single protein and the generalized scoring function becomes:

$$S^{(M)}(Z, Y) = -\ln C_l^{r_Y} - r_Y \ln p_Y - (l - r_Y) \ln(1 - p_Y), \quad (9)$$

where  $r_Y = |\bigcup_{j=1}^k M_Z(X_{u_j})|$  and  $p_Y = 1 - (1 - 2\sigma/\Delta)^{\sum_{j=1}^k n_{u_j}}$ .

## 2.4 Local Search Algorithms for Protein Mixture Identification

**Local Search Algorithm with Known  $k$  (Losak).** This section presents a local search algorithm with known  $k$  for protein mixtures. We name it Losak and describe the details in Algorithm II.

Losak takes the number of target proteins as input and iteratively improves the value of objective function. Initially, we randomly select  $k$  proteins and label them as “target” proteins. In the iteration process, for each protein labeled as “non-target” protein, its label is exchanged with each of the  $k$  target proteins and the objective value is re-evaluated. If the objective value increases, the protein’s “non-target” label is exchanged with the “target” label of the protein that achieves the best objective value and the algorithm proceeds to the next protein. When all “non-target” proteins have been checked for possible improvements, a full iteration is completed. If at least one label has been changed in one iteration,

**Algorithm 1.** Losak

---

**Input** :  $D$ : a database of  $g$  proteins;  $Z$ : observed peak list;  
 $\sigma$ : mass tolerance threshold;  $k$ : number of target proteins;

**Output**:  $Y$ : a set of  $k$  proteins;

```

/* ----- Phase 1-Initialization ----- */
1 Randomly select  $k$  proteins into  $Y$  as “target” proteins;
/* ----- Phase 2-Iteration ----- */
2 Initialize  $hasSwap \leftarrow True$ ;
3 while  $hasSwap=True$  do
4    $hasSwap \leftarrow False$ ;
5   for  $i = 1$  to  $g$  do
6     if  $X_i$  does not belong to  $Y$  then
7        $h \leftarrow \arg \max_j S^{(M)}(Z, Y + \{X_i\} - \{X_{u_j}\})$ ;
8       if  $S^{(M)}(Z, Y + \{X_i\} - \{X_{u_h}\}) > S^{(M)}(Z, Y)$  then
9          $Y \leftarrow Y + \{X_i\} - \{X_{u_h}\}$ ,  $hasSwap \leftarrow True$ ;
10  return  $Y$ 
```

---

we initiate a new iteration. The algorithm terminates when a full iteration does not change any labels, thereby indicating that a local optimum is reached.

In this algorithm, the key step is to efficiently calculate the new score when two proteins are swapped. Thanks to the good property of the scoring function in Equation (9), we can calculate the score incrementally:

When  $X_{u_h}$  in  $Y$  is swapped out and  $X_i$  is swapped in, we can store  $\sum_{j=1}^k n_{u_j}$  as a constant  $C$  so that  $p_Y = 1 - (1 - 2\sigma/\Delta)^{C-n_{u_h}+n_i}$ .

In order to calculate  $r_Y$  efficiently, we use an integer array  $A$  of length  $l$  to record the number of proteins in  $Y$  that hit each observed peak. Given  $A$ ,  $r_Y$  is calculated as the number of non-zero entries of  $A$ . When  $X_{u_h}$  is exchanged with  $X_i$ , we just need to use the non-overlapping elements of  $M_Z(X_i)$  and  $M_Z(X_{u_h})$  to update  $A$  and calculate the new  $r_Y$  value. Therefore,  $r_Y$  can be computed with a time complexity of  $O(l)$ .

After obtaining the values of  $p_Y$  and  $r_Y$ , we can calculate the score according to Equation (9) with a complexity of  $O(l)$ . In total, Losak has a time complexity of  $O(qgkl)$ , where  $q$  is the number of iterations,  $g$  is the number of proteins in the database,  $k$  is the number of target proteins, and  $l$  is the number of observed peaks. Obviously, Losak has a good scalability since its time complexity is linear to all major parameters.

**Local Search Algorithm with Unknown  $k$  (Losau).** Losau is an extension of Losak with details described in Algorithm 2. It has several salient features:

(1) It is adaptive. To determine the number of target proteins automatically, we introduce the “insert” operation and “delete” operation into the local optimization process. If protein  $X_i$  is contained in  $Y$ , we will delete it from  $Y$  if such an operation increases the score. Similarly, if protein  $X_i$  is not contained in  $Y$ ,

we will either insert it into  $Y$  or exchange it with another protein in  $Y$  if such an operation increases the score.

To achieve a simplicity-quality tradeoff, we introduce a penalty  $\omega$  on the insert operation to reflect our intention of “explaining”  $Z$  using as few number of proteins as possible. In the first iteration, we use the number of proteins in  $Y$  to update  $\omega$  value so that  $Y$  can be expanded to a reasonable size. In the subsequent iterations, we use the parameter  $df (0 < df < 1)$  as the decay factor to decrease  $\omega$  at each iteration:  $\omega \leftarrow df \cdot \omega$ .

(2) It has a filtering mechanism to remove false positives effectively. The adaptive nature of Losau may introduce many false positives into the final protein list. It is desirable to filter out these incorrect proteins. Meanwhile, it is also necessary to provide a significance-test-alike procedure to evaluate the confidence of each single protein. Algorithm 3 describes our filtering procedure.

In this algorithm, we evaluate each protein  $X_{u_j}$  using the peak subset  $M_Z(X_{u_j})$ . The idea is very simple: Since all the peaks in  $M_Z(X_{u_j})$  match  $X_{u_j}$ , the probability that other proteins in the database achieve better single protein identification score than  $X_{u_j}$  on  $M_Z(X_{u_j})$  is very low if  $X_{u_j}$  is the ground-truth protein. We use the number of “winning proteins” to measure the rank uncertainty and  $\theta$  as the threshold. If there are more than  $\theta$  proteins outperform  $X_{u_j}$  on  $M_Z(X_{u_j})$  in terms of single protein identification score, we remove it from the result set.

While we can also apply the filtering procedure to Losak, the performance gain is not as significant as that of Losau. The effectiveness of this filtering procedure in improving the overall performance is verified experimentally (see the supplement).

(3) It has the same time complexity as Losak. We can show that the time complexity of incremental score calculation when “insert” operation and “delete” operation are invoked is also  $O(l)$ . If there are at most  $k$  proteins contained in  $Y$  in the intermediate steps, then the time complexity of Losau is still  $O(qgk)$ .

## Convergence of Two Algorithms

**Theorem 1.** *Both Losak and Losau converge to a local maximal solution in a finite number of iterations.*

*Proof.* (1) There are only a finite number ( $2^g$ ) of possible subset of  $D$ ; (2) Each possible subset  $Y$  appears at most once since the sequence  $S^{(M)}(\cdot, \cdot)$  is strictly increasing in both algorithms. Hence, the result follows.  $\square$

Theoretically, the algorithms will converge to different local optimal solutions using different initializations. In our experiments, different starting points usually lead to similar results (please refer to the supplement for more details).

## 3 Experiments

In the experiments, we use standard performance metrics in information retrieval, including *precision*, *recall*, and *F1-measure*, to evaluate the identification performance. *Precision* is the proportion of identified ground-truth proteins

**Algorithm 2.** Losau

---

```

Input :  $D$ : a database of  $g$  proteins;  $Z$ : observed peak list;
         $\sigma$ : mass tolerance threshold;
         $df$ : decay factor;  $\theta$ : rank threshold in filtering;
Output:  $Y$ : a set of  $k$  proteins ; /*  $k$  is determined automatically */

/* ----- Phase 1-Initialization ----- */
1 Randomly select 2 proteins into  $Y$  as “target” proteins;
2 Initialize  $\omega \leftarrow 0$  and  $q \leftarrow 1$  /*  $\omega$ : penalty value;  $q$ : iteration number */

/* ----- Phase 2-Iteration ----- */
3 Initialize  $hasOperation \leftarrow True$ ;
4 while  $hasOperation=True$  do
5    $hasOperation \leftarrow False$ ;
6   if  $q > 1$  then  $\omega \leftarrow df \cdot \omega$ ;
7   for  $i = 1$  to  $g$  do
8      $\zeta_{noop} \leftarrow S^{(M)}(Z, Y)$ ;
9     if  $q = 1$  then  $\omega \leftarrow |Y|$ ;
10    if  $X_i \in Y$  then
11      if  $S^{(M)}(Z, Y - \{X_i\}) > \zeta_{noop}$  then
12         $Y \leftarrow Y - \{X_i\}$ ,  $hasOperation \leftarrow True$ ; /* Delete */
13    else
14       $h \leftarrow \arg \max_j S^{(M)}(Z, Y + \{X_i\} - \{X_{u_j}\})$ ;
15       $\zeta_{swap} \leftarrow S^{(M)}(Z, Y + \{X_i\} - \{X_{u_h}\})$ ;
16       $\zeta_{inst} \leftarrow S^{(M)}(Z, Y + \{X_i\}) - \omega$ ;
17      if  $\zeta_{swap} > \zeta_{inst}$  and  $\zeta_{swap} > \zeta_{noop}$  then
18         $Y \leftarrow Y + \{X_i\} - \{X_{u_h}\}$ ,  $hasOperation \leftarrow True$ ; /* Swap */
19      if  $\zeta_{inst} > \zeta_{swap}$  and  $\zeta_{inst} > \zeta_{noop}$  then
20         $Y \leftarrow Y + \{X_i\}$ ,  $hasOperation \leftarrow True$ ; /* Insert */
21    $q \leftarrow q + 1$ ;
/* ----- Phase 3-Filtering (See Algorithm 3) ----- */
22  $Y \leftarrow \text{ProteinFilter}(D, Z, \sigma, \theta, Y)$ ;
23 return  $Y$ 

```

---

**Algorithm 3.** PoteinFilter Algorithm

---

```

Input :  $D, Z, \sigma, \theta$ , and  $Y$  is a set of unfiltered proteins.
Output:  $F$ : a refined set of proteins,  $F \subseteq Y$ .

1 Initialize  $F \leftarrow \emptyset$ 
2 for  $j = 1$  to  $|Y|$  do
3   Initialize  $Winner \leftarrow 0$ 
4   for  $i = 1$  to  $g$  do
5     if  $S^{(L)}(M_Z(X_{u_j}), X_i) > S^{(L)}(M_Z(X_{u_j}), X_{u_j})$  then  $Winner ++$ 
6     if  $Winner < \theta$  then  $F \leftarrow F + \{X_{u_j}\}$ 
7 return  $F$ 

```

---

to all identified proteins. *Recall* is the proportion of identified ground-truth proteins to all ground-truth proteins. *F1-measure* is the harmonic mean of *recall* and *precision*:  $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ .

In evaluation, we compare Losak and Losau with the following algorithms:

- ★ SPA: Single protein identification algorithm with Equation (8) as the scoring function. Since SPA ranks each protein in the database separately, we report  $k$  top-ranked proteins, where  $k$  is a user-specified parameter.
- ★ Subtraction: An implementation of the *subtraction* strategy with Equation (8) as the scoring function. In our implementation, the algorithm repeats  $k$  rounds to identify  $k$  proteins as output, where  $k$  is specified by the user. In each round, the masses matching the protein identified in the previous round are removed prior to the next round of searching.

Throughout the experiments, we use the same parameter setting in all PMF algorithms: Trypsin digestion with a maximum of one missed cleavage, monoisotopic peaks, single charge state, unrestricted protein mass. Other parameter specifications will be given at the right time.

In Losau,  $df$  is fixed to 0.9 and  $\theta$  is fixed to 2. In both Losak and Losau, only proteins in the database that match at least five peaks are considered as potential candidates in the local optimization process.

### 3.1 Simulated Data

We generate synthetic protein mixture data by following the procedure in [8].

Firstly, we randomly select a set of proteins from the sequence database (Swiss-Prot, Release 52) as the ground-truth proteins. To ensure that each ground-truth protein has a reasonable number of digested peptides, we restrict the molecular weight of each protein between 30,000 Da and 100,000 Da.

Secondly, we perform trypsin-based protein digestion in silico (1 missed cleavage sites) and simulate the peptide detectability by retaining only a portion of proteolytic peptides according to the *sequence coverage* parameter. Here we define the *sequence coverage* as the ratio between the number of detectable peptides and the number of all peptides within the mass acquisition range (800 Da  $\sim$  4500 Da in our simulation).

Finally, we simulate the experimental mass error and noise. We alter the mass of each peptide by adding a random number generated from a Gaussian distribution with zero mean and standard deviation  $\sigma$  (here  $\sigma$  is set as the mass tolerance threshold). We also add a set of noisy peaks that are randomly generated and uniformly distributed within the mass acquisition range. We determine the number of noisy peaks using the *noise level* parameter, which is defined as the ratio between the number of man-made noisy peaks and the number of total peaks (after adding these noisy peaks).

**Effect of Mass Accuracy.** To test the effect of mass accuracy on the performance of different algorithms, we generate simulation data with mass error ( $\sigma$ ) of 0.01 Da, 0.02 Da, 0.03 Da, and 0.04 Da, respectively. Each simulation

data contains 10 protein mixtures. In each mixture, the number of ground-truth proteins is 20, the sequence coverage is 0.3, and the noise level is 50%.

In all PMF methods, the mass tolerance threshold is set to be the known mass error. The average *precision/recall/F1-measure* at each mass error over 10 protein mixtures are used to compare different methods, as shown in Fig 2(a).

The increase of mass error will decrease the identification performance of various PMF algorithms. This general trend indicates that high mass accuracy is a necessary condition for the success of PMF in protein mixture identification.

Our algorithms are significantly better than traditional PMF algorithms. Moreover, our algorithms are very robust to the increase of mass error and still can produce good results when the mass error is relatively large. In contrast, Subtraction requires smaller mass error to achieve comparable performance.

The mass error can also affect the running time of our algorithms. Fig 2(a) shows that running time of our algorithms increases with the increase of mass error. This is because we use the mass error as the mass tolerance threshold. A larger mass tolerance threshold will introduce more candidate proteins so that the potential search space of our algorithms is enlarged. Therefore, our algorithms need more time to complete database searching. In contrast, SPA and Subtraction are less affected.

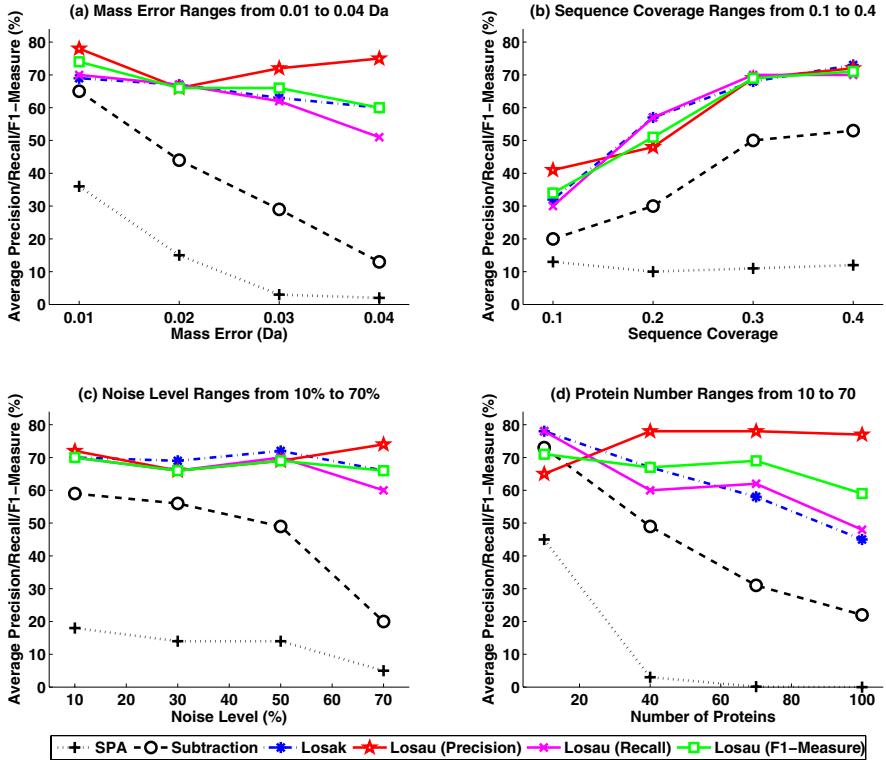
The running time of Losak increases linearly with respect to mass error. However, the running time of Losau increases dramatically when mass error is raised from 0.02 Da to 0.03 Da and thereafter begins to drop down. It indicates two facts: (1) Losak is more stable than Losau with respect to mass error and (2) Losau achieves comparable identification performance at the expense of much more computational time.

It should be noted that the complexity of Losau is not completely predictable, as shown by its remarkable drop in the running time when the mass tolerance is increased from 0.03 Da to 0.04 Da. The reason probably lies in the random initialization of starting points since our algorithms are more sensitive to initialization when the mass error is increased (see the supplement).

**Effect of Sequence Coverage.** Sequence coverage is the ratio between the number of detectable peptides and the number of all peptides within the mass acquisition range. Obtaining sufficient sequence coverage is of primary importance in the context of PMF. We generate simulation data with sequence coverage 0.1, 0.2, 0.3 and 0.4, respectively. At each specific sequence coverage value, we synthesize 10 protein mixtures using the following parameters: 20 proteins, mass error 0.02 Da and 50% noisy peaks.

Fig 2(b) shows that high sequence coverage is a necessary condition to accurately identify component proteins from mixtures using PMF. When the sequence coverage is low (e.g. 0.1), all algorithms perform poorly. When the coverage ratio increases, our algorithms outperform other methods significantly.

Fig 2(b) shows that the running time of different PMF algorithms increases when the sequence coverage increases. Since a high sequence coverage will introduce more peaks into the mixture data, PMF algorithms need more running time to perform protein-peak matching.

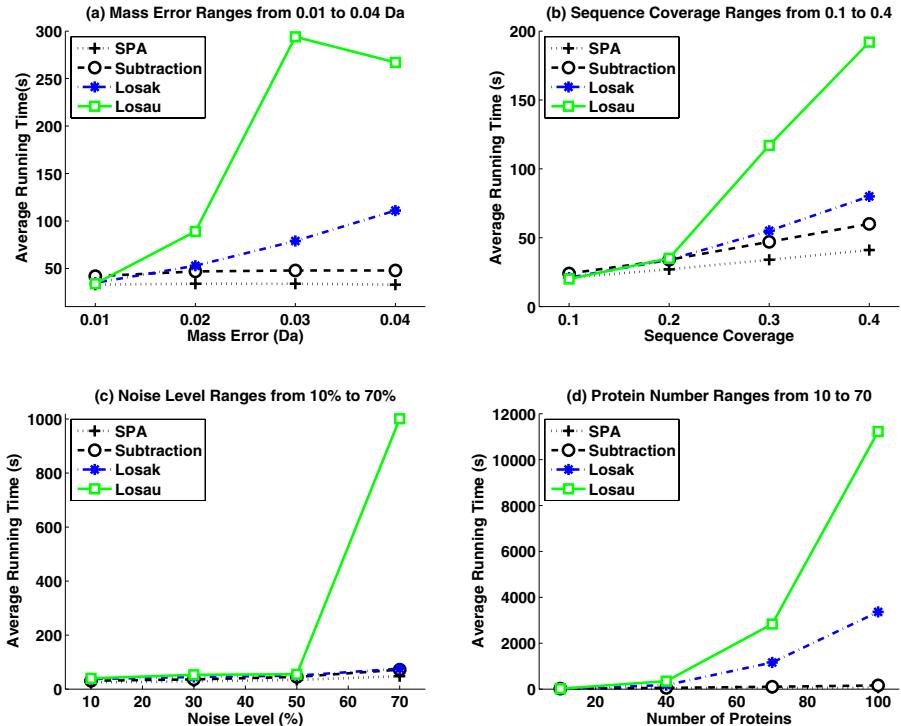


**Fig. 1.** Identification performance of different PMF algorithms on simulation data. In the experiments, we let SPA, Subtraction, and Losak to report  $k$  proteins, where  $k$  is number of ground-truth proteins. In this context, the *precision*, *recall* and *F1-measure* of each algorithm are identical. In Losau, these performance metrics will have different values since it adaptively determines the number of target proteins. Thus, we report the *precision*, *recall* and *F1-measure* of Losau, respectively.

**Effect of Noise Level.** MS data is noisy. Even in spectra generated from a single protein, there are usually more than 50% noisy peaks. Therefore, the ability to identify proteins from noisy mixtures is absolutely indispensable.

We set the noise level to 10%, 30%, 50% and 70%, respectively. At each noise level, we generate 10 mixtures using the following parameters: 20 proteins, mass error 0.02 Da and sequence coverage 0.3.

Fig. II(c) plots the identification results of different methods when the noise level ranges from 10% to 70%. The performance of Subtraction and SPA declines significantly when more noisy peaks are included, while our algorithms are very robust at different noise levels. As the real MS data can be more complicated than simulation data, it is not surprising that Subtraction will fail.



**Fig. 2.** Running time of different PMF algorithms on simulation data

Fig 2(c) describes the running time of different methods at different noise levels. All methods take more computational time when the number of noisy peaks increases.

**Effect of Protein Number in the Mixture.** The number of component proteins also has an effect on the performance of PMF algorithms. Generally, the ground-truth peaks of one protein may be considered as noise to other proteins. Therefore, the more proteins present in the mixture, the more difficult the identification problem becomes.

We set the number of component proteins to 10, 40, 70 and 100, respectively. For each fixed protein number, we generate 10 mixtures using the following parameters: Mass error 0.02 Da, sequence coverage 0.3 and no noisy peaks.

Fig 2(d) depicts that the identification performance of all PMF algorithms declines when the protein number increases. The performance decay of Subtraction is very fast because it is sensitive to “noise”. Here the “noise” corresponds to peaks digested by other proteins.

Fig 2(d) shows that the running time of PMF algorithms increases if there are more proteins in the mixture. Note that the computational time of Losak and Losau is almost quadratic to the protein number. This is because the increase

**Table 1.** Identification performance and running time of different algorithms on the real MS data. Here the number of reported proteins for SPA, Subtraction, and Losak is 49, i.e. the number of ground-truth proteins. The decay factor  $df$  is set to 0.95 in Losau.

Algorithms	Precision	Recall	F1-Measure	Running Time(s)
SPA	24%	24%	24%	7.9
Subtraction	43%	43%	43%	24.0
Losak	67%	67%	67%	21.2
Losau	61%	71%	66%	19.6

of protein number will increase number of peaks. In other words,  $k$  and  $l$  are increased simultaneously so that the running time of both algorithms increases significantly.

### 3.2 Real Data

Here we use a real MS data obtained from a mixture of 49 standard human proteins in the ABRF sPRG2006 study (<http://www.abrf.org/index.cfm/group.show/ProteomicsInformaticsResearchGroup.53.htm>). The data is generated using a linear ion trap-orbitrap (LTQ-Orbitrap) mass spectrometer.

We first use a program named Decon2LS (<http://ncrr.pnl.gov/software/>) to find peaks (and their mono-isotopic masses) in individual mass spectra of raw LC-MS data. Then, we use VIPER [11] to identify LC-MS peaks from MS peaks across elution time. We obtain 3366 de-convoluted peaks as the input for different PMF methods. Here we use the default parameter settings in both Decon2LS and VIPER.

We search against a sequence database that consists of all Swiss-Prot Human proteins and some bonus proteins and contaminant compounds (<http://www.abrf.org/ResearchGroups/ProteomicsInformaticsResearchGroup/Studies/sPRGBICFASTA.zip>). In database searching, we set the mass tolerance threshold to 1 ppm.

The results in Table 1 show that Losak and Losau achieve significant higher protein identification rate than previous PMF methods. Meanwhile, the performance of our algorithms can be further improved from the following perspectives:

1. The data generation process of current LC-MS/MS data is designed for peptide sequencing rather than PMF. It tries to separate peptides of different masses using High-Performance Liquid Chromatography (HPLC) so as to generate MS/MS data effectively. However, such setting creates additional difficulties of combining signals from the same peptide across adjacent scans and insufficient sequence coverage of single protein at each scan.
2. High accuracy of mass measurement is a double-edged sword: It reduced the search space of PMF algorithms while making the algorithms subtle to mono-isotopic mass determination. The assignment of true mono-isotopic masses is a complicated task in the analysis of complex mixtures. As we have observed

in the experiment, there are only 34 ground-truth proteins that match more than five peaks in candidate selection process. On the one hand, it indicates that the sequence coverage is insufficient during data generation; on the other hand, it means that some mono-isotopic masses of ground-truth proteins are not assigned correctly.

We also record the running time of different algorithms in Table 1. It shows that Losak and Losau are slower than SPA but are faster than Subtraction. The good running efficiency of our algorithms comes from two sources: The high mass accuracy of LTQ-Orbitrap data and the candidate generation procedure. The high mass accuracy enables us to use a very small mass tolerance threshold (e.g. 1 ppm) in the experiments. Consequently, the candidate generation procedure will only retain a small subset of proteins in the local search process. For instance, there is only 1215 candidate proteins that can match at least 5 peaks at the mass tolerance threshold of 1 ppm.

## 4 Conclusions

Through the use of two local search based algorithms, we show that there is a great potential to use PMF for protein mixture identification. We also discuss the bottlenecks that hamper the widespread use of PMF for protein mixture identification. Finally, it is promising to overcome these limitations and make PMF a standard tool in protein mixture identification.

## Acknowledgements

The comments and suggestions from anonymous reviewers greatly improved the paper. This work was supported with the general research fund 621707 from the Hong Kong Research Grant Council, a research proposal competition award RPC07/08EG.25 and a postdoctoral fellowship from the Hong Kong University of Science and Technology.

## References

1. Yates, J.R., Speicher, S., Griffin, P.R., Hunkapiller, T.: Peptide mass maps: a highly informative approach to protein identification. *Anal. Biochem.* 214(2), 297–408 (1993)
2. Perkins, D.N., Pappin, D.J.C., Creasy, D.M., Cottrell, J.S.: Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* 20(18), 3551–3567 (1999)
3. Clauser, K.R., Baker, P., Burlingame, A.L.: Role of accurate mass measurement ( $\pm 10$  ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal. Chem.* 71(14), 2871–2882 (1999)
4. Zhang, W., Chait, B.T.: Profound: an expert system for protein identification using mass spectrometric peptide mapping information. *Anal. Chem.* 72(11), 2482–2489 (2000)

5. Aebersold, R., Mann, M.: Mass spectrometry-based proteomics. *Nature* 422, 198–207 (2003)
6. Jensen, O.N., Podtelejnikov, A.V., Mann, M.: Identification of the components of simple protein mixtures by high-accuracy peptide mass mapping and database searching. *Anal. Chem.* 69(23), 4741–4750 (1997)
7. Park, Z.Y., Russell, D.H.: Identification of individual proteins in complex protein mixtures by high-resolution, high-mass-accuracy MALDI TOF-mass spectrometry analysis of in-solution thermal denaturation/enzymatic digestion. *Anal. Chem.* 73(11), 2558–2564 (2001)
8. Eriksson, J., Fenyö, D.: Protein identification in complex mixtures. *J. Proteome Res.* 4(2), 387–393 (2005)
9. Lu, B., Motoyama, A., Ruse, C., Venable, J., Yates, J.R.: Improving protein identification sensitivity by combining MS and MS/MS information for shotgun proteomics using LTQ-Orbitrap high mass accuracy data. *Anal. Chem.* 80(6), 2018–2025 (2008)
10. Samuelsson, J., Dalevi, D., Levander, F., Rögnvaldsson, T.: Modular, scriptable and automated analysis tools for high-throughput peptide mass fingerprinting. *Bioinformatics* 20(18), 3628–3635 (2004)
11. Monroe, M.E., Tolic, N., Jaitly, N., Shaw, J.L., Adkins, J.N., Smith, R.D.: VIPER: an advanced software package to support high-throughput LC-MS peptide identification. *Bioinformatics* 23(15), 2021–2023 (2007)

# Boosting Protein Threading Accuracy

Jian Peng and Jinbo Xu\*

Toyota Technological Institute at Chicago, Chicago, IL, USA, 60637  
{pengjian, j3xu}@tti-c.org

**Abstract.** Protein threading is one of the most successful protein structure prediction methods. Most protein threading methods use a scoring function linearly combining sequence and structure features to measure the quality of a sequence-template alignment so that a dynamic programming algorithm can be used to optimize the scoring function. However, a linear scoring function cannot fully exploit interdependency among features and thus, limits alignment accuracy.

This paper presents a nonlinear scoring function for protein threading, which not only can model interactions among different protein features, but also can be efficiently optimized using a dynamic programming algorithm. We achieve this by modeling the threading problem using a probabilistic graphical model Conditional Random Fields (CRF) and training the model using the gradient tree boosting algorithm. The resultant model is a nonlinear scoring function consisting of a collection of regression trees. Each regression tree models a type of nonlinear relationship among sequence and structure features. Experimental results indicate that this new threading model can effectively leverage weak biological signals and improve both alignment accuracy and fold recognition rate greatly.

**Keywords:** protein threading, conditional random fields, gradient tree boosting, regression tree, nonlinear scoring function.

## 1 Introduction

Protein structure prediction based on protein threading has made a significant progress due to both enlargement of the Protein Data Bank (PDB) and improvement of prediction protocols. Protein threading predicts the three-dimensional structure of a new protein (i.e., target) by aligning its primary sequence to a similar experimental structure (i.e., template) in the PDB. The PDB statistics<sup>1</sup> shows that in recent years, only a limited number of completely new protein folds appear although several thousand new structures are deposited to the PDB. According to [12], almost all the single-domain proteins with up to 200 residues can be aligned to a protein in the PDB with an average RMSD less than 5Å and an average coverage of 70%. These observations imply that in principle, a reasonably good template can be identified for most proteins with unknown structures if a perfect protein threading protocol is available.

The three-dimensional structure model of a target protein is built upon its alignment to the template. Therefore, the alignment accuracy of a protein threading method is

---

\* Corresponding author.

<sup>1</sup> [http://www.rcsb.org/pdb/static.do?p=general\\_information/pdb\\_statistics/index.html](http://www.rcsb.org/pdb/static.do?p=general_information/pdb_statistics/index.html)

critical to the model quality [3]. The best sequence-template alignment is generated by optimizing a scoring function, usually consisting of several sequence and structure items such as sequence profile, secondary structure, solvent accessibility, contact capacity and pairwise interactions. It is difficult to construct accurate alignment between two proteins with low sequence identity even if they are structurally similar. It has been observed that alignment quality drops rapidly when two proteins share less than 25% sequence identity [4]. There are usually errors in the alignment of two proteins with less than 40% sequence identity [5][6][7].

There are a number of protein threading programs in the literature based on various methods such as the profile-profile alignment [8][9][10][11][12][13][14][15], the structural profile alignment [16][17][18][19], the HMM models [20][21], the optimization approaches [22][23][24][25] and the multiple mapping method [26]. The pure sequence profile-based alignment methods have been very successful in identifying a good template for the target. However, it has been demonstrated by many programs that using structure information can improve both alignment accuracy and fold recognition rate. For example, several leading threading programs such as HHpred [21], SPARKS [14][19] and RAPTOR [22][23] make use of structure information such as secondary structure, solvent accessibility and residue depth. Zhang *et al* have shown that by using five structure features plus sequence profile, their threading program MUSTER [27] outperforms their profile-profile alignment program PPA [28]. Zhou *et al* show that a residue-depth based structure profile can improve both the sensitivity and specificity of the sequence profile alignments [14]. Silva shows that a simplified hydrophobicity matrix can help detect remote homologs [29]. Skolnick *et al* show that contact predictions are helpful for the targets without close homologs in the PDB [30].

Although various structure features have been used by many threading programs, they usually use a very simple method to combine these features. Most threading programs use a linear combination of these features as a scoring function [14][27][31][16][18][17]. A linear scoring function can be easily tuned and also can be efficiently optimized using a dynamic programming algorithm. However, a linear scoring function can not accurately account for the interdependency among features. It has been observed that some sequence and structure features (e.g., secondary structure and solvent accessibility) are highly correlated. To model interactions among features, the SVM-align method in [32] explicitly enumerates hundred-thousands of complex features, which leads to the same number of model parameters to be trained. A complex feature is a combination of some basic features, e.g., secondary structure, solvent accessibility and amino acid type. A model with a large number of parameters is not amenable to training since 1) it needs a large number of training examples to fit these parameters; 2) it needs careful tuning to avoid overfitting; and 3) the training process is also time-consuming. Using such a complicated model, it is also computationally intensive to find the best sequence-template alignment between a protein pair, which makes the model unsuitable for protein structure prediction at genome scale. In addition, not all the model features are equally important and some unimportant features may introduce noise into the model.

This paper presents a nonlinear scoring function for protein threading, which not only can model interactions among various sequence and structure features, but also can be optimized quickly using a dynamic programming algorithm. We fulfill this by

modeling the protein threading problem using a probabilistic graphical model Conditional Random Fields (CRFs) and training the model using the gradient tree boosting algorithm proposed in [33]. The resultant threading scoring function consists of only dozens of regression trees, which are automatically constructed during model training process to capture the nonlinear relationship among sequence and structure features. Experimental results indicate that by modeling feature interactions using regression trees, we can effectively leverage weak biological signals and greatly improve alignment accuracy and fold recognition rate.

## 2 Methods

### 2.1 Conditional Random Fields Model for Protein Threading

Conditional random fields are probabilistic graphical models that have been extensively used in modeling sequential data [34][35]. Recently, CRFs have also been used to model various computational biology problems such as protein secondary structure prediction [36][37], protein conformation sampling [38] and protein sequence alignment [39]. Here we describe how to model the protein threading problem using conditional random fields. In this paper, for a given pair of target and template, their sequence and structure features are called observations and their alignment is viewed as a sequence of hidden states or labels.

Let  $s$  denote the target protein and its associated features, e.g., PSI-BLAST sequence profile, PSIPRED-predicted secondary structure [40] and predicted solvent accessibility. Let  $t$  denote the template and its associated information, e.g., position-specific scoring matrix, solvent accessibility and secondary structure. Let  $X = \{M, I_s, I_t\}$  be a set of three possible alignment states. Meanwhile,  $M$  indicates that two positions are matched and the two  $I$ s indicate insertion/deletion states.  $I_s$  and  $I_t$  indicate insertion at sequence and template, respectively. We also tried to differentiate gaps at the two ends from gaps in the middle region by using four more states, but the resultant 7-state model is only slightly better than the three-state model. Let  $a = \{a_1, a_2, \dots, a_L\}$  ( $a_i \in X$ ) denote an alignment between  $s$  and  $t$  where  $a_i \in X$  represents the state (or label) at position  $i$ . Our CRF model for threading defines the conditional probability of  $a$  given  $s$  and  $t$  as follows.

$$P_\theta(a|s, t) = \exp\left(\sum_{i=1}^L F(a, s, t, i)\right)/Z(s, t) \quad (1)$$

where  $\theta=(\lambda_1, \lambda_2, \dots, \lambda_p)$  is the model parameter and  $Z(s, t)=\sum_a \exp(\sum_{i=1}^{L_a} F(a, s, t, i))$  is a normalization factor summing over all the possible alignments between  $s$  and  $t$ <sup>2</sup>.  $F(a, s, t, i)$  is the sum of the CRF features at alignment position  $i$ :

$$F(a, s, t, i) = \sum_k \lambda_k e_k(a_{i-1}, a_i, s, t) + \sum_l \lambda_l v_l(a_i, s, t) \quad (2)$$

---

<sup>2</sup> We use  $L_a$  to denote the length of an alignment  $a$  in the case we need to differentiate the length of two alignments.

Where  $e_k(a_{i-1}, a_i, s, t)$  and  $v_l(a_i, s, t)$  are called edge and label feature functions, respectively. The edge features model the dependency of the state transition (from  $i-1$  to  $i$ ) on the target and template information around positions  $i-1$  and  $i$ . The label features model the relationship between the state at position  $i$  and the target and template information around this position. Once the model parameters are determined, we can find the best sequence-template alignment by maximizing  $P_\theta(a|s, t)$ , which can be done using a dynamic programming algorithm since  $P_\theta(a|s, t)$  only models state transition between two adjacent positions.

Our CRF-based threading model is more expressive than the generative threading models [2][41][42]. First, both the label and edge feature functions can be a nonlinear function, which can be used to capture the complex relationship among alignment state, sequence information and structural information. Second, the alignment state at position  $i$  may depend on sequence and structure information at positions around  $i$  instead of only at position  $i$ . So is the state transition between two adjacent alignment positions. Therefore, the CRF model can accommodate complex feature sets that may be difficult to incorporate within a generative HMM model. The underlying reason is that CRFs only optimize the conditional probability  $P_\theta(a|s, t)$  instead of joint probability  $P_\theta(a, s, t)$ , avoiding calculating the generative probability of the observations.

In Equation 2 the function potential  $F(a, s, t, i)$  at position  $i$  is a linear combination of some features. This kind of linear parameterization implicitly assumes that all the features are linearly independent. This contradicts with the fact that the sequence and structure features are highly correlated. For example, the target secondary structure is usually predicted from PSI-BLAST sequence profile using PSIPRED [43]. The secondary structure is also correlated with primary sequence and solvent accessibility. To model interactions among features using Equation 2 one way is to explicitly define more complex features, each of which is a combination of some basic features, just like what [39] and [32] have done. However, explicitly defining complex features leads to a combinatorial explosion in the number of features, and hence, in the model complexity. The parameter training procedure for such a model needs careful tuning to avoid overfitting. In addition, explicit enumeration of features may also introduce a large number of unnecessary features, which greatly increases the computational time of finding the best sequence-template alignment.

## 2.2 Model Training Using Gradient Tree Boosting

Instead of explicitly enumerating thousands of complex features, we implicitly construct only a small number of important features using regression trees. In this method, the left hand side of Equation 2 is represented as a combination of regression trees instead of a linear combination of features. Each regression tree models the complex interactions among the basic features and each path from the tree root to a leaf corresponds to a single complex feature. We can build these regression trees automatically during the training process using the gradient boosting algorithm proposed by Dietterich *et al* [33]. Only those important features emerge as a path (from tree root to a leaf) in the trees. The resulting scoring function has the form of a linear combination of regression trees. One advantage of the gradient tree boosting approach is that it is unnecessary to explicitly enumerate all the complex features. The important features

can be automatically learned during the training process. By contrast, explicit enumeration may not generate features as good as those learned by regression trees. Another advantage is that we may avoid overfitting because of the ensemble effect of combining multiple regression trees and much fewer complex features used. Finally, once the regression-tree-based threading model is trained, we can find the best alignment very quickly using dynamic programming since there are only dozens of regression trees in the scoring function.

To use the gradient tree boosting algorithm [33], we use a slightly different representation of Equations 1 and 2. Let  $F^{a_i}(a_{i-1}, s, t)$  be a function that calculate the log-likelihood of the alignment state at position  $i$  given the alignment state at position  $i-1$  and the target and the template information around positions  $i-1$  and  $i$ . Then the CRF threading model has the form given by

$$P(a|s, t) = \frac{\exp(\sum_i F^{a_i}(a_{i-1}, s, t))}{Z(s, t)} \quad (3)$$

In this new representation, there are no concepts of edge and label features. Instead,  $\ln P(a|s, t)$  is a linear combination of some functions of form  $F^{a_i}(a_{i-1}, s, t)$  where  $F^{a_i}(a_{i-1}, s, t)$  is a linear combination of regression trees. To train such a model, we need to calculate the functional gradient of the conditional probability with respect to  $F^{a_i}(a_{i-1}, s, t)$ . Using a similar technique described in [33], we have the following result.

**Lemma 1.** *Let  $u$  and  $v$  denote the alignment states at positions  $i-1$  and  $i$ , respectively. The functional gradient of  $\ln P(a|s, t)$  with respect to  $F^{a_i}(a_{i-1}, s, t)$  is given by*

$$\frac{\partial \ln P(a|s, t)}{\partial F^v(u, s, t)} = I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t) \quad (4)$$

Where  $I(a_{i-1} = u, a_i = v)$  is a 0-1 function. Its value equals to 1 if and only if in the training alignment the state transition from  $i-1$  to  $i$  is  $u \rightarrow v$ .  $P(a_{i-1} = u, a_i = v|s, t)$  is the predicted probability of the state transition  $u \rightarrow v$  under current threading model.

The functional gradient in Equation 4 is easy to interpret. Given a training alignment, if the transition  $u \rightarrow v$  is observed at position  $i$ , then ideally the predicted probability  $P(a_{i-1} = u, a_i = v|s, t)$  should be 1 in order to make  $\frac{\partial \ln P(a|s, t)}{\partial F^v(u, s, t)}$  be 0 and thus, to maximize  $P(a|s, t)$ . Similarly, if the transition is not observed, then the predicted probability should be 0 to maximize  $P(a|s, t)$ . Given an initial  $F^v(u, \dots)$ , to maximize  $P(a|s, t)$ , we need to move  $F^v(u, \dots)$  along the gradient direction defined by all the  $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$ . Since  $F^v(u, \dots)$  is a function taking as input the sequence and structure features around each alignment position, the gradient direction is also a function with the same input variables. We can use a function  $T^v(u, \dots)$  to fit  $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$  with the corresponding input values being the sequence and template features around positions  $i-1$  and  $i$ . Then  $F^v(u, \dots)$  is updated by  $F^v(u, \dots) + wT^v(u, \dots)$  where  $w$  is the step size and  $T^v(u, \dots)$  is the gradient direction. The gradient tree boosting algorithm simply involves fitting regression trees to the difference between the observed and the predicted probabilities

of each possible state transition. There are many possible functions that can fit a given set of data. Regression trees are chosen because they are easy to interpret and can be quickly trained from a large number of examples. In addition, we can also control the tree depth or the number of leaves to avoid overfitting.

Given a threading model and a training alignment, we can calculate  $P(a_{i-1} = u, a_i = v | s, t)$  using a forward-backward method. Let  $\alpha(v, i)$  and  $\beta(v, i)$  denote the probabilities of reaching state  $v$  at position  $i$ , starting from the N-terminal and C-terminal of the alignment, respectively. Both  $\alpha(v, i)$  and  $\beta(v, i)$  can be recursively calculated as follows.

$$\alpha(v, 1) = \exp F^v(\phi, s, t) \quad (5)$$

$$\alpha(v, i) = \sum_u [\exp F^v(u, s, t)] \alpha(u, i - 1) \quad (6)$$

Where  $\phi$  represents a dummy state.

$$\beta(u, L) = 1 \quad (7)$$

$$\beta(u, i) = \sum_v [\exp F^v(u, s, t)] \beta(v, i + 1) \quad (8)$$

Then  $P(a_{i-1} = u, a_i = v | s, t)$  can be calculated by

$$P(a_{i-1} = u, a_i = v | s, t) = \frac{\alpha(u, i - 1) [\exp F^v(u, s, t)] \beta(v, i)}{Z(s, t)} \quad (9)$$

and the normalizer  $Z(s, t)$  can be calculated by

$$Z(s, t) = \sum_u \alpha(u, 0) \beta(u, 0) \quad (10)$$

Given a set of training alignments, the gradient tree boosting algorithm to train the threading model is shown in Algorithm II. The main component of the algorithm is to generate a set of examples to train a regression tree  $T^v(u, ., .)$  for any feasible state transition  $u \rightarrow v$ . At any two adjacent positions of a training alignment, we generate an example by calculating  $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v | s, t)$  as the response value and extracting sequence and structure features around these two positions as the input values. Then we fit a regression tree to these examples and update  $F^v(u, ., .)$  accordingly.

There are some tricky issues in building the regression trees due to the extremely unbalanced number of positive and negative examples. A training example is positive if its response value is positive, otherwise negative. Given a training alignment with 200 residues in each protein and 150 aligned positions, the ratio between the number of positive examples and that of negative ones is approximately  $\frac{200+200-150}{200 \times 200 \times 3}$ . This will result in serious bias in regression tree training. We employed two strategies to resolve this issue. One is to add more weight to the positive examples and the other is that we randomly sample a small subset of negative examples for training [44].

---

**Algorithm 1.** Gradient Tree Boosting Training Algorithm

---

```

function TreeBoostThreader(Data) //  $Data = \{(a^j, s^j, t^j)\}$  where  $j$  indicates the  $j^{th}$  alignment example.  $L^j$  is the length of the  $j^{th}$  alignment.
for all state transition  $u \rightarrow v$  do
    initialize  $F_0^v(u, ., .) = 0$  //  $u \rightarrow v$  is the feasible state transition at two adjacent positions
    //the second variable of  $F_0^v(u, ., .)$  is the features extracted from the target protein
    //the third variable of  $F_0^v(u, ., .)$  is the structure features from the template protein
end for
//training at most  $M$  iterations
for  $m$  from 1 to  $M$  do
    for all state transition  $u \rightarrow v$  do
         $S(u,v):=\text{GenerateExamples}(u, v, \text{Data})$ 
         $T_m(u, v):=\text{FitRegressionTree}(S(u,v))$ 
         $F_m^v(u, ., .) := F_{m-1}^v(u, ., .) + T_m(u, v)$ 
    end for
end for
return  $F_M^v(u, s, t)$  as  $F^v(u, ., .)$  for all  $u \rightarrow v$ 
end function

function GenerateExamples( $u, v, \text{Data}$ )
for all training alignments do
    for all  $i$  from 1 to  $L^j$  do
        Calculate  $\alpha$  and  $\beta$  according to Equations 6 and 8
        for all state transition  $u \rightarrow v$  do
            Calculate  $P(u, v|s, t)$  using Equation 9
             $\delta(i, u, v, s, t) = I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$ 
            // $\delta$  is the response value to be fitted by a regression tree
            // $s(i)$  and  $t(i)$  are the sequence and structure features around position  $i$ 
            insert an example data entry  $(s(i), t(i), \delta)$  into  $S(u, v)$ 
        end for
    end for
end for
return  $S(u, v)$ 
end function

```

---

Unlike the traditional CRF using  $L2$  norm to regularize the model complexity and avoid overfitting [35], the complexity of our model is regularized by the tree depth. In building each regression tree, we use an internal 5-fold cross-validation procedure to determine the best tree depth. In our case the average tree depth is 4. Using such a regularization, we can avoid overfitting in training the model.

### 2.3 Sequence-Template Alignment Algorithm

Once a regression-tree-based threading model has been trained, we can find the best alignment  $a$  by maximizing  $P(a|s, t)$  using a dynamic programming algorithm. This step is similar to all the HMM-based sequence alignment procedure. The best sequence-template alignment can be computed by the well-known Viterbi algorithm [45], which has the advantage that it does not need to compute the normalizer  $Z(s, t)$ .

## 2.4 Sequence and Structure Features

We use both evolutionary information and structure information to build regression trees for our threading model. We generate position specific score matrix (PSSM) for a template and position specific frequency matrix (PSFM) for a target using PSI-BLAST with five iterations and E-value 0.001.  $PSSM(i, a)$  is the mutation potential for amino acid  $a$  at template position  $i$  and  $PSFM(j, b)$  is the occurring frequency of amino acid  $b$  at target position  $j$ . The secondary structure and solvent accessibility of a template is calculated by the DSSP program [46]. For a target protein, we use PSIPRED [43] and SSpro [47] to predict its secondary structure and solvent accessibility, respectively.

*Features for match state.* We use the following features to build regression trees for a state transition to a match state. Suppose that template position  $i$  is aligned to target position  $j$ .

1. Sequence similarity. The sequence similarity score between two aligned positions is calculated by  $\sum_a PSSM(i, a) \times PSFM(j, a)$ .
2. Contact capacity score. The contact capacity potential describes the hydrophobic contribution of free energy, measured by the capability of a residue make a certain number of contacts with other residues in a protein. The two residues are in physical contact if the spatial distance between their  $C_\beta$  atoms is smaller than 8Å. Let  $CC(a, k)$  denote the contact potential of amino acid  $a$  having  $k$  contacts (see Section 3 in [48]). The contact capacity score is calculated by  $\sum_a CC(a, c) \times PSFM(j, a)$  where  $c$  is the number of contacts at template position  $i$ .
3. Environmental fitness score. This score measures how well to align one target residue to a template local environment, which is defined by a combination of three secondary structure types and three solvent accessibility states. Let  $F(env, a)$  denote the environment fitness potential for amino acid  $a$  being in a local environment  $env$  (see Section 3 in [48]). The environment fitness score is given by  $\sum_a F(env_i, a) \times PSFM(j, a)$
4. Secondary structure match score. Supposing the secondary structure type at template position  $i$  is  $ss$ , then the predicted likelihood of  $ss$  at target position  $j$  is used as the secondary structure match score.
5. Solvent accessibility match score. This is a binary feature used to indicate if the template position and the target position are in the same solvent accessibility state.

*Features for gap state.* The simplest scoring model for gap penalty is an affine function  $o + e \times g$  where  $o$  is the gap open penalty,  $e$  gap extension penalty and  $g$  the number of gapped positions. To improve alignment accuracy, some threading programs such as SALIGN [49] also use a context-specific gap penalty function while others such as HHpred [21], SP5 [50] and the new PROSPECT [51] use a position-specific gap penalty model. In our threading model, we use a more sophisticated context-specific gap penalty function. Our future plan is to also incorporate position-specific gap penalty into our threading model. The regression trees for a state transition to an insertion state at the template depend on the following features: secondary structure type, solvent accessibility, amino acid identity and hydropathy count [39]. Similarly, the regression trees for a state transition to an insertion state at the target depend on the following features:

predicted secondary structure likelihood scores, predicted solvent accessibility, amino acid identity and hydropathy count.

## 3 Results

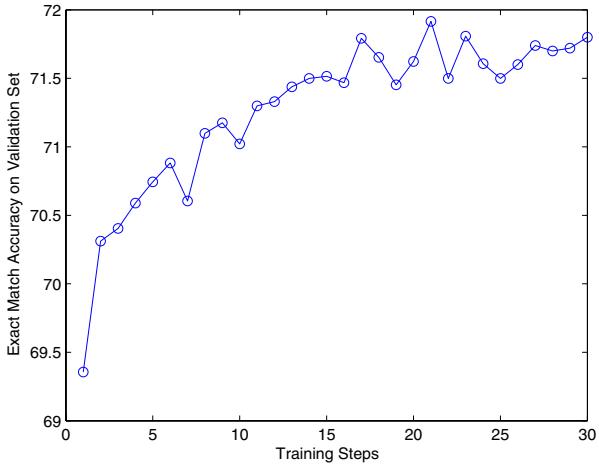
### 3.1 Model Training

Similar to CONTRAlign [39], our boosting-based threading model does not need a large data set for training. The alignment accuracy on the validation set does not increase with respect to the training set size as long as it is at least 30. We arbitrarily choose 30 protein pairs from the PDB as our training set and 40 pairs as the validation set. The average size of a protein contains 200 residues. In the training set, 20 pairs are in the same fold level but different superfamily level according to the SCOP classification [52]. The other 10 pairs are in the same superfamily but different family level. Any two proteins in the training and validation set have sequence identity less than 30%. Reference alignments are built by the structure alignment program TM-align [53]. We also guarantee that the proteins used for model training have no high sequence identity (30%) with the proteins in the Prosup and SALIGN benchmarks (see Section 3.2).

As shown in Figure 1, the training process runs very fast. It takes approximately two minutes for each training iteration and achieves very good alignment accuracy after only six or seven iterations. The alignment accuracy reaches the best value after 21 iterations. More training iterations do not improve alignment accuracy but result in more regression trees. The more regression trees are used in the threading model, the more running time will be spent aligning a protein pair. As a result, we choose the model trained after 21 iterations as our final threading model. For each state transition, the model has twenty-one regression trees with an average depth four.

### 3.2 Alignment Accuracy

To compare our method with other state-of-art threading programs, we evaluate them on two popular benchmarks: Prosup[54] and SALIGN [49]. The Prosup benchmark [54] has 127 protein pairs with structural alignment generated by Prosup. The SALIGN benchmark [49] contains 200 protein pairs. On average, two proteins in a pair share



**Fig. 1.** The alignment accuracy of the models on the validation data set during the whole training process

**Table 1.** Alignment accuracy (%) of our method BoostThreader and other alignment methods on the Prosup and SALIGN benchmarks

Prosup			SALIGN		
Methods	Exact	4-offset	Methods	Exact	4-offset
CONTRAlign	52.79	69.42	CONTRAlign	44.38	57.37
PSIBLAST	35.60		PSIBLAST	26.10	
SPARKS	57.20		SPARKS	53.10	
SSALGN	58.30		SALIGN	56.40	
RAPTOR	61.30	79.32	RAPTOR	40.20	59.80
SP3	65.30	82.20	SP3	56.30	56.60
SP5	68.70		SP5	59.70	
BoostThreader	<b>74.05</b>	<b>88.90</b>	BoostThreader	<b>63.60</b>	<b>79.01</b>

20% sequence identity and 65% of structurally equivalent  $C_\alpha$  atoms superposed with RMSD 3.5 Å.

We used TM-align [53] to generate structural alignments for the SALIGN benchmark. The SALIGN benchmark is more difficult than the Prosup benchmark and includes many pairs of proteins with very different sizes. To evaluate the alignment quality, we use the exact match accuracy which is computed as the percentage of one-to-one match positions in the benchmark pairs. We also evaluate the 4-offset match accuracy, which is defined as the percentage of the matches within 4 positions shift from one-to-one match.

Table I compares the performance of various alignment methods on the Prosup benchmark. Our method, denoted as BoostThreader, shows a significant improvement over the others. The absolute improvement over SP3/SP5, a leading threading program, is more than 5%. The major difference between our method and SP3/SP5 is that SP3/SP5 linearly combines various sequence and structure features as its scoring function while our method uses a nonlinear scoring function. CONTRAlign [39] is run locally with the default hydropathy model. CONTRAlign mainly aims at sequence alignment, so it is not surprising that its performance is not as competitive as some leading threading methods. The results of other methods are taken from [55][50][41].

Also as shown in the right three columns of Table I, our method also has the best alignment accuracy on the SALIGN benchmark. This benchmark contains many pairs of proteins with very different sizes, which is the major reason why RAPTOR performs badly on this benchmark.

### 3.3 Fold Recognition Rate

We also evaluate the fold recognition rate of our new method BoostThreader on the Lindahl’s benchmark [56], which contains 976 proteins. Any two proteins in this set share less than 40% sequence identity. All-against-all threading of these proteins can generate  $976 \times 975$  pairs. After generating the alignments of all the pairs using our regression-tree-based threading model, we rank all the templates for each sequence using a method similar to [48] and then evaluate the fold recognition rate of our method.

**Table 2.** Fold recognition rate (%) of various threading programs. The PSI-BLAST, SPARKS, SP3, SP5 and HHpred results are taken from [50]. The FOLDpro, HMMER, FUGUE, SAM-98 results are from [57]. The RAPTOR and PROSPECT-II results are from [48].

Methods	Family		Superfamily		Fold	
	Top1	Top5	Top1	Top5	Top1	Top5
PSIBLAST	71.2	72.3	27.4	27.9	4.0	4.7
HMMER	67.7	73.5	20.7	31.3	4.4	14.6
SAM-98	70.1	75.4	28.3	38.9	3.4	18.7
THREADER	49.2	58.9	10.8	24.7	14.6	37.7
FUGUE	82.2	85.8	41.9	53.2	12.5	26.8
PROSPECT-II	84.1	88.2	52.6	64.8	27.7	50.3
SPARKS	81.6	88.1	52.5	69.1	24.3	47.7
SP3	81.6	86.8	55.3	67.7	28.7	47.4
FOLDpro	85.0	89.9	55.5	70.0	26.5	48.3
SP5	81.6	87.0	59.9	70.2	37.4	58.6
HHpred	82.9	87.1	58.8	70.0	25.2	39.4
RAPTOR	<b>86.6</b>	89.3	56.3	69.0	38.2	<b>58.7</b>
BoostThreader	86.5	<b>90.5</b>	<b>66.1</b>	<b>76.4</b>	<b>42.6</b>	57.4

When evaluating the performance in the superfamily level, all the templates similar in the family level are ignored. Similarly, when evaluating the performance in the fold level, all the templates similar in the superfamily or family level are ignored. “Top 1” means that the only the first-ranked templates are evaluated while “Top 5” indicates that the best templates out of the top 5 are evaluated. As shown in Table 2, our method performs well in all three similarity levels. The fold recognition rate of our new method is much better than SP3/SP5, HHpred and RAPTOR, especially in the superfamily and fold levels. These three programs performed very well in recent CASP (Critical Assessment of Structure Prediction) events.

## 4 Conclusion

This paper has presented a new protein threading model based on conditional random fields and gradient tree boosting. By using regression trees to represent the scoring function, this CRF-based threading model can accommodate as many sequence and structure features as possible and accurately model their interactions. Our model can also capture complex feature interactions without introducing a large number of redundant features. Although complex, such a scoring function still can be efficiently optimized by a dynamic programming algorithm. It takes less than half a second to thread a typical protein pair. Experimental results also demonstrate that by carefully account for the interdependency among features, we can greatly improve alignment accuracy over other leading threading programs. The improved alignment accuracy also leads to the improvement of fold recognition rate.

Currently, our threading model only considers state transition between two adjacent positions. A straight-forward generalization is to model state dependency among three

adjacent positions. We can also model pairwise interaction between two nonadjacent positions. The challenge of modeling non-local interactions is that it is computationally hard to train and test such a model. Some approximation algorithms may be resorted. It is also possible to use tree decomposition [25] or linear programming [22,23] to train such a model efficiently.

## Acknowledgements

This work is supported by TTI-C internal research funding and NIH research grant. The authors are grateful to Liefeng Bo and Kristian Kersting for their help with the gradient tree boosting technique. We also thank Chuong Do for his help with the CONTRAlign software.

## References

1. Kihara, D., Skolnick, J.: The PDB is a covering set of small protein structures. *Journal of Molecular Biology* 334(4), 793–802 (2003)
2. Zhang, Y., Skolnick, J.: The protein structure prediction problem could be solved using the current PDB library. *Proceedings of National Academy Sciences, USA* 102(4), 1029–1034 (2005)
3. Jones, D.T.: Progress in protein structure prediction. *Current Opinion in Structural Biology* 7(3), 377–387 (1997)
4. Rost, B.: Twilight zone of protein sequence alignments. *Protein Engineering* 12, 85–94 (1999)
5. John, B., Sali, A.: Comparative protein structure modeling by iterative alignment model building and model assessment. *Nucleic Acids Research* 31(14), 3982–3992 (2003)
6. Chivian, Dylan, Baker, David: Homology modeling using parametric alignment ensemble generation with consensus and energy-based model selection. *Nucleic Acids Research* 34(17), e112 (2006)
7. Marko, A.C., Stafford, K., Wymore, T.: Stochastic Pairwise Alignments and Scoring Methods for Comparative Protein Structure Modeling. *Journal of Chemical Information and Modeling* (March 2007)
8. Jaroszewski, L., Rychlewski, L., Li, Z., Li, W., Godzik, A.: FFAS03: a server for profile-profile sequence alignments. *Nucleic Acids Research* 33(Web Server issue) (July 2005)
9. Rychlewski, L., Jaroszewski, L., Li, W., Godzik, A.: Comparison of sequence profiles. Strategies for structural predictions using sequence information. *Protein Science* 9(2), 232–241 (2000)
10. Yona, G., Levitt, M.: Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *Journal of Molecular Biology* (315), 1257–1275 (2002)
11. Pei, J., Sadreyev, R., Grishin, N.V.: PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics* 19(3), 427–428 (2003)
12. Marti-Renom, M.A., Madhusudhan, M.S., Sali, A.: Alignment of protein sequences by their profiles. *Protein Science* 13(4), 1071–1087 (2004)
13. Ginalski, K., Pas, J., Wyrwicz, L.S., von Grotthuss, M., Bujnicki, J.M., Rychlewski, L.: OR-Feus: Detection of distant homology using sequence profiles and predicted secondary structure. *Nucleic Acids Research* 31(13), 3804–3807 (2003)

14. Zhou, H., Zhou, Y.: Single-body residue-level knowledge-based energy score combined with sequence-profile and secondary structure information for fold recognition. *Proteins: Structure, Function, and Bioinformatics* 55(4), 1005–1013 (2004)
15. Han, S., Lee, B.-C., Yu, S.T., Jeong, C.-S., Lee, S., Kim, D.: Fold recognition by combining profile-profile alignment and support vector machine. *Bioinformatics* 21(11), 2667–2673 (2005)
16. Shi, J., Blundell, T.L., Mizuguchi, K.: FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *Journal of Molecular Biology* 310(1), 243–257 (2001)
17. Jones, D.T.: GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *Journal of Molecular Biology* 287(4), 797–815 (1999)
18. Kelley, L.A., MacCallum, R.M., Sternberg, M.J.: Enhanced genome annotation using structural profiles in the program 3D-PSSM. *Journal of Molecular Biology* 299(2), 499–520 (2000)
19. Zhou, H., Zhou, Y.: Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins: Structure, Function, and Bioinformatics* 58(2), 321–328 (2005)
20. Karplus, K., Barrett, C., Hughey, R.: Hidden Markov Models for Detecting Remote Protein Homologies. *Bioinformatics* 14(10), 846–856 (1998)
21. Johannes, S.: Protein homology detection by HMM-HMM comparison. *Bioinformatics* 21(7), 951–960 (2005)
22. Xu, J., Li, M., Lin, G., Kim, D., Xu, Y.: Protein threading by linear programming. In: The Pacific Symposium on Biocomputing, pp. 264–275 (2003)
23. Xu, J., Li, M., Kim, D., Xu, Y.: RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology* 1(1), 95–117 (2003)
24. Xu, J., Li, M.: Assessment of RAPTOR's linear programming approach in CAFASP3. *Proteins: Structure, Function and Genetics* (2003)
25. Xu, J., Jiao, F., Berger, B.: A tree-decomposition approach to protein structure prediction. In: Proceedings of IEEE Computational Systems Bioinformatics Conference, pp. 247–256 (2005)
26. Rai, B.K., Fiser, A.: Multiple mapping method: a novel approach to the sequence-to-structure alignment problem in comparative protein structure modeling. *Proteins: Structure, Function, and Bioinformatics* 63(3), 644–661 (2006)
27. Wu, S., Zhang, Y.: MUSTER: Improving protein sequence profile-profile alignments by using multiple sources of structure information. *Proteins: Structure, Function, and Bioinformatics* 9999(9999), NA+ (2008)
28. Wu, S., Skolnick, J., Zhang, Y.: Ab initio modeling of small proteins by iterative TASSER simulations. *BMC Biology* 5, 17+ (2007)
29. Silva, P.J.: Assessing the reliability of sequence similarities detected through hydrophobic cluster analysis. *Proteins: Structure, Function, and Bioinformatics* 70(4), 1588–1594 (2008)
30. Skolnick, J., Kihara, D.: Defrosting the frozen approximation: PROSPECTOR - a new approach to threading. *Proteins: Structure, Function, and Genetics* 42(3), 319–331 (2001)
31. Kim, D., Xu, D., Guo, J., Ellrott, K., Xu, Y.: PROSPECT II: Protein structure prediction method for genome-scale applications. *Protein Engineering* (2002)
32. Yu, C.N., Joachims, T., Elber, R., Pillardy, J.: Support vector training of protein alignment models. *Journal of Computational Biology* 15(7), 867–880 (2008)

33. Dietterich, T.G., Ashenfelter, A., Bulatov, Y.: Training Conditional Random Fields via Gradient Tree Boosting. In: Proceedings of the 21st International Conference on Machine Learning (ICML), pp. 217–224 (2004)
34. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML: Proc. 18th International Conf. on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)
35. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of Human Language Technology NAACL 2003, pp. 134–141 (2003)
36. Shen, R.: Protein secondary structure prediction using conditional random fields and profiles. Master Thesis, Department of Computer Science, Oregon State University (2006)
37. Lafferty, J., Zhu, X., Liu, Y.: Kernel Conditional Random Fields: Representation and Clique Selection. In: ICML 2004: Proceedings of the twenty-first international conference on Machine learning. ACM Press, New York (2004)
38. Zhao, F., Li, S., Sterner, B.W., Xu, J.: Discriminative learning for protein conformation sampling. *Proteins: Structure, Function, and Bioinformatics* 73(1), 228–240 (2008)
39. Do, C., Gross, S., Batzoglou, S.: CONTRALign: Discriminative Training for Protein Sequence Alignment (2006)
40. McGuffin, L.J., Bryson, K., Jones, D.T.: The PSIPRED protein structure prediction server. *Bioinformatics* 16(4), 404–405 (2000)
41. Qiu, J., Elber, R.: SSALN: An alignment algorithm using structure-dependent substitution matrices and gap penalties learned from structurally aligned protein pairs. *Proteins: Structure, Function, and Bioinformatics* 62(4), 881–891 (2006)
42. Karplus, K., Karchin, R., Shackelford, G., Hughey, R.: Calibrating E-values for Hidden Markov Models using Reverse-Sequence Null Models. *Bioinformatics* 21(22), 4107–4115 (2005)
43. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology* 292(2), 195–202 (1999)
44. Gutmann, B., Kersting, K.: Stratified Gradient Boosting for Fast Training of Conditional Random Fields. In: Proceedings of the 6th International Workshop on Multi-Relational Data Mining, pp. 56–68
45. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 267–296 (1990)
46. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22(12), 2577–2637 (1983)
47. Pollastri, G., Baldi, P., Fariselli, P., Casadio, R.: Prediction of coordination number and relative solvent accessibility in proteins. *Proteins: Structure, Function, and Genetics* 47(2), 142–153 (2002)
48. Xu, J.: Fold Recognition by Predicted Alignment Accuracy. *IEEE/ACM Transaction of Computational Biology and Bioinformatics* 2(2), 157–165 (2005)
49. Marti-Renom, M.A., Madhusudhan, M.S., Sali, A.: Alignment of protein sequences by their profiles. *Protein Science* 13(4), 1071–1087 (2004)
50. Zhang, W., Liu, S., Zhou, Y.: SP5: Improving protein fold recognition by using torsion angle profiles and profile-based gap penalty model. *PLoS ONE* 3(6) (2008)
51. Ellrott, K., Guo, J.T., Olman, V., Xu, Y.: Improvement in protein sequence-structure alignment using insertion/deletion frequency arrays. In: Computational systems bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference, vol. 6, pp. 335–342 (2007)
52. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* 247(4), 536–540 (1995)

53. Zhang, Y., Skolnick, J.: TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research* 33(7), 2302–2309 (2005)
54. Lackner, P., Koppensteiner, W.A., Sippl, M.J., Domingues, F.S.: ProSup: a refined tool for protein structure alignment. *Protein Engineering* 13(11), 745–752 (2000)
55. Liu, S., Zhang, C., Liang, S., Zhou, Y.: Fold Recognition by Concurrent Use of Solvent Accessibility and Residue Depth. *Proteins: Structure, Function, and Bioinformatics* 68(3), 636–645 (2007)
56. Lindahl, E., Elofsson, A.: Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology* 295(3), 613–625 (2000)
57. Cheng, J., Baldi, P.: A machine learning information retrieval approach to protein fold recognition. *Bioinformatics* 22(12), 1456–1463 (2006)

# New Perspectives on Gene Family Evolution: Losses in Reconciliation and a Link with Supertrees\*

Cedric Chauve<sup>1</sup> and Nadia El-Mabrouk<sup>2</sup>

<sup>1</sup> Department of Mathematics, Simon Fraser University, Burnaby (BC), Canada

<sup>2</sup> Département Informatique et Recherche Opérationnelle, Université de Montréal,  
Montréal (QC), Canada

`cedric.chauve@sfu.ca, mabrouk@iro.umontreal.ca`

**Abstract.** Reconciliation between a set of gene trees and a species tree is the most commonly used approach to infer the duplication and loss events in the evolution of gene families, given a species tree. When a species tree is not known, a natural algorithmic problem is to infer a species tree such that the corresponding reconciliation minimizes the number of duplications and/or losses. In this paper, we clarify several theoretical questions and study various algorithmic issues related to these two problems. (1) For a given gene tree  $T$  and species tree  $S$ , we show that there is a single history explaining  $T$  and consistent with  $S$  that minimizes gene losses, and that this history also minimizes the number of duplications. We describe a simple linear-time and space algorithm to compute this parsimonious history, that is not based on the Lowest Common Ancestor (LCA) mapping approach; (2) We show that the problem of computing a species tree that minimizes the number of gene duplications, given a set of gene trees, is in fact a slight variant of a supertree problem; (3) We show that deciding if a set of gene trees can be explained using only apparent duplications can be done efficiently, as well as computing a parsimonious species tree for such gene trees. We also characterize gene trees that can be explained using only apparent duplications in terms of compatible triplets of leaves.

## 1 Introduction

Applying local similarity search tools to genomes of closely related species usually reveal large clusters of homologous genes, also called *gene families*. Such grouping by sequence similarity is not sufficient to infer a common function for genes. Indeed, in addition to orthologs which are copies in different species related through speciation, gene families are likely to contain paralogs, which are copies that have evolved by duplication. Paralogs are more likely to have acquired new functions. In addition to gene duplication, gene losses, arising through the pseudogenization of previously functional genes, also play a key role in the

---

\* Both authors are supported by grants from NSERC.

evolution of gene families [23,21,14,10,4,11,19]. Understanding the evolution of gene families is thus a fundamental question in functional genomics, but also in evolutionary biology and phylogenomics [28,32].

The most commonly used methods to infer evolutionary scenarios for gene families are based on the *reconciliation* approach that compares the species tree (describing the relationships among taxa) to the gene trees, and implicitly infers a set of gene duplications and losses. Given a species tree and a set of gene trees, there can be several reconciliations, and a natural approach is then to select one optimizing a given criterion, either combinatorial [22] or probabilistic [3]. Natural combinatorial criteria are the number of duplications (duplication cost), losses (loss cost) or both combined (mutation cost). The so called Lowest Common Ancestor (LCA) mapping between a gene tree and a species tree, introduced in [17] and is widely used studies [24,15,25,33,22,26,5,16,13], defines a reconciliation that minimizes both the duplication and mutation costs [16]. Although losses appear to be an important phenomenon in the evolution of a gene family, they have only recently been explicitly used as a parsimony criterion [7]. It can be computed efficiently, in linear time [33] or using a simple quadratic time algorithm [34]. When no preliminary knowledge on the species tree is given, a natural problem is to infer, from a set of gene trees, a species tree leading to a parsimonious evolution scenario, for a chosen cost. Similarly to the case of a known species tree, methods have been developed for the duplication and mutation costs [22,18,8]. For both criteria, the problem of inferring an optimal species tree given a set of gene trees is hard [22].

In this paper, we present various theoretical results related to the optimization problems of inferring, for a given gene tree (or a forest of gene trees), an evolution scenario minimizing a given cost, in both cases of a known and an unknown species tree.

In Section 3, we clarify the link between the duplication and loss cost criteria for reconciliation. Given a gene tree  $T$  and a species tree  $S$ , we show that there is a single history explaining  $T$  and consistent with  $S$  minimizing losses, and that this history also minimizes duplications. This refines recent results showing that there is a unique reconciliation minimizing the mutation cost [16]. We describe a simple linear-time reconciliation method, not based on the LCA mapping, computing this most parsimonious history. Although our new reconciliation algorithm is not the only one running in linear time [33,15], its implementation is simpler, and it highlights the important combinatorial role of gene losses regarding parsimonious evolution scenarios for gene families.

In Section 4, we describe the problem of computing, from a set of gene trees, a most parsimonious species tree for the duplication cost (the Minimum Duplication Problem), as an instance of the following restricted supertree problem: given a set of uniquely-leaf labeled gene trees where only the first speciation is resolved, compute a species tree that agrees with the largest number of such gene trees. Clearly, these two problems share some common ground in terms of goal – inferring a species tree from a collection of gene trees –, but differ in terms of data – duplicated leaves versus uniquely leaf-labeled trees – and

considered evolutionary mechanisms: duplication are ignored, at least explicitly, in supertree problems. The link between these two problems suggests that heuristics for the supertree problem, such as the min-cut greedy heuristic [29,27], are natural candidate heuristics for the Minimum Duplication Problem. The parallel with supertrees implies also an efficient algorithm to decide if a set of gene trees can be explained using only apparent duplications, as well as an efficient algorithm to compute all most parsimonious species trees for a set of such *MD-trees* (Minimum-Duplication trees). We also provide a combinatorial characterization of MD-trees as trees not containing triplets of leaves leading to contradictory phylogenetic information. The latter characterization of gene trees may be useful to detect ambiguous phylogenetic relationships or possible errors in a set of gene trees, as we illustrate in Section 5 on a simulated dataset.

## 2 Preliminaries

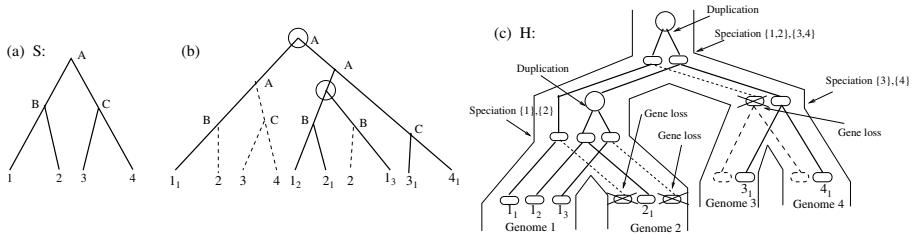
*Trees.* Let  $\mathcal{G} = \{1, 2, \dots, g\}$  be a set of integers representing  $g$  different species (genomes). A *species tree* on  $\mathcal{G}$  is a binary tree with exactly  $g$  leaves, where each  $i \in \mathcal{G}$  is the label of a single leaf. A *gene tree* on  $\mathcal{G}$  is a binary tree where each leaf is labeled by an integer from  $\mathcal{G}$  (each tree represents a gene family, where each leaf labeled  $i$  represents a gene copy located on genome  $i$ ).

For a given vertex  $x$  of a tree  $T$ , we denote by  $T_x$  the subtree of  $T$  rooted at  $x$  and by  $L(x)$  the subset of  $\mathcal{G}$  defined by the labels of the leaves of  $T_x$ .  $L(x)$  is called the *genome set of  $x$* . We denote by  $x_\ell$  and  $x_r$  the two children of  $x$ , if  $x$  is not a leaf, and by  $x_p$  its parent if  $x$  is not the root. An *expanded leaf* of  $T$  is a vertex  $x$  such that  $|L(x)| = 1$  and  $L(x) \neq L(x_p)$ , or  $x$  is the root of  $T$ . A *cherry* of a tree is an internal vertex  $x$  for which both children are expanded leaves.

*Reconciliation.* There are several definitions of reconciliation between a gene tree and a species tree. Here we define reconciliation in terms of subtree insertions, following an approach used in [16,7]. 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 a sequence of subtree insertions in  $T$ .

Given a gene tree  $T$  on  $\mathcal{G}$  and a species tree  $S$  on  $\mathcal{G}$ ,  $T$  is said to be *DS-consistent with  $S$*  (following the terminology used in [7]) if, for every vertex  $x$  of  $T$  such that  $|L(x)| \geq 2$ , there exists a vertex  $u$  of  $S$  such that  $L(x) = L(u)$  and one of the two following conditions (D) or (S) holds: (D) either  $L(x_r) = L(x_\ell)$ , or (S)  $L(x_r) = L(u_r)$  and  $L(x_\ell) = L(u_\ell)$ .

A *reconciliation* between a gene tree  $T$  and a species tree  $S$  is an extension  $R$  of  $T$  that is DS-consistent with  $S$  (this definition is easily shown to be equivalent to other definitions of reconciliation [3,12]). Such a reconciliation between  $T$  and  $S$  implies an unambiguous evolution scenario for the gene family  $T$  where a vertex of  $R$  that satisfies property (D) represents a duplication (the number of duplications induced by  $R$  is denoted by  $d(R, S)$ ), and an inserted subtree represents a gene loss (the number of gene losses induced by  $R$  is denoted by  $\ell(R, S)$ ). Vertices of  $R$  that satisfy property (S) represent speciation events (see Fig. II).



**Fig. 1.** (a) A species tree  $S$ ; (b) The reconciliation  $R$  of  $S$  with the gene tree  $T$  represented by plain lines. Dotted lines represent subtree insertions (3 insertions). The correspondence between vertices of  $R$  and  $S$  is indicated by vertices labels. Circles represent duplications. All other internal vertices of  $R$  are speciation vertices; (c) Evolution scenario resulting from  $R$ . Each oval is a gene copy.

Given a gene tree  $T$ , it is immediate to see that every vertex  $x$  of  $T$  such that  $L(x_\ell) \cap L(x_r) \neq \emptyset$  will always be a duplication vertex in any reconciliation  $R$  between  $T$  and  $S$ . Such a vertex is called an *apparent duplication vertex* (or just apparent duplication for short). For example, in the gene tree  $T$  represented by plain lines in Fig. 1b., both duplication vertices are apparent duplications.

The notion of reconciliation can naturally be extended to the case of a set, or *forest*, of gene trees  $\mathcal{F} = \{T_1, \dots, T_m\}$ : a reconciliation between  $\mathcal{F}$  and  $S$  is a set  $\mathcal{R} = \{R_1, \dots, R_m\}$  of reconciliations, respectively for  $T_1, \dots, T_m$ , such that each  $R_i$  is DS-consistent with  $S$ . We denote by  $\mathcal{R}(\mathcal{F}, S)$  the set of all reconciliations between  $\mathcal{F}$  and  $S$ .

*Optimization problems:* We consider three cost measures for a reconciliation  $\mathcal{R}(\mathcal{F}, S)$  between a gene tree forest  $\mathcal{F} = \{T_1, \dots, T_m\}$  and a species tree  $S$ . The *duplication cost* is given by  $d(\mathcal{R}, S) = \sum_{i=1}^m d(R_i, S)$ , the *loss cost* by  $\ell(\mathcal{R}, S) = \sum_{i=1}^m \ell(R_i, S)$  and the *mutation cost* by  $m(\mathcal{R}, S) = \sum_{i=1}^m d(R_i, S) + \ell(R_i, S)$ . For a given cost measure  $C$  (here  $d$ ,  $\ell$  or  $m$ ), there are two natural combinatorial optimization problems, depending on whether a species tree is known or not.

#### MINIMUM RECONCILIATION $C$ PROBLEM:

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

**Output:** A reconciliation  $\mathcal{R}$  with minimum cost  $C(\mathcal{R}, S)$ .

#### MINIMUM $C$ PROBLEM:

**Input:** A gene tree forest  $\mathcal{F}$  on  $\mathcal{G}$ ;

**Output:** A species tree  $S$  such that  $\min_{\mathcal{R} \in \mathcal{R}(\mathcal{F}, S)} C(\mathcal{R}, S)$  is minimum.

The MINIMUM DUPLICATION PROBLEM and MINIMUM MUTATION PROBLEM with multiple gene trees are NP-complete [22]. The complexity status of the MINIMUM LOSS PROBLEM is still unknown.

## 3 Reconciled Trees

Let  $T$  be a gene tree on  $\mathcal{G}$ . We assume that a species tree  $S$  is already known for  $\mathcal{G}$ . The LCA mapping between  $T$  and  $S$ , denoted by  $M$ , maps every vertex

$x$  of a gene tree  $T$  towards the Lowest Common Ancestor (LCA) of  $L(x)$  in  $S$ . This mapping induces a reconciliation between  $T$  and  $S$  (see [12] for example) where an internal vertex  $x$  of  $T$  leads to a duplication vertex if  $M(x_\ell) = M(x)$  and/or  $M(x_r) = M(x)$ . We denote by  $M(T, S)$  the reconciliation between  $T$  and  $S$  defined by the LCA mapping. It has been shown recently [16] that  $M(T, S)$  is the only reconciliation that minimizes the mutation cost, while there can be several reconciliations that minimize the duplication cost. The following theorem refines this result.

**Theorem 1.** *Given a gene tree  $T$  and a species tree  $S$ ,  $M(T, S)$  minimizes the duplication, loss and mutation costs. Moreover,  $M(T, S)$  is the only reconciliation between  $T$  and  $S$  that minimizes the loss cost and minimizes the mutation cost.*

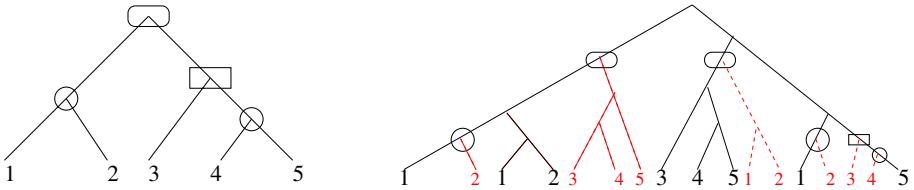
In [7, Prop. 1], it was shown that  $M(T, S)$  is optimal for the loss cost. On the other hand, the fact that  $M(T, S)$  is optimal for the duplication cost is a well known result (see [16] for a recent reference for example). It follows that  $M(T, S)$  is optimal for each of the three costs. It then remains to show that  $M(T, S)$  is the unique reconciliation between  $T$  and  $S$  that is optimal for the loss cost. This would imply that  $M(T, S)$  is also the unique reconciliation that minimizes the mutation cost (a result proved in [16] although in a more complicated way) and complete our proof. To do so, we rely on a new simple linear-time algorithm that computes a reconciliation between  $T$  and  $S$  and minimizes the loss cost.

Algorithm Minimum-Reconciliation described below takes a gene tree  $T$  and a species tree  $S$  as input, and returns a reconciled tree  $R$ . Roughly speaking, the algorithm proceeds as follows: it traverses the gene tree  $T$  from the leaves to the root, and completes by subtree insertions the subtrees of  $T$  corresponding to the successive speciation events of  $S$ , from the latest ones to the earliest one. An example is given in Figure 2.

ALGORITHM MINIMUM-RECONCILIATION ( $T, S$ ):

1. Set  $R = T$ ,  $R' = R$  and  $S' = S$ ;
2. While  $S'$  is not reduced to a single vertex.
  - (a) Visit the expanded leaves of  $R'$  given  $S'$ 
    - i. Let  $x$  be the current expanded leaf of  $R'$  and  $y$  its sibling;
    - ii. Let  $u$  be the leaf of  $S'$  with  $L(u) = L(x)$  and  $v$  its sibling;
    - iii. If  $L(y) \neq L(v)$  then insert in  $R$  on the branch between  $x$  and  $x_p$  a leaf labeled by  $L(v)$ ;
  - (b) Reduce each subtree of  $S'$  and  $R'$  corresponding to a cherry of  $S'$  to a single leaf labeled with the cherry genome set;
3. Return ( $R$ );

**Theorem 2.** *Given a gene tree  $T$  on  $\mathcal{G}$  and a species tree  $S$  for  $\mathcal{G}$ , ALGORITHM MINIMUM-RECONCILIATION reconstructs the unique reconciliation between  $T$  and  $S$  that minimizes the number of gene losses. It can be implemented to run in  $O(n)$  time and space.*



**Fig. 2.** Left: speciation tree  $S$ . Right: reconciliation tree  $R$  constructed over the gene tree  $T$  represented by solid lines. Three executions of Algorithm Minimum-Reconciliation loop of step 2 are required, and the successive lists of considered cherries of  $S'$  are represented by circles for the first iteration, a rectangle for the second iteration and ovals for the third iteration.

*Proof.* At the end of this algorithm, the resulting tree  $R$  is DS-consistent with  $S$  as it recursively completes cherries in  $R$  according to  $S$ . Therefore,  $R$  is a reconciliation of  $T$  and  $S$ . Moreover, as the algorithm considers each vertex of  $T$  exactly once, it can be implemented in a single post-order traversal of  $T$ . Handling vertices labeled by genome sets can be done efficiently by replacing such sets by integers, as was done in the algorithm DS-RECOGNITION described in [7] that decides if a gene tree can be explained without gene loss by comparing the cherries of the gene tree to the cherries of the species tree. In fact ALGORITHM MINIMUM-RECONCILIATION can be implemented as a direct extension of algorithm DS-RECOGNITION. Combined with the fact that subtree insertions can be implemented in constant times using pointers on vertices of  $S$ , this leads to a linear time and space complexity with low constants and using only simple data structures.

From now on, we denote by  $\text{MinR}(S, T)$  the reconciliation tree obtained by ALGORITHM MINIMUM-RECONCILIATION, and we show that  $\text{MinR}(S, T)$  is the only reconciliation between  $T$  and  $S$  that minimizes the number of losses. To prove this, we use the fact that each gene loss is represented by a subtree insertion. Given a species tree  $S$ , a gene tree  $T$ , and a cherry  $u$  of  $S$  such that  $L(u) = \{a, b\}$ , it follows from the definition of DS-consistency that every expanded leaf  $x$  of  $T$  with  $L(x) = \{a\}$  or  $L(x) = \{b\}$  has to be completed, if required, by inserting a sibling in order to form a cherry labeled  $\{a, b\}$ . Hence, all subtree insertions performed by ALGORITHM MINIMUM-RECONCILIATION when visiting the set of expanded leaves of  $T$  (first iteration of step 2) are required in order to extend  $T$  into a tree that is DS-consistent with  $S$ . The same property holds recursively to the following iterations of step 2, which implies that all subtree insertions performed by ALGORITHM MINIMUM-RECONCILIATION are necessary in order to extend  $T$  into a tree that is DS-consistent with  $S$ . Combined with the fact that, at the end of the algorithm,  $\text{MinR}(S, T)$  is DS-consistent with  $S$  and the fact that the number of subtree insertions is the number of gene losses induced by  $\text{MinR}(S, T)$ , this completes the proof.  $\square$

*Remark 1.* It follows from Theorem 1 that minimizing losses results in minimizing duplications, as the unique solution to the MINIMUM RECONCILIATION

LOSS PROBLEM is a solution to the MINIMUM RECONCILIATION DUPLICATION PROBLEM. The converse is not true, as more than one solution may exist, in general, for the MINIMUM RECONCILIATION DUPLICATION PROBLEM. Stated differently, the loss cost criterion is more constraining than the duplication cost criterion for reconciliation. This property does not hold anymore in the case of inferring a species tree from a set of gene trees: the species tree that minimizes losses does not always minimizes duplications, and conversely. Even a weaker property does not hold in this case: there is not always a common solution to the MINIMUM DUPLICATION PROBLEM and the MINIMUM LOSS PROBLEM.

## 4 Gene Duplication and Supertrees

The general supertree problem can be stated as follows: given a set of uniquely leaf-labeled gene trees (i.e. in each gene tree no two leaves have the same label), compute a species tree optimizing some combinatorial criterion. A natural criterion is to maximize the number of input gene trees that are DS-consistent with the species tree (called a supertree).

*The Minimum Duplication Problem is a supertree problem.* We follow [31] to introduce terminology on supertrees. Given two uniquely leaf-labeled trees  $T$  and  $T'$ , possibly non-binary, we say that  $T'$  refines  $T$  (denoted by  $T' \rightarrow T$ ) if  $T$  can be obtained from  $T'$  by a sequence of contraction of internal edges of  $T'$ . Given a species tree  $S$  on  $\mathcal{G}$  and a subset  $\mathcal{H}$  of  $\mathcal{G}$ , we denote by  $S|_{\mathcal{H}}$  the induced species tree on  $\mathcal{H}$ , obtained by first removing all vertices  $x$  of  $S$  such that  $L(x) \cap \mathcal{H} = \emptyset$  and next removing all vertices of degree two. A, possibly non-binary, gene tree  $T$  is *consistent* with a species tree  $S$  if  $S|_{L(T)} \rightarrow T$ , and *inconsistent* otherwise. Finally, a uniquely leaf-labeled tree  $T$  on  $\mathcal{G}$  is said to be a *bipartition* of  $\mathcal{G}$  if  $T$  contains only three internal vertices  $x$  (the root),  $x_r$  and  $x_\ell$ , and  $L(x_r) \cap L(x_\ell) = \emptyset$  ( $x_r$  and  $x_\ell$  are possibly non-binary vertices). For a given set  $\mathcal{B} = \{B_1, \dots, B_m\}$  of bipartitions of  $\mathcal{G}$  and a species tree  $S$  on  $\mathcal{G}$ , we denote by  $c(\mathcal{B}, S)$  the number of  $B_i$ 's that are inconsistent with  $S$ .

We now introduce a simple variant of the general supertree problem, where each gene tree indicates a single speciation.

MINIMUM BIPARTITION INCONSISTENCY SUPERTREE (MBIS) PROBLEM

**Input:** A set of bipartitions  $\mathcal{B}$  of  $\mathcal{G}$ ;

**Output:** A species tree  $S$  such that  $c(\mathcal{B}, S)$  is minimum.

Given a binary gene tree  $T$  and a vertex  $x$  of  $T$  that is not an apparent duplication, we define the bipartition associated to  $x$ , denoted  $B(T, x)$ , as the bipartition with root  $y$  and internal vertices  $y_r$  (resp.  $y_\ell$ ), such that  $L(y_r) = L(x_r)$  and  $L(y_\ell) = L(x_\ell)$ . Given a forest  $\mathcal{F} = \{T_1, \dots, T_m\}$  of binary gene trees, we denote by  $\mathcal{B}(\mathcal{F})$  the set of bipartitions associated to all vertices of the trees of  $\mathcal{F}$  that are not apparent duplications.

**Theorem 3.** Let  $\mathcal{F}$  be a forest of gene trees on  $\mathcal{G}$  and  $k$  be the number of apparent duplications present in the trees of  $\mathcal{F}$ . Then, for any species tree  $S$  on  $\mathcal{G}$ ,  $d(\mathcal{F}, S) = k + c(\mathcal{B}(\mathcal{F}), S)$ .

*Proof.* Apparent duplications are associated to duplications for every species tree  $S$ . Hence the remaining duplications (there are  $d(\mathcal{F}, S) - k$  such duplications, for a given species tree  $S$ ) are not apparent duplications. Let  $x$  be such a vertex, belonging to a tree  $T$  of  $\mathcal{F}$ . As the reconciliation with  $S$  implies that  $x$  is a duplication, without loss of generality, we can assume that  $M(x_r) = M(x)$  ( $M$  is the LCA mapping between  $T$  and  $S$ ). Hence there are two elements  $a, b \in \mathcal{G}$  such that  $a, b \in L(x_r)$  and, if  $M(x) = u$ ,  $a \in L(u_r)$  and  $b \in L(u_\ell)$ . This implies that  $S|_{L(x)}$  does not refine  $B(T, x)$ , and thus  $B(T, x)$  is not consistent with  $S$ .

Conversely, let  $B = B(T, x)$  be a bipartition of  $\mathcal{B}(\mathcal{F})$  that is not consistent with  $S$ . It is clear that if  $S|_{L(B)}$  refines  $B$ , then it can be transformed into  $B$  by contracting all internal edges that are not incident to its root. Thus, if  $\mathcal{B}(\mathcal{F})$  is not consistent with  $S$ , there should be two elements  $a, b \in \mathcal{G}$  that do not belong to a proper subtree of  $S|_{L(B)}$ , but belong to a proper subtree of  $B$  (say the subtree rooted at  $x_r$ ). This implies that  $M(x) = M(x_r)$  and then that  $x$  is no an apparent duplication but counts for a duplication when reconciled with  $S$ .  $\square$

This result shows that inferring a most parsimonious species tree for the duplication cost is equivalent to a restricted supertree problem that considers only very pathological input gene trees (bipartitions). Note however that despite the very restricted nature of its input trees, the MBIS PROBLEM is NP-complete, which is deduced from the NP-completeness of the MINIMUM DUP. PROBLEM. Another simple variant of the supertree problem, where input gene trees are rooted triplets (the Max. Triplet Consistency Supertree prob.), has been shown to be NP-complete [6].

The link between the two problems has the interesting consequence that heuristics for the supertree problem are then natural candidate heuristics for the MINIMUM DUPLICATION PROBLEM. In particular, min-cut based heuristics such as those developed in [29, 27] can be directly applied to bipartitions (see Section 5). Such heuristics can then be seen as greedy approaches to the MINIMUM DUPLICATION PROBLEM, that, as far as we know, has never been used for the MINIMUM DUP. PROBLEM, while it follows very naturally from its description as a supertree problem. The resulting species tree can then be used as a starting point for local-search algorithms such as the one presented in [2].

*Minimum Duplication trees and compatible trees.* A gene tree forest  $\mathcal{F}$  is said to be a *Minimum Duplication forest* (from now an *MD-forest*) if there exists a species tree  $S$  such that  $d(\mathcal{F}, S)$  is exactly the number of apparent duplications present in the trees of  $\mathcal{F}$ . In such case,  $\mathcal{F}$  is said to be *MD-consistent* with  $S$ .

**Theorem 4.** Deciding whether a forest of gene trees  $\mathcal{F}$  is an MD-forest and computing the set of all species trees  $S$  such that  $\mathcal{F}$  is MD-consistent with  $S$  can be done in polynomial time and space.

*Proof.* Assume that  $\mathcal{F}$  contains  $p$  vertices that are non-apparent duplications, we first note that  $\mathcal{B}(\mathcal{F})$  contains  $O(p)$  bipartitions. Following Theorem 3, the problem of deciding if  $\mathcal{F}$  is an MD-forest reduces to deciding if  $c(\mathcal{B}(\mathcal{F}), S) = 0$ . Several algorithms exist that answer this question in polynomial when the input consists of rooted triplets [1] or unconstrained rooted binary trees [9,20]. However, these algorithms can be easily adapted to our situation. For example, the algorithm of [1], as described in [30], can be used if we simply define the edges of connectivity graph as follows: for two elements  $i, j \in \mathcal{G}$ , there is an edge between  $i$  and  $j$  if and only if there is a bipartition  $B$  of  $\mathcal{F}$  such that  $i$  and  $j$  belong to the same proper subtree of  $B$ .

To compute the set of all species trees such that  $\mathcal{F}$  is MD-consistent with  $S$ , we can use the polynomial time and space algorithm of [9] by replacing each subtree rooted at a non-binary vertex  $x$  of the bipartitions  $\mathcal{B}(\mathcal{F})$ , with leaf set  $L(x) = \{i_1, \dots, i_k\}$  such that  $i_1 < i_2 < \dots < i_k$  by the caterpillar tree  $(i_1, (i_2, \dots (i_{k-1}, i_k), \dots))$ .  $\square$

We now provide a simple combinatorial characterization of MD-trees and MD-forests in terms of triplets of species. A vertex of  $T$  is said to *split* 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  $b$ . Let  $x$  and  $y$  be two vertices of a gene tree  $T$ , that are non-apparent duplications. They *disagree* on a triplet  $\{a, b, c\}$  of species if they split  $\{a, b, c\}$  in different ways (say  $\{a, b; c\}$  and  $\{a, c; b\}$  for example). A gene tree  $T$  on  $\mathcal{G}$  is *compatible* if no pair of non-apparent duplication vertices disagrees on any triplet of species. For a given species tree  $S$  on  $\mathcal{G}$ ,  $T$  is said to be a *compatible gene tree consistent with  $S$*  if it is compatible, and every triplet of species  $\{a, b, c\}$  is split in the same way by the LCA of these species in  $S$  and by any non-apparent duplication vertex of  $T$  that split them. These definitions extend naturally to a forest of gene trees.

**Theorem 5.** *Let  $\mathcal{F}$  be a gene tree forest on  $\mathcal{G}$ , and  $S$  be a species tree for  $\mathcal{G}$ . Then  $\mathcal{F}$  is a compatible gene tree forest consistent with  $S$  if and only if  $\mathcal{F}$  is an MD-forest consistent with  $S$ .*

*Proof.* We consider a compatible gene tree  $T$ , as the proof generalizes in a straightforward way to forests.

Suppose first that  $T$  is not a compatible gene tree. Then there are two non-apparent duplication vertices  $v$  and  $w$  that split a triplet of species  $\{a, b, c\}$  into two different ways, say  $\{a, b; c\}$  for  $v$  and  $\{a, c; b\}$  for  $w$ . If  $\{a, b, c\}$  are split into  $\{a, b; c\}$  in a species tree  $S$ , then  $w$  is a duplication vertex as it maps to the same vertex of  $S$  than its child that contains leaves labeled by  $a$  and  $c$ , and then  $T$  is not an MD-tree. Similarly,  $v$  is a duplication vertex if  $\{a, b, c\}$  are split into  $\{a, c; b\}$  in  $S$  and both  $v$  and  $w$  are duplication vertices if  $\{a, b, c\}$  are split into  $\{b, c; a\}$  in  $S$ .

Suppose that  $T$  is a compatible gene tree that is not consistent with  $S$ . Then there is a triplet  $\{a, b, c\}$  of elements of  $\mathcal{G}$  and a non-apparent duplication vertex  $v$  of  $T$  that splits  $\{a, b, c\}$  in a different way than they are in  $S$ . W.l.o.g, let assume that  $v$  splits them into  $\{a, b; c\}$  while in  $S$  they are split into  $\{b, c; a\}$ .

Then  $v$  is a duplication vertex, as it maps to the same vertex of  $S$  than its child that contains leaves labeled by  $a$  and  $b$ . Therefore, as  $T$  contains a vertex that is a duplication vertex but not an apparent duplication vertex,  $T$  is not an MD-tree consistent with  $S$ .

Conversely, suppose that  $T$  is not an MD-tree consistent with  $S$ . Then there is a vertex  $v$  in  $T$  that is a duplication vertex but not an apparent duplication vertex. As  $v$  is a duplication vertex,  $v$  maps to the same vertex of  $S$  than one of its child  $v_\ell$  or  $v_r$ , let say its left vertex  $v_\ell$ . Moreover, as  $v$  is not an apparent duplication, there are two leaves  $x_a$  and  $x_b$  of  $T_{v_\ell}$  labeled respectively  $a$  and  $b$ , and a leaf  $x_c$  in  $T_{v_r}$  labeled  $c$  that imply that  $\{a, b, c\}$  is split into  $\{a, b; c\}$  by  $v$ , while  $\{a, b, c\}$  is split into  $\{b, c; a\}$  in  $S$ . Therefore,  $T$  is not a compatible gene tree consistent with  $S$ .  $\square$

**Corollary 1.** *A gene tree forest  $\mathcal{F}$  on  $\mathcal{G}$  is a compatible gene tree forest if and only if  $\mathcal{F}$  is an MD-forest.*

*Proof.* We will prove the result on a single gene tree  $T$ . The generalization to a forest  $F$  is straightforward.

“ $\Leftarrow$ ” This case follows directly from the previous proof.

“ $\Rightarrow$ ” Suppose that  $T$  is a compatible tree. Then for any triplet  $\{a, b, c\}$  of distinct elements of  $\mathcal{G}$ , any non-apparent duplication vertex of  $T$  splits them in the same way. Then there is a DLS-history  $H$  for  $T$  leading to a species tree  $S$  such that, for any triplet  $\{a, b, c\}$  of distinct elements of  $\mathcal{G}$ ,  $S$  splits  $\{a, b, c\}$  in the same way than any non-apparent duplication vertex of  $T$ . It follows that any vertex  $v$  of  $T$  that is not an apparent duplication vertex is not a duplication vertex for  $H$ .  $\square$

From a theoretical point of view, the above results are interesting as they can be seen to be the MD-trees counterpart of a well known result about supertrees stating that deciding if, given a set of gene trees, there is a species tree that agrees with all of them, is equivalent to checking the same property on all triplets induced by these gene trees. From a practical point of view, triplets of species can be used to point at possibly ambiguous phylogenetic relationships and possibly misplaced genes in the gene tree, as we illustrate in the next section.

## 5 Experimental Results

We generated gene families, as in [7], using the species tree of 12 *Drosophila* species given in [19] (including branch length) and a birth-and-death process, starting from a single ancestral gene, with four different gene gain/loss rates (expected number of events by million years): 0.02 (the highest rate identified in [19]), 0.05, 0.1 and 0.2. For each rate, we generated 250 gene trees, described in Table 1. Note that more than 95% of gene duplications lead to an apparent duplication vertex. Note also that the number of informative bipartitions (i.e. bipartitions with at least two leaves) induced by non-apparent duplication vertices decreases dramatically as the rate of gene gain/loss increases.<sup>1</sup>

<sup>1</sup> All the material is available at: <http://www.cecm.sfu.ca/~cchauve/SUPP/RECOMB09>

**Table 1.** Characteristics of simulated gene trees. Considered bipartitions are those containing more than two species.

Rate	Nb. of Duplications	Nb. of Losses	Nb. of Genes	Nb. of Int. vertices	Nb. of Apparent duplications	Nb. of Bipartitions
0.02	1080	976	3014	2752	1057	831
0.05	2018	1366	3622	3360	1948	593
0.1	3126	1603	4376	4114	3007	358
0.2	6123	2552	7709	7447	5875	429

For each of the four datasets, we extracted the informative bipartitions induced by the non-apparent duplication vertices. We then used the Modified Min-Cut algorithm described in [27] to compute a species tree from these bipartitions. With rates 0.02 and 0.04, this species tree is the correct species tree, while with rate 0.1, it differs from the correct one by a single branch swap, and with rate 0.2, it differs from the correct one by the fact that two consecutive binary nodes have been replaced by a single quaternary node. The fit statistic associated to the inferred species tree, that measures how well it agrees with the bipartitions, is very high, ranging from 0.98 to 0.855 (maximum fit is 1). This shows the effectiveness of the supertree approach using bipartitions, at least on a dataset of relatively close species where few vertices indicating a speciation are false positive.

We also studied the phylogenetic signal given by triplets of species that were split by non-apparent duplication vertices. With rates 0.02 and 0.05, for each triplet of species, there is a phylogeny that appears in most cases. However, with rates 0.1 and 0.2, among the triplets that appear a significant number of times (at least 50 times), the ones where the dominant phylogeny appears in less than 90% of the bipartitions splitting this triplet, contain the two species involved in the branch swap or species involved in the unresolved node that differs from the correct species tree. This illustrates the interest in using triplets of species that are split by non-apparent duplication vertices to point at possible locations of an inferred species tree that are associated with a weaker phylogenetic signal.

## 6 Conclusion

In this paper, we show that minimizing losses is a more constraining criterion than minimizing duplications for reconciliation. This highlights the importance of the former criterion from a combinatorial point of view, although it has been rarely considered alone in reconciliation approaches. Our second main result relates the problem of inferring a species tree minimizing duplications (given a set of gene trees), to a supertree problem. This link has important implications, as it allows, for example, to use min-cut based algorithms to infer a species tree from a set of gene trees. Moreover, this link with supertree problems allowed us to highlight properties of gene trees that could be exploited for gene tree correction. Indeed, a major problem with reconciliation, and its generalization

to an unknown species tree, is that errors in gene trees usually lead to erroneous duplication/loss histories, and potentially to a wrong species tree. Therefore, eliminating a number of potentially misleading gene copies is an important preliminary step to any reconciliation approach. In this context, non-apparent duplications, or triplets leading to contradictory phylogenetic informations, may point at gene copies that are possibly erroneously placed in the gene tree. Our preliminary experimental results tend to support this strategy for pruning gene trees.

## References

1. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput.* 10, 405–421 (1981)
2. Bansal, M.S., Burleigh, J.G., Eulenstein, O., Wehe, A.: Heuristics for the gene-duplication problem: A  $\Theta(n)$  speed-up for the local search. In: Speed, T., Huang, H. (eds.) *RECOMB 2007. LNCS (LNBI)*, vol. 4453, pp. 238–252. Springer, Heidelberg (2007)
3. Arvestad, L., Berglung, A.-C., Lagergren, J., Sennblad, B.: Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In: *RECOMB 2004*, pp. 326–335 (2004)
4. 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 Biol.* 7, R43 (2006)
5. Bonizzoni, P., Della Vedova, G., Dondi, R.: Reconciling a gene tree to a species tree under the duplication cost model. *Theoret. Comput. Sci.* 347, 36–53 (2005)
6. Bryant, D.: Hunting for trees, building trees and comparing trees: theory and methods in phylogenetic analysis. Ph.D. thesis, Dept. of Math., Univ. of Canterbury, New Zealand (1997)
7. Chauve, C., Doyon, J.-P., El-Mabrouk, N.: Gene family evolution by duplication, speciation and loss. *J. Comput. Biol.* 15, 1043–1062 (2008)
8. Chen, K., Durand, D., Farach-Colton, M.: NOTUNG: a program for dating gene duplications and optimizing gene family trees. *J. Comput. Biol.* 7, 429–444 (2000)
9. Constantinescu, M., Sankoff, D.: An efficient algorithm for supertrees. *J. Classification* 12, 101–112 (1995)
10. Cotton, J.A., Page, R.D.M.: Rates and patterns of gene duplication and loss in the human genome. *Proc. R. Soc. Lond. B* 272, 277–283 (2005)
11. Demuth, J.P., De Bie, T., Stajich, J., Cristianini, N., Hahn, M.W.: The evolution of mammalian gene families. *PLoS ONE* 1, e85 (2006)
12. Doyon, J.-P., Chauve, C., Hamel, S.: Algorithms for exploring the space of gene tree/species tree reconciliations. In: Nelson, C.E., Vialette, S. (eds.) *RECOMB-CG 2008. LNCS (LNBI)*, vol. 5267, pp. 1–13. Springer, Heidelberg (2008)
13. Durand, D., Halldórsson, B.V., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comput. Biol.* 13, 320–3354 (2006)
14. Eichler, E.E., Sankoff, D.: Structural dynamics of eukaryotic chromosome evolution. *Science* 301, 793–797 (2003)
15. Eulenstein, O., Mirkin, B., Vingron, M.: Comparison of annotating duplication, tree mapping, and copying as methods to compare gene trees with species trees. In: Mathematical hierarchies and biology. *DIMACS Series Discrete Math. Theoret. Comput. Sci.*, vol. 37, pp. 71–93 (1997)

16. Gorecki, P., Tiutyn, J.: DLS-trees: a model of evolutionary scenarios. *Theoret. Comput. Sci.* 359, 378–399 (2006)
17. 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. *Syst. Zool.* 28, 132–163 (1979)
18. Hallett, M.T., Lagergren, J.: New algorithms for the duplication-loss model. In: RECOMB 2000, pp. 138–146 (2000)
19. Hahn, M.W., Han, M.V., Han, S.-G.: Gene family evolution across 12 *Drosophila* genomes. *PLoS Genet.* 3, e197 (2007)
20. Henzinger, M.R., King, V., Warnow, T.: Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Evolutionary Biology. *Algorithmica* 24, 1–13 (1999)
21. Lynch, M., Conery, J.S.: The evolutionary fate and consequences of duplicate genes. *Science* 290, 1151–1155 (2000)
22. Ma, B., Li, M., Zhang, L.: From gene trees to species trees. *SIAM J. Comput.* 30, 729–752 (2000)
23. Ohno, S.: Evolution by gene duplication. Springer, Heidelberg (1970)
24. Page, R.D.M.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.* 43, 58–77 (1994)
25. Page, R.D.M., Charleston, M.A.: Reconciled trees and incongruent gene and species trees. Mathematical hierarchies and biology. *DIMACS Series Discrete Math. Theoret. Comput. Sci.* 37, 57–70 (1997)
26. Page, R.D.M.: GeneTree: comparing gene and species phylogenies using reconciled trees. *Bioinformatics* 14, 819–820 (1998)
27. Page, R.D.M.: Modified mincut supertrees. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 537–551. Springer, Heidelberg (2002)
28. Sanderson, M.J., McMahon, M.M.: Inferring angiosperm phylogeny from EST data with widespread gene duplication. *BMC Evol. Biol.* 7, S3 (2007)
29. Semple, C., Steel, M.: A supertree method for rooted trees. *Discrete Appl. Math.* 105, 147–158 (2000)
30. Snir, S., Rao, S.: Using max cut to enhance rooted trees consistency. *IEEE/ACM Trans. Comput. Biol. and Bioinform.* 3, 323–333 (2006)
31. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classification* 9, 91–116 (1992)
32. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Natural history and evolutionary principles of gene duplication in fungi. *Nature* 449, 54–61 (2007)
33. Zhang, L.X.: On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *J. Comput. Biol.* 4, 177–188 (1997)
34. Zmasek, C.M., Eddy, S.R.: A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics* 17, 821–828 (2001)

# A Probabilistic Graphical Model for Ab Initio Folding

Feng Zhao<sup>1</sup>, Jian Peng<sup>1</sup>, Joe DeBartolo<sup>2</sup>, Karl F. Freed<sup>3</sup>,  
Tobin R. Sosnick<sup>2</sup>, and Jinbo Xu<sup>1,\*,\*\*</sup>

<sup>1</sup> Toyota Technological Institute at Chicago, Chicago, IL 60637  
[{lfzhao,pengjian,j3xu}@tti-c.org](mailto:{lfzhao,pengjian,j3xu}@tti-c.org)

<sup>2</sup> Department of Biochemistry and Molecular Biology, the University of Chicago,  
Chicago, IL 60637  
[{bartolo, trsosnick}@uchicago.edu](mailto:{bartolo, trsosnick}@uchicago.edu)

<sup>3</sup> Department of Chemistry, the University of Chicago, Chicago, IL 60637  
[freed@uchicago.edu](mailto:freed@uchicago.edu)

**Abstract.** Despite significant progress in recent years, *ab initio* folding is still one of the most challenging problems in structural biology. This paper presents a probabilistic graphical model for ab initio folding, which employs Conditional Random Fields (CRFs) and directional statistics to model the relationship between the primary sequence of a protein and its three-dimensional structure. Different from the widely-used fragment assembly method and the lattice model for protein folding, our graphical model can explore protein conformations in a continuous space according to their probability. The probability of a protein conformation reflects its stability and is estimated from PSI-BLAST sequence profile and predicted secondary structure. Experimental results indicate that this new method compares favorably with the fragment assembly method and the lattice model.

**Keywords:** protein structure prediction, ab initio folding, conditional random fields (CRFs), directional statistics, fragment assembly, lattice model.

## 1 Introduction

Various genome sequencing projects have generated millions of proteins with unknown structures. To fully understand the biological functions of these proteins, the knowledge of their three-dimensional structures is essential. Many computational methods have been developed to predict the structure of a protein from its primary sequence. These methods can be roughly classified into two categories: ab initio folding and comparative modeling. Although ab initio folding is not as accurate as comparative modeling, ab initio folding is still being actively studied for many purposes. The excellent performance of several groups (e.g., Zhang's I-TASSER [1], Baker's Robetta [2] and

\* The first two authors contribute equally to this paper. Correspondence should be addressed to Dr. Jinbo Xu.

\*\* Author contributions: J.X. designed and performed research and wrote the paper; J.P. performed research; F.Z. performed research and analyzed data; J.D., K.F., and T.S. helped with energy function.

Skolnick's TASSER [3] in recent CASP (Critical Assessment of Structure Prediction) events [45] indicates that by combining comparative modeling and ab initio folding, it is possible to improve the accuracy of a model built from template. Despite significant progress in recent years, ab initio folding is still one of the most challenging problems in structural biology.

Ab initio folding based on fragment assembly [6][7][8][9][10][11][12][13] and the lattice model [14][15][16] has been extensively studied. These two popular methods and their combination for ab initio protein structure prediction have achieved great success in CASP competitions [17][4][18]. For example, the widely-used fragment assembly program Robetta [19] is one of the most accurate ab initio folding programs. The Zhang-Server [1], which combines the lattice model, fragment assembly and threading-generated distance restraints, has outperformed all the other structure prediction servers in both CASP7 and CASP8. Although these methods demonstrate exciting results, several important issues have yet been addressed. First, due to the limited number of experimental protein structures in the Protein Data Bank (PDB), it is still very difficult to have a library of even moderate-sized fragments that can cover all the possible local conformations of a protein, especially in loop regions. Second, the conformation space defined by a fragment library is discrete in nature. The predicted structural model is not sampled from a continuous space. This discrete nature may restrict the search space and cause loss of prediction accuracy. Fragment-HMM [20], a close variant of Robetta, can sample from a continuous space, but still has the coverage problem. The lattice model used in the TOUCHSTONE programs [15][16] does not have the coverage problem, but it samples protein conformations from a three-dimensional lattice with finite resolution. That is, the conformation space defined by a lattice model is also discrete. More importantly, the sampled conformations may not have a protein-like local structure because the TOUCHSTONE programs do not sample a conformation based upon the primary sequence of a protein. Instead, the TOUCHSTONE programs use a few short-range statistical potentials in its energy function to guide the formation of protein-like local structure.

In addition to the fragment assembly method and the lattice model, there are also a few methods that attempt to sample protein conformations from a continuous space according to the probability of a conformation. The probability of a conformation approximately reflects its stability and is estimated from sequence information. In [21], Feldman and Hogue developed a program FOLDTRAJ, which implements a probabilistic all-atom protein conformation sampling algorithm. Tested on three small proteins 1VII, 1ENH, and 1PMC, FOLDTRAJ can obtain the best structural models with RMSD from native being 3.95, 5.12, and 5.95 Å, respectively, out of 100,000 decoys for each protein. However, neither sequence profile nor the local conformation-dependency between two adjacent residues is used in FOLDTRAJ to estimate the probability of a conformation. Therefore, FOLDTRAJ cannot generate models with quality comparable with Robetta. Recently, Hamelryck *et al* have developed an FB5-HMM model [22] and a Torus-HMM model [23] for protein conformation sampling in a continuous space. The HMM models not only capture the relationship between backbone angles and their corresponding primary sequence and predicted secondary structure, but also consider the angle-dependency between two adjacent residues. They demonstrated that their

Torus-HMM model can generate local conformations as accurately as the fragment assembly method in Robetta [23]. However, these HMM models do not consider conformation-dependency among more than two residues. It is also very difficult for these HMM models to make use of enriched sequence information such as PSI-BLAST sequence profile or threading-generated restraints to further improve sampling efficiency. Furthermore, it has not been studied if these HMM models can be applied to true ab initio folding. Recently, we have proposed a protein conformation sampling algorithm based on the first-order conditional random fields model [24] and directional statistics. The CRF model is a generalization of the HMM model and is much more expressive. Various sequence and structure features can be easily incorporated to the CRF model so that the probability of a conformation can be more accurately estimated. Our experimental results indicate that using the first-order CRF model, we can sample conformations with better quality than the FB5-HMM model [24]. All these studies have demonstrated that it is promising to probabilistically sample protein conformations from a continuous space. However, there is no ab initio folding program based upon probabilistic sampling of a continuous space that performs as well as the fragment assembly method and the lattice model.

This paper studies if probabilistic conformation sampling in a continuous space can be used to ab initio folding. In particular, we propose a second-order CRF model for ab initio protein structure prediction. The second-order model captures the conformation-dependency among three adjacent residues instead of only between two adjacent residues. In addition, the second-order model also considers the dependency of the local conformation transition on sequence information. The second-order model has millions of model parameters while the first-order model has only one hundred-thousand model parameters. Therefore, the second-order model is much more expressive in describing the complex protein sequence-structure relationship. Using this new CRF model and directional statistics, we can explore a continuous conformation space very efficiently in a probabilistic way. The probability of a conformation is estimated from PSI-BLAST sequence profile and PISPRED-predicted secondary structure. Furthermore, we test the structure prediction capability of this new graphical model by guiding conformation search using a simple energy function consisting of only three items: DOPE [25] (a distance-dependent pairwise statistical potential), KMBhbond hydrogen bonding energy [26] and ESP (a simplified solvent potential). Experimental results indicate that although using a simple energy function, this new ab initio folding method compares favorably with the fragment assembly program Robetta [19] and the lattice model program TOUCHSTONE II [16].

## 2 Methods

### 2.1 Continuous Representation of Protein Conformations

It is time-consuming to evaluate a full-atom energy function, but a residue-level energy function usually is not as accurate as an atom-level energy function. In this paper, we use a simplified and continuous representation of a protein model. In particular, we only consider the main chain and  $C_\beta$  atoms in folding simulation.

*C<sub>α</sub>-trace Representation.* Since the virtual bond length between two adjacent C<sub>α</sub> atoms can be approximated as a constant (i.e., 3.8Å)<sup>1</sup>, we can represent the C<sub>α</sub>-trace of a protein using a set of pseudo backbone angles ( $\theta, \tau$ ) [27]. Given a residue at position  $i$ , its corresponding  $\theta$  is defined as the pseudo bond angle formed by the C<sub>α</sub> atoms at positions  $i-1, i$  and  $i+1$ ;  $\tau$  is a pseudo dihedral angle around virtual bond between  $i-1$  and  $i$  and can be calculated from the coordinates of the C<sub>α</sub> atoms at positions  $i-2, i-1, i$  and  $i+1$ . Given the coordinates of the C<sub>α</sub> atoms at positions  $i-2, i-1$ , and  $i$ , the coordinates of the C<sub>α</sub> atom at position  $i+1$  can be calculated from ( $\theta, \tau$ ) at position  $i$ . Therefore, given the first three C<sub>α</sub> positions and  $N-2$  pairs of ( $\theta, \tau$ ), we can build the C<sub>α</sub> trace of a protein with  $N$  residues. The relative positions of the first three C<sub>α</sub> atoms are determined by the  $\theta$  angle at the second residue.

*Distribution of Bond Angles.* The preferred conformations of an amino acid in the protein backbone can be described as a probabilistic distribution of the  $\theta$  and  $\tau$  angles. Each ( $\theta, \tau$ ) corresponds to a unit vector in the three-dimensional space (i.e., a point on a unit sphere surface). We can use the 5-parameter Fisher-Bingham (FB5) distribution [28][22] to model the probability distributions over unit vectors. FB5 is the analogue on the unit sphere of the bivariate normal distribution with an unconstrained covariance matrix. The ( $\theta, \tau$ )-space is clustered into one hundred groups, each of which is described by an FB5 distribution. We calculate the ( $\theta, \tau$ ) distribution for each group from a set of approximately 3000 proteins with high-resolution X-ray structures using KentEstimator [22]. Any two proteins in this protein set share no more than 25% sequence identity. Please refer to Section 3 of [24] for a detailed description on how these distribution parameters are calculated. Therefore, given a distribution of ( $\theta, \tau$ ) at one residue, we can easily sample a pair of real-valued ( $\theta, \tau$ ) angles.

*Building Backbone Atoms.* Using ( $\theta, \tau$ ) representation, only the coordinates of the C<sub>α</sub> atoms can be built. To use an atom-level energy function, we also need to build the coordinates of other atoms. Given a C<sub>α</sub> trace, there are many methods that can build the coordinates for the main chain and C<sub>β</sub> atoms [29][30][31]. For protein folding simulation, we want a method that is both accurate and efficient. We choose to use a method similar to BBQ [30]. The original BBQ method can only build coordinates for the backbone N, C, and O atoms. We extend the method to build coordinates for the C<sub>β</sub> atom. Experimental results (data not shown in this paper) indicate that RMSD of this method is approximately 0.5Å supposing the native C<sub>α</sub>-trace is available. This level of accuracy is good enough for our folding simulation.

To employ the KMB hydrogen-bonding energy [26] for β-containing proteins, we also need to build the backbone hydrogen atoms. We use a quick and dirty method to build coordinates for the hydrogen atom HN [32]. Let  $N_i$  denote the position of the main chain N atom in the same residue as the HN atom. Let  $N_iC_{i-1}$  denote the normalized bond vector from the N atom to the C atom in the previous residue. Let  $N_iC_\alpha$  denote the normalized bond vector from the N atom to the C<sub>α</sub> atom in the same residue. Then the position of the hydrogen atom HN can be estimated by  $N_i - \frac{N_iC_{i-1} + N_iC_\alpha}{|N_iC_{i-1} + N_iC_\alpha|}$ .

---

<sup>1</sup> One rare exception is when the second residue is *cis* proline, the virtual bond length is approximately 3.2Å.

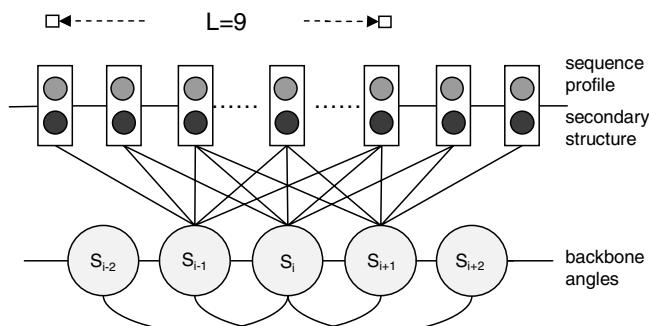
The average RMSD of this method is approximately 0.2Å (data not shown) supposing the native coordinates of other main chain atoms are available.

## 2.2 A Second-Order CRF Model for Protein Sequence-Structure Relationship

In [24], we have described a first-order CRF model for protein conformation sampling. In this paper, we extend our first-order CRF model to a second-order model to more accurately capture sequence-structure relationship. Experimental results (see Section 3.1) indicate that the second-order model is much more effective in conformation sampling than the first-order model. Please refer to [24] for a detailed description of the first-order CRF model. Here we only briefly explain our extension to the second-order CRF model.

In the context of conditional random fields, the primary sequence (or PSI-BLAST sequence profile) and predicted secondary structure are viewed as observations; the backbone pseudo angles and their distributions are treated as hidden states or labels. Let  $X$  denote the PSIPRED-predicted secondary structure likelihood scores. It is a matrix with  $3 \times N$  elements (where  $N$  is the number of residues in a protein), each of which is the predicted likelihood of one secondary structure type at a specific position. Let  $X_i$  denote the predicted likelihood of three secondary structure types at position  $i$ .  $X_i$  is a vector of three elements and  $X_i(x)$  is the predicted likelihood of secondary structure type  $x$  at position  $i$ . Let  $M$  denote the PSI-BLAST sequence profile, which is a position-specific frequency matrix containing  $20 \times N$  entries. Each element in this matrix is the occurring frequency of one amino acid at a given position. Let  $M_i$ , a vector of 20 elements, denote the sequence profile at position  $i$ . Let  $M_i(aa)$  denote the occurring frequency of amino acid  $aa$  at position  $i$ . Let  $H = \{h_1, h_2, \dots, h_{100}\}$  denote the set of one hundred backbone angle states, each of which represents an FB5 distribution. Let  $S = \{s_1, s_2, \dots, s_N\}$  ( $s_i \in H$ ) be a sequence of labels (i.e., FB5 distribution) corresponding to the observations  $X$  and  $M$ .

As shown in Figure 1, we use a second-order CRF model to capture the complex relationship between the state sequence  $S$  and the observations  $X$  and  $M$ . Our CRF model defines the conditional probability of  $S$  given  $M$  and  $X$  as follows.



**Fig. 1.** The second-order CRF model used to describe protein sequence-structure relationship. Each  $S_i$  represents the angle distribution at position  $i$ .

$$P_\theta(S|M, X) = \exp\left(\sum_{i=1}^N F(S, M, X, i)\right)/Z(M, X) \quad (1)$$

where  $\theta=(\lambda_1, \lambda_2, \dots, \lambda_p)$  is the model parameter and  $Z(M, X) = \sum_S \exp(\sum_{i=1}^N F(S, M, X, i))$  is a normalization factor summing over all the possible label sequences for the given observations.  $F(S, M, X, i)$  consists of two edge features and two label features at position  $i$ . It is given by

$$\begin{aligned} F(S, M, X, i) &= e_1(s_{i-1}, s_i) + e_2(s_{i-1}, s_i, s_{i+1}) \\ &+ \sum_{j=i-w}^{i+w} v_1(s_i, M_j, X_j) + \sum_{j=i-w}^{i+w} v_2(s_{i-1}, s_i, M_j, X_j) \end{aligned} \quad (2)$$

Meanwhile,  $e_1(s_{i-1}, s_i)$  and  $e_2(s_{i-1}, s_i, s_{i+1})$  are the first-order and second-order edge feature functions, respectively. and  $v_1(s_i, M_j, X_j)$  and  $v_2(s_{i-1}, s_i, M_j, X_j)$  are two label feature functions.

The two edge feature functions are given by

$$e_1(s_{i-1}, s_i) = \lambda(h', h'')[s_{i-1} == h'][s_i == h''] \quad (3)$$

$$e_2(s_{i-1}, s_i, s_{i+1}) = \lambda(h', h'', h''') [s_{i-1} == h'][s_i == h''][s_{i+1} == h'''] \quad (4)$$

In the above equations, the indicator function  $[s_i == h]$  is equal to 1 if the state at position  $i$  is  $h \in H$ , otherwise 0. The edge feature functions are independent of the observations. They capture the conformation-dependency of two or three adjacent residues. The model parameters (i.e.,  $\lambda$ ) in Eqs. 3 and 4 are identified by the states at two or three adjacent residues, respectively.

The two label feature functions are given by

$$\begin{aligned} v_1(s_i, M_j, X_j) &= \sum_s \sum_{aa} \lambda(j - i, s, aa, h) X_j(s) M_j(aa) [s_i == h] \\ &+ \sum_s \lambda(j - i, s, h) X_j(s) [s_i == h] \\ &+ \sum_{aa} \lambda(j - i, aa, h) M_j(aa) [s_i == h] \end{aligned} \quad (5)$$

$$\begin{aligned} v_2(s_{i-1}, s_i, M_j, X_j) &= \sum_s \sum_{aa} \lambda(j - i, s, aa, h', h'') X_j(s) M_j(aa) [s_{i-1} == h'][s_i == h''] \\ &+ \sum_s \lambda(j - i, s, h', h'') X_j(s) [s_{i-1} == h'][s_i == h''] \\ &+ \sum_{aa} \lambda(j - i, aa, h', h'') M_j(aa) [s_{i-1} == h'][s_i == h''] \end{aligned} \quad (6)$$

Label feature functions model the dependency of backbone angles on protein sequence and secondary structure. Eqs. 5 and 6 define the first-order and second-order label feature functions, respectively. These two equations indicate that not only the

state (i.e., angle) itself but also the state transition depend on the sequence profile and secondary structure. The label feature functions also model the interactions between secondary structure and primary sequence, as shown in the first items of the right hand side of Equations 5 and 6. According to the third and fourth items in the right hand side of Eq. 2, the state (or state transition) at one position depends on the sequence profile and secondary structure information in a window of width  $2w+1$ . In our implementation, we set the half window width  $w$  to 4. The model parameters for label features are identified by one or two states, secondary structure type, amino acid identity and the relative position of the observations.

The second-order CRF model has millions of features, each of which has a model parameter to be trained. We train the model parameters by maximizing  $P_\theta(S|M, X)$  on a set of 3000 proteins chosen by the PISCES server [33]. Please refer to [24] for a detailed description of how the CRF model can be trained and how the model parameters are chosen using cross-validation. The second-order CRF model is much more expressive than the first-order one and can describe the protein sequence-structure relationship more accurately. We revised the FlexCRFs program [34] to train our CRF model and it takes approximately 24 hours to train a single model on a cluster of 150 CPUs.

Once the CRF model is trained, we can efficiently estimate the probability of a protein conformation, which can be used to sample a protein conformation or resample the local conformation of a protein segment. First we probabilistically sample labels (i.e., angle distribution) from the CRF model and then sample real-valued angles from the labels. Please refer to [24] for more details. The conformation sampling algorithm in [24] is based on the first-order model, but it can be easily extended to the second-order model.

## 2.3 Energy Function

The energy function we used for protein folding simulation consists of three items: DOPE, KMBhbond and ESP. DOPE is a full-atom, distance-dependent pairwise statistical potential designed by Shen and Sali [25]. DOPE performs as well or better than many other statistical potentials and force fields in differentiating a native structure from decoys [25][35]. The statistical potential in DOPE distinguishes the amino acid identity and atomic identity of two interacting particles. In our folding simulation, we only build coordinates for main chain and  $C_\beta$  atoms, so only the statistical potentials related to main-chain and  $C_\beta$  atoms are used to calculate the energy of a conformation. We denote this revised DOPE as DOPE- $C_\beta$ . According to [35], DOPE- $C_\beta$  is highly correlated with the full-atom DOPE. DOPE- $C_\beta$  also performs favorably in applications to intra-basin protein folding [36].

KMBhbond is a statistical potential for hydrogen bonding developed by Baker's group [26]. It depends on the distance between the geometric centers of the N-H bond vector and the C=O bond vector, the bond angle between the N-H bond vector and the hydrogen bond, the bond angle between the C=O bond vector and the hydrogen bond, and the dihedral angle about the acceptor-acceptor base bond. The three angles describe the relative orientation of the bond vectors in the hydrogen bond.

ESP is an approximation to the Ooi-Scheraga solvent-accessible surface area (SASA) potential [37]. Since our conformation representation does not contain side-chain atoms, which are necessary for the calculation of the solvent-accessible surface area potential,

we employ a simple ESP that assigns each residue with an environmental energy score. ESP is a function of the protein size and the number of  $C_\alpha$  atoms contained within an 8.5Å sphere centered on the residue's  $C_\alpha$  atom [38]. Explicitly, the ESP statistical potential has the form given by

$$ESP(aa, n) = -\ln \frac{P(n|R, aa)}{p(n|R)} \quad (7)$$

where  $n$  is the number of  $C_\alpha$  atoms in an 8.5Å sphere centered on the residue's  $C_\alpha$  atom,  $R$  is the radius of gyration of the protein,  $aa$  is the amino acid identity of the residue,  $p(n|R)$  is the number of  $C_\alpha$  atoms in an 8.5Å sphere for a given protein radius regardless of amino acid identity, and  $p(n|R, aa)$  is the number of  $C_\alpha$  atoms in an 8.5Å sphere for a given protein radius and amino acid identity. We calculate  $ESP(aa, n)$  from a set of 3000 non-redundant experimental structures chosen by the PISCES server [33]. Each protein in this set has resolution at least 2.0Å,  $R$  factor no bigger than 0.25 and at least 30 residues. Any two proteins in this set share no more than 30% sequence identity.

The weight factors combining these three energy items are trained on the proteins in Table II using grid search in a progressive way. First, we fix the weight factor of DOPE to 1 and determine the weight factor for ESP by minimizing the average RMSDs of generated decoys. Then we fix the weight factors of both DOPE and ESP and determine the weight factor for KMBhbond using the same way.

## 2.4 Energy Minimization

We employ a simulated annealing (SA) algorithm to minimize the energy function for a given protein. The SA routine is based on the algorithm described by Aarts and Korst [39]. We start with sampling an initial conformation and then search for a better one by minimizing the energy function. Given a conformation, we propose a new conformation by resampling the local conformation of a small segment using the CRF model (see [24]). The new conformation is rejected if there are serious steric clashes among atoms<sup>2</sup>, otherwise we calculate its energy. If the new conformation has an energy value smaller than the previous one, then we accept this conformation, otherwise we accept it by a probability  $e^{-\frac{\Delta E}{t}}$  where  $\Delta E$  is the energy increment and  $t$  is the annealing temperature.

The initial annealing temperature is chosen so that at the beginning of the annealing process an energy increase is accepted with a given probability  $p_0$  ( $=0.8$ ). The initial temperature  $t_0$  is determined by

$$t_0 = -\Delta E / \ln(p_0) \quad (8)$$

Where  $\Delta E$  is the average energy increase. Given a protein, we conduct a series of trial conformation samplings and accept all the generated conformations. Then we estimate  $\Delta E$  by calculating the average energy increase observed in our trial samplings. During the folding simulation process, we decrease the annealing temperature gradually using an exponential cooling schedule. The temperature is updated as follows.

---

<sup>2</sup> There is a steric clash if the distance between two  $C_\alpha$  atoms is less than 4Å.

$$t_{k+1} = \alpha t_k \quad (9)$$

Where  $\alpha$  is set to 0.9.

At each annealing temperature, the number of sampled conformations is set to  $100 \times (1 + N/100)$  where  $N$  is the number of residues in the protein. This number is set to achieve thermal equilibrium. The termination of the SA process is triggered when any of the following two conditions is satisfied: 1) either the temperature is low enough such that almost no energy increase is accepted and the annealing process is trapped at a local minima; 2) or the number of conformations generated in a single simulation process reaches a threshold (say 10,000).

## 3 Results

### 3.1 Comparison with the First-Order Model

First, we compare our second-order CRF model with the first-order model described in [24] to see how much improvement we can achieve by considering the interdependency among three adjacent residues. In this comparison, we guide conformation search using only compactness and self-avoiding constraints but not energy function. We generated 20,000 decoys for each test protein using these two CRF models and then calculated the average RMSDs of the top 1%, 2%, 5% and 10% decoys, respectively. Table I shows the quality of the decoys generated by the first-order and second-order models on a variety of test proteins. In terms of the best decoys, the second-order model is better on 13 out of 22 test proteins and worse on seven proteins. The best decoys may be generated by chance, so they cannot be reliably used to evaluate the performance of two CRF models. We further examine the performance of these two CRF models by comparing the average RMSDs of their top decoys. The second-order model outperforms the first-order model on almost all the test proteins except 1aa2 and 4icb. Both CRF models have similar performance on 1aa2 maybe because that 1aa2 has a large conformation search space so that neither CRF model can search the space efficiently. The reason that the second-order model performs worse on 4icb is because there is a *cis* proline in this protein, and the length of the virtual  $C_\alpha$ -bond ending at this proline is approximately 3.2Å instead of our assumption 3.8Å. Therefore, the more accurately can the model predict the backbone angles, the more the decoys deviate from the native. This problem will be addressed later since we can predict if a residue is a *cis* proline or not with accuracy 92%.

### 3.2 Comparison with TOUCHSTONE II

By combining the energy functions described in Section 2.3 and our second-order CRF model, we build a program, denoted as CRFFolder, for ab initio protein structure prediction. We compare CRFFolder with TOUCHSTONE II, a representative lattice-model-based ab initio protein structure prediction program developed by Skolnick *et al.* TOUCHSTONE II is an excellent ab initio folding program and its two derivatives TASSER [3] and I-TASSER [1] perform very well in both CASP7 and CASP8. We do not compare CRFFolder with the two derivatives because both TASSER and I-TASSER use

**Table 1.** Decoy quality comparison between the first-order and second-order CRF conformation samplers. Columns 1-3 list the PDB code, protein size and the type of the test proteins. Columns “best” list the RMSDs of the best decoys; the other columns list the average RMSDs of the top decoys. “O-1” and “O-2” denote the first-order and the second-order CRF models, respectively. In total 20,000 decoys are generated for each protein without using energy function.

PDB code	Length	Class	best		1%		2%		5%		10%	
			O-1	O-2	O-1	O-2	O-1	O-2	O-1	O-2	O-1	O-2
1aa2	108	$\alpha$	7.3	7.3	9.3	9.5	9.8	9.9	10.4	10.4	10.9	10.8
1beo	98	$\alpha$	6.4	5.8	8.4	8.1	8.8	8.6	9.5	9.3	10.1	9.9
1ctfA	68	$\alpha\beta$	3.7	3.7	5.1	4.6	5.4	4.9	5.9	5.3	6.5	5.7
1dktA	72	$\beta$	6.1	5.1	7.6	6.4	8.0	6.7	8.5	7.2	9.0	7.7
1enhA	54	$\alpha$	2.3	2.2	3.1	2.6	3.3	2.6	3.7	2.8	4.1	2.9
1fc2C	43	$\alpha$	1.9	2.3	2.7	2.6	2.8	2.7	3.1	2.9	3.4	3.0
1fca	55	$\beta$	4.9	5.0	6.4	6.2	6.7	6.5	7.3	6.9	7.7	7.2
1fgp	67	$\beta$	7.4	5.9	8.9	7.8	9.2	8.1	9.6	8.6	10.0	8.9
1jer	110	$\beta$	9.6	10.2	11.6	11.5	11.9	11.8	12.4	12.3	12.9	12.7
1nkl	78	$\alpha$	3.6	3.1	4.7	3.8	5.0	3.9	5.4	4.3	5.8	4.6
1pgb	56	$\alpha\beta$	3.1	2.6	4.1	3.6	4.3	3.8	4.6	4.0	4.9	4.2
1sro	76	$\beta$	6.2	5.4	7.8	7.2	8.2	7.6	8.8	8.2	9.3	8.8
1trIA	62	$\alpha$	3.5	3.7	4.4	4.5	4.6	4.6	4.9	4.8	5.2	5.0
2croA	65	$\alpha$	2.8	2.6	3.6	3.2	3.8	3.4	4.2	3.6	4.6	3.9
2gb1A	56	$\beta$	2.9	2.0	4.0	3.5	4.2	3.6	4.6	3.9	4.9	4.1
4icbA	76	$\alpha$	4.6	4.4	5.9	6.7	6.2	7.1	6.8	7.7	7.4	8.2
T052	98	$\beta$	7.6	8.4	10.6	10.1	11.0	10.5	11.6	11.1	12.1	11.6
T056	114	$\alpha$	7.8	7.6	9.8	9.6	10.3	9.9	10.9	10.5	11.5	11.2
T059	71	$\beta$	6.3	6.2	8.5	8.0	8.8	8.2	9.3	8.7	9.6	9.0
T061	76	$\alpha$	5.3	6.0	7.0	7.1	7.3	7.3	7.8	7.6	8.2	7.9
T064	103	$\alpha$	7.2	7.5	9.4	9.0	9.9	9.5	10.7	10.4	11.4	11.1
T074	98	$\alpha$	4.9	4.2	7.3	6.6	7.7	7.0	8.4	7.6	9.0	8.1
average RMSD			5.25	<b>5.05</b>	6.8	<b>6.4</b>	7.2	<b>6.7</b>	7.7	<b>7.2</b>	8.1	<b>7.6</b>

threading-generated constraints to guide conformation search while CRFFolder does not. Due to the limitations of computational power, we tested CRFFolder on only 15 test proteins, which were also tested by TOUCHSTONE II. These test proteins have very different secondary structures and protein sizes ranging from 47 to 157. We generated approximately 3000 decoys for each alpha protein, 7000 decoys for each alpha-beta protein, and 10,000 decoys for each beta protein. By contrast, TOUCHSTONE II used a complex energy function consisting of 21 items and generated 24,000 decoys for each test protein [16]. As shown in Table 2, CRFFolder performs much better than TOUCHSTONE II on all the alpha proteins except one. CRFFolder also has comparable performance on beta and alpha-beta proteins. Looks like that on larger test proteins, CRFFolder is slightly worse than TOUCHSTONE II. This may be because that the replica exchange Monte Carlo algorithm used by TOUCHSTONE II for energy minimization is better than the simulated annealing algorithm used in CRFFolder. Note that since two programs use very different clustering methods, it is not easy to compare these two programs fairly. TOUCHSTONE II used a program SCAR [16] to do decoy clustering while we use

**Table 2.** Decoy quality comparison between CRFFolder and TOUCHSTONE II. Columns 1-3 list the PDB code, length and the type of the test proteins. The two “Best Cluster” columns list the RMSDs of the representative decoys of the best clusters. In these two columns, the first number in parentheses denotes the rank of the best cluster and the second number is the total number of clusters. Column “best” lists the RMSDs of the best decoys. Columns “1%” and “2%” list the average RMSDs of the top 1% and 2% decoys, respectively. The results of TOUCHSTONE II are from [16].

PDB code	Length	Class	TOUCHSTONE II	CRFFolder			
			Best Cluster	Best Cluster	Best	1%	2%
1bw6A	56	$\alpha$	4.79(2/3)	<b>3.82(3/3)</b>	2.75	3.38	3.54
1lea	72	$\alpha$	5.69(5/5)	<b>4.10(5/7)</b>	3.41	4.19	4.48
2af8	86	$\alpha$	11.07(5/6)	<b>8.9(12/19)</b>	7.07	8.53	8.97
256bA	106	$\alpha$	3.61(2/3)	<b>2.75(6/11)</b>	2.50	3.45	3.70
1sra	151	$\alpha$	10.71(3/12)	13.95(17/25)	10.82	13.76	14.24
1gpt	47	$\alpha\beta$	6.30(1/25)	<b>5.55(42/67)</b>	4.34	5.20	5.47
1kp6A	79	$\alpha\beta$	10.01(8/14)	<b>7.99(1/7)</b>	6.29	7.51	7.81
1poh	85	$\alpha\beta$	9.10(5/9)	<b>8.84(5/10)</b>	7.49	8.70	9.04
1npsA	88	$\alpha\beta$	6.89(33/34)	9.91(41/57)	7.87	9.19	9.66
1t1dA	100	$\alpha\beta$	8.96(7/13)	9.22(10/13)	6.51	9.51	9.94
1msi	66	$\beta$	7.72(19/28)	7.77(12/15)	6.24	7.55	7.89
1hoe	74	$\beta$	9.39(5/13)	9.87(16/35)	7.96	10.00	10.37
1ezgA	82	$\beta$	11.03(40/44)	<b>10.42(42/66)</b>	9.66	10.35	10.62
1sfp	111	$\beta$	7.48(2/18)	11.07(5/11)	9.32	11.09	11.59
1b2pA	119	$\beta$	12.52(31/56)	<b>10.01(18/25)</b>	8.76	10.89	11.32

MaxCluster<sup>3</sup>. For the purpose of comparison, we also show the RMSD of the best decoys and the average RMSDs of the top 1% and 2% decoys generated by CRFFolder.

### 3.3 Comparison with Robetta in CASP8

In this section, we compare the performance of our method CRFFolder with Baker’s Robetta on some hard CASP8 targets. CASP8 was held during the summer of 2008 and these hard targets have no good homologs in the PDB. Note that our second-order CRF model was trained before CASP8 started and the energy function was gradually improved during CASP8. We use the hard targets on which both Robetta and CRFFolder did ab initio folding. Robetta is a well-known fragment assembly method for ab initio protein structure prediction. The top five models generated by Robetta *ab initio method* for each hard target are available at the Robetta web site.<sup>4</sup> Using CRFFolder, we generated 7000 decoys for each target and then chose top five models. Note that the top models chosen by CRFFolder are not exactly same as our CASP8 submissions since we also submitted threading-generated models for some hard targets.

Table 3 compares CRFFolder and Robetta in terms of the quality of the first-ranked structure models. The model quality is evaluated by a program TM-score [40], which

<sup>3</sup> <http://www.sbg.bio.ic.ac.uk/~maxcluster/>

<sup>4</sup> <http://robbetta.bakerlab.org/queue.jsp?UserName=casp8&rpp=100>.

**Table 3.** Decoy quality comparison between CRFFolder and Robetta. The “Robetta” column lists TM-scores of the first-ranked decoys generated by Baker’s Robetta. The “CRFFolder” column lists TM-scores of the first-ranked decoys generated by CRFFolder.

Target ID	Length	Class	Robetta	CRFFolder
T0397_1	70	$\alpha/\beta$	0.25	<b>0.258</b>
T0460	111	$\alpha/\beta$	0.262	<b>0.308</b>
T0465	157	$\alpha/\beta$	0.243	<b>0.253</b>
T0466	128	$\beta$	0.326	0.217
T0467	97	$\beta$	0.303	<b>0.364</b>
T0468	109	$\alpha/\beta$	0.253	<b>0.308</b>
T0476	108	$\alpha/\beta$	0.279	0.250
T0480	55	$\beta$	0.208	<b>0.307</b>
T0482	120	$\alpha/\beta$	0.352	0.223
T0484	62	$\alpha$	0.253	0.249
T0495_2	65	$\alpha/\beta$	0.312	<b>0.436</b>
T0496_1	110	$\alpha/\beta$	0.235	<b>0.293</b>
T0496_2	68	$\alpha$	0.291	<b>0.500</b>
T0510_3	43	$\alpha/\beta$	0.147	<b>0.352</b>
T0513_1	77	$\alpha/\beta$	0.581	0.367
T0514	145	$\alpha/\beta$	0.283	0.277
Average			0.286	<b>0.310</b>
Sum			4.578	<b>4.960</b>

generates a number between 0 and 1 to indicate the quality of a structure model. Roughly speaking, the higher the TM-score, the better the model quality. As shown in this table, on average CRFFolder is better than Robetta by approximately 0.024 (i.e., 8%). On a variety of proteins such as T0467, T0468, T0480, T0495\_2, T0496\_1, T0496\_2 and T0510\_3, CRFFolder is much better than Robetta. By contrast, Robetta is also much better than CRFFolder on some beta-containing proteins such as T0466, T0482, and T0513\_1. In particular, Robetta did very well on T0513\_1, an alpha-beta protein with two beta sheets. This may indicate that Robetta is better in sampling conformations for large beta sheets or has a better hydrogen-bonding energy item for the formation of large beta sheets. Please note that the TM-scores of T0397\_1, T0495\_2, T0496\_1, T0496\_2 and T0513\_1 are different from those on Zhang’s CASP8 assessment page because of different domain definitions. The domain definition of T0510\_3 is from Zhang’s CASP8 assessment page<sup>5</sup> while the domain definitions of the others are from Robetta’s CASP8 web site.

## 4 Conclusion

This paper presented a probabilistic graphical model for ab initio protein structure prediction. By using the second-order CRF model and directional statistics, we can accurately describe the protein sequence-structure relationship and search a continuous protein conformation space very efficiently. Although still at its infancy stage and using a simple energy function, our method can do ab initio protein structure prediction

<sup>5</sup> <http://zhang.bioinformatics.ku.edu/casp8/index.html>

as well as two well-developed ab initio folding programs TOUCHSTONE-II and Robetta. These two programs have been developed for many years and have well-tuned and sophisticated energy functions. Our method is much better than TOUCHSTONE II on alpha proteins and has similar accuracy on beta-containing proteins with TOUCHSTONE II. Our method is also better than Robetta on quite a few CASP8 test proteins but worse than Robetta on some beta-containing proteins. To improve the performance of our method on beta-containing proteins, we will further improve our conformation sampling algorithm on beta regions and develop a better hydrogen-bonding energy item for the formation of beta sheets.

## Acknowledgements

This work is supported by the internal research funding of TTI-C and NIH research grant. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:[www.sharcnet.ca](http://www.sharcnet.ca)), the Open Science Grid Engagement VO and the University of Chicago Computation Institute. The authors are also grateful to Dr. Ian Foster, Dr. John McGee and Mats Rynge for their help with computational resources.

## References

1. Wu, S., Skolnick, J., Zhang, Y.: Ab initio modeling of small proteins by iterative TASSER simulations. *BMC Biology* 5, 17+ (2007)
2. Misura, K.M., Chivian, D., Rohl, C.A., Kim, D.E., Baker, D.: Physically realistic homology models built with ROSETTA can be more accurate than their templates. *Proceedings of National Academy Sciences* 103(14), 5361–5366 (2006)
3. Zhang, Y., Skolnick, J.: The protein structure prediction problem could be solved using the current PDB library. *Proceedings of National Academy Sciences, USA* 102(4), 1029–1034 (2005)
4. Moult, J., Fidelis, K., Rost, B., Hubbard, T., Tramontano, A.: Critical assessment of methods of protein structure prediction (CASP)–round 6. *Proteins: Structure, Function and Bioinformatics* 61(suppl. 7), 3–7 (2005)
5. Moult, J., Fidelis, K., Kryshtafovych, A., Rost, B., Hubbard, T., Tramontano, A.: Critical assessment of methods of protein structure prediction–Round VII. *Proteins: Structure, Function, and Bioinformatics* 69(S8), 3–9 (2007)
6. Jones, T.A., Thirup, S.: Using known substructures in protein model building and crystallography. *EMBO Journal* 5, 819–823 (1986)
7. Claessens, M., van Cutsem, E., Lasters, I., Wodak, S.: Modelling the polypeptide backbone with spare parts from known protein structures. *Protein Engineering* 2(5), 335–345 (1989)
8. Unger, R., Harel, D., Wherland, S., Sussman, J.L.: A 3D building blocks approach to analyzing and predicting structure of proteins. *Proteins: Structure, Function and Genetics* 5(4), 355–373 (1989)
9. Simon, I., Glasser, L., Scheraga, H.A.: Calculation of Protein Conformation as an Assembly of Stable Overlapping Segments: Application to Bovine Pancreatic Trypsin Inhibitor. *Proceedings of National Academy Sciences, USA* 88(9), 3661–3665 (1991)
10. Levitt, M.: Accurate modeling of protein conformation by automatic segment matching. *Journal of Molecular Biology* 226(2), 507–533 (1992)

11. Sippl, M.: Recognition of errors in three-dimensional structures of proteins. *Proteins: Structure, Function, and Bioinformatics* 17, 355–362 (1993)
12. Wendoloski, J.J., Salemme, F.R.: PROBIT: a statistical approach to modeling proteins from partial coordinate data using substructure libraries. *Journal of Molecular Graphics* 10(2), 124–126 (1992)
13. Bowie, J.U., Eisenberg, D.: An Evolutionary Approach to Folding Small  $\alpha$ -Helical Proteins that Uses Sequence Information and an Empirical Guiding Fitness Function. *Proceedings of National Academy Sciences, USA* 91(10), 4436–4440 (1994)
14. Xia, Y., Huang, E.S., Levitt, M., Samudrala, R.: Ab initio construction of protein tertiary structures using a hierarchical approach. *Journal of Molecular Biology* 300(1), 171–185 (2000)
15. Kihara, D., Lu, H., Kolinski, A., Skolnick, J.: TOUCHSTONE: An ab initio protein structure prediction method that uses threading-based tertiary restraints. *Proceedings of the National Academy of Sciences* 98(18), 10125–10130 (2001)
16. Zhang, Y., Kolinski, A., Skolnick, J.: TOUCHSTONE II: a new approach to ab initio protein structure prediction. *Biophysical Journal* 85(2), 1145–1164 (2003)
17. Moult, J., Fidelis, K., Zemla, A., Hubbard, T.: Critical assessment of methods of protein structure prediction (CASP)-round V. *Proteins: Structure, Function, and Genetics* 53(S6), 334–339 (2003)
18. Moult, J.: A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Current Opinion in Structure Biology* (June 2005)
19. Simons, K.T., Kooperberg, C., Huang, E., Baker, D.: Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *Journal of Molecular Biology* 268(1), 209–225 (1997)
20. Li, S.C., Bu, D., Xu, J., Li, M.: Fragment-hmm: A new approach to protein structure prediction. *Protein Science*, ps.036442.108+ (August 2008)
21. Feldman, H.J., Hogue, C.W.V.: Probabilistic sampling of protein conformations: New hope for brute force? *Proteins: Structure, Function, and Genetics* 46(1), 8–23 (2002)
22. Hamelryck, T., Kent, J.T.T., Krogh, A.: Sampling Realistic Protein Conformations Using Local Structural Bias. *PLoS Comput Biology* 2(9) (September 2006)
23. Boomsma, W., Mardia, K.V., Taylor, C.C., Ferkinghoff-Borg, J., Krogh, A., Hamelryck, T.: A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences* 105(26), 8932–8937 (2008)
24. Zhao, F., Li, S., Sterner, B.W., Xu, J.: Discriminative learning for protein conformation sampling. *Proteins: Structure, Function, and Bioinformatics* 73(1), 228–240 (2008)
25. Shen, M.Y., Sali, A.: Statistical potential for assessment and prediction of protein structures. *Protein Science* 15(11), 2507–2524 (2006)
26. Morozov, A.V., Kortemme, T., Tsemekhman, K., Baker, D.: Close agreement between the orientation dependence of hydrogen bonds observed in protein structures and quantum mechanical calculations. *Proceedings of National Academy Sciences* 101(18), 6946–6951 (2004)
27. Levitt, M.: A simplified representation of protein conformations for rapid simulation of protein folding. *Journal of Molecular Biology* 104, 59–107 (1976)
28. Kent, J.T.: The Fisher-Bingham Distribution on the Sphere. *Journal of Royal Statistical Society* 44, 71–80 (1982)
29. Holm, L., Sander, C.: Database algorithm for generating protein backbone and side-chain co-ordinates from a C alpha trace application to model building and detection of co-ordinate errors. *Journal of Molecular Biology* 218(1), 183–194 (1991)
30. Gront, D., Kmiecik, S., Kolinski, A.: Backbone building from quadrilaterals: A fast and accurate algorithm for protein backbone reconstruction from alpha carbon coordinates. *Journal of Computational Chemistry* 28(9), 1593–1597 (2007)

31. Maupetit, J., Gautier, R., Tufféry, P.: SABBAC: online structural alphabet-based protein backbone reconstruction from alpha-carbon trace. *Nucleic Acids Research* 34(Web-server issue) (July 2006)
32. Branden, C.-I., Tooze, J.: *Introduction to Protein Structure*, 2nd edn. Garland Publishing (1999)
33. Wang, G., Dunbrack, R.L.: PISCES: a protein sequence culling server. *Bioinformatics* 19(12), 1589–1591 (2003)
34. Phan, X.-H., Nguyen, L.-M., Nguyen, C.-T.: FlexCRFs: Flexible Conditional Random Field Toolkit (2005), <http://flexcrfs.sourceforge.net>
35. Fitzgerald, J.E., Jha, A.K., Colubri, A., Sosnick, T.R., Freed, K.F.: Reduced Cbeta statistical potentials can outperform all-atom potentials in decoy identification. *Protein Science* 16(10), 2123–2139 (2007)
36. Colubri, A., Jha, A.K., Shen, M.Y., Sali, A., Berry, R.S., Sosnick, T.R., Freed, K.F.: Minimalist representations and the importance of nearest neighbor effects in protein folding simulations. *Journal of Molecular Biology* 363(4), 835–857 (2006)
37. Ooi, T., Oobatake, M., Nemethy, G., Scheraga, H.A.: Accessible Surface Areas as a Measure of the Thermodynamic Parameters of Hydration of Peptides. *Proceedings of the National Academy of Sciences* 84(10), 3086–3090 (1987)
38. Fernández, A., Sosnick, T.R., Colubri, A.: Dynamics of hydrogen bond desolvation in protein folding. *Journal of molecular biology* 321(4), 659–675 (2002)
39. Aarts, E., Korst, J.: *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, Chichester (1991)
40. Zhang, Y., Skolnick, J.: TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research* 33(7), 2302–2309 (2005)

# Topology-Free Querying of Protein Interaction Networks

Sharon Bruckner<sup>1</sup>, Falk Hüffner<sup>1</sup>, Richard M. Karp<sup>2</sup>, Ron Shamir<sup>1</sup>,  
and Roded Sharan<sup>1</sup>

<sup>1</sup> School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel  
`{bruckner,hueffner,rshamir,roded}@tau.ac.il`

<sup>2</sup> International Computer Science Institute, 1947 Center St., Berkeley,  
CA 94704, USA  
`karp@icsi.berkeley.edu`

**Abstract.** In the network querying problem, one is given a protein complex or pathway of species  $A$  and a protein–protein interaction network of species  $B$ ; the goal is to identify subnetworks of  $B$  that are similar to the query. Existing approaches mostly depend on knowledge of the interaction topology of the query in the network of species  $A$ ; however, in practice, this topology is often not known. To combat this problem, we develop a topology-free querying algorithm, which we call TORQUE. Given a query, represented as a set of proteins, TORQUE seeks a matching set of proteins that are sequence-similar to the query proteins and span a connected region of the network, while allowing both insertions and deletions. The algorithm uses alternatively dynamic programming and integer linear programming for the search task. We test TORQUE with queries from yeast, fly, and human, where we compare it to the QNet topology-based approach, and with queries from less studied species, where only topology-free algorithms apply. TORQUE detects many more matches than QNet, while in both cases giving results that are highly functionally coherent.

## 1 Introduction

Sequence-based searches have revolutionized modern biology, serving to infer gene function, homology relations, protein structure, and more. In the last few years, there has been an effort to generalize these techniques to the network level. In a *network querying* problem, one is given a small subnetwork, corresponding to a pathway or a complex of interest. The goal is to identify similar instances in a large network, where similarity is measured in terms of sequence or interaction patterns.

The largest body of previous work on network querying concerns querying subnetworks across species. Kelley et al. [14] and later Shlomi et al. [29] devised fixed-parameter algorithms for querying linear paths within a protein–protein interaction (PPI) network. These algorithms were subsequently extended in the QNet software to allow searching for trees and bounded treewidth graphs [23].

A related work by Pinter et al. [21] presented a polynomial algorithm for detecting homeomorphic subtrees within a tree representing a collection of metabolic pathways. Another approach that relaxes the homomorphism but requires target and query nodes to agree in their neighborhood was given by Narayanan and Karp [18]. Sohler and Zimmer [30] developed a general framework for subnetwork querying, which is based on translating the problem to that of finding a clique in an appropriately defined graph. Due to its complexity, their method is applicable only to very small queries. Yang and Sze [35] examine both query paths and the general case, but since their method is based on exhaustive enumeration, it can also handle only small queries.

Another line of work on network querying has been the search for small motifs that are defined in terms of the functional attributes of their member proteins and the interactions among them. Lacroix et al. [16] suggested a branch-and-bound approach for finding connected subgraphs whose vertex set matches a query, which they applied to small queries (of size 2–4) only. Betzler et al. [4] gave a fixed parameter algorithm for the latter problem and some extensions of it. An additional heuristic solution was offered by Zheng et al. [37] to a similar problem, in the context of querying metabolic networks. Finally, Ferro et al. [9] presented the GraphFind algorithm, which utilizes fast heuristics for subgraph isomorphism to identify approximate matches of queries within a collection of networks.

A limitation of the approaches above (except for [16], [4]) is that they rely on precise information on the interaction pattern of the query pathway. However, often this information is missing. For example, hundreds of protein complexes have been reported in the literature for yeast [27], human [25], and other species. However, for most of these complexes no information exists on their interaction patterns [36], motivating a topology-free approach for the querying problem.

Here we devise TORQUE (TOpology-free netwoRk QUerying), a novel approach for network querying that does not rely on knowledge of the query topology. The input to our method is a set of proteins, representing a protein complex or pathway of interest and a network in which the search is to be conducted. The goal is to find matching sets of proteins that span connected regions in the network. The corresponding theoretical problem that we study is searching a colored graph for connected subgraphs whose vertices have distinct given colors. We provide fixed-parameter algorithms that are based on the color-coding paradigm [1] and dynamic programming (DP) for several variants of this problem. In addition, we provide an integer programming (ILP) formulation of it. That formulation includes a novel way to describe subgraph connectivity constraints, which can be useful in other problems as well. The methods can handle edge weights, insertions of network vertices (that do not match any query protein), and deletions of query nodes. By using DP and ILP approaches, we can query complexes of all sizes within current networks in reasonable time.

We applied TORQUE to query about 600 known complexes of size 4–25 from a variety of species in the PPI networks of yeast, fly and human. We tested our algorithm both on queries from species for which a PPI network is available,

where we compared it to the QNet [23] topology-based approach, and on queries from less studied species, where only topology-free algorithms apply. TORQUE detected many more matches than QNet, while in both cases giving results that are highly functionally coherent.

Due to space limitations, some proofs are omitted. They will be included in a full version of this manuscript.

## 2 Algorithms

Let  $G = (V, E)$  be a PPI network where vertices represent proteins and edges correspond to PPIs. Denote  $|V| = n$  and  $|E| = m$ . For a vertex  $v$ , let  $N(v)$  denote the set of its neighbors, i.e.,  $N(v) = \{u : (u, v) \in E\}$ . For two disjoint sets  $S_1$  and  $S_2$ , we write  $S_1 \uplus S_2$  for their union  $S_1 \cup S_2$ . We denote by  $G[K]$  the subgraph of  $G$  induced by the vertex set  $K$ .

Given a set of colors  $C = \{1, 2, \dots, k\}$ , a *coloring constraint* function  $\Gamma : V \rightarrow 2^C$  associates with each  $v \in V$  a subset of colors  $\Gamma(v) \subseteq C$ . For  $S \subseteq C$ , we define a subset  $H \subseteq V$  as  $S$ -colorful if  $|H| = |S|$  and there is a function  $c$  that assigns each  $v \in H$  a color from  $\Gamma(v)$ , such that there is exactly one vertex in  $H$  of each color in  $S$ . The basic problem that we study is the following:

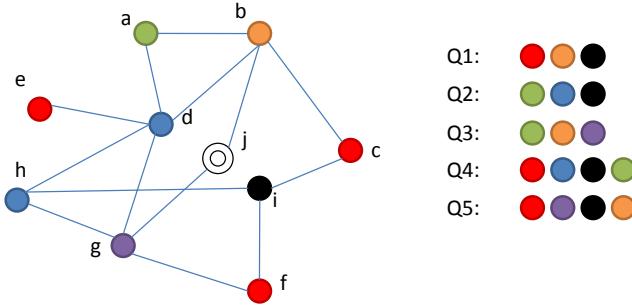
*Problem 1 (C-COLORFUL CONNECTED SUBGRAPH).* Given a graph  $G = (V, E)$ , a color set  $C$ , and a coloring constraint function  $\Gamma : V \rightarrow 2^C$ , is there a connected subgraph of  $G$  that is  $C$ -colorful?

This problem was shown to be NP-complete by Fellows et al. [8], even for the case of trees of maximum degree 3. Here we provide fixed-parameter tractable algorithms for several variants of this problem, where the parameter is the size of the query complex. A problem is fixed-parameter tractable with respect to a parameter  $k$  if an instance of size  $n$  can be solved in  $O(f(k) \cdot n^{O(1)})$  time, where  $f$  is an arbitrary function. Thus, fixed-parameter algorithms allow solving relatively large instances of NP-hard problems exactly [19], as long as the parameter value is modest.

### 2.1 Single Color Constraints

In the first variant of the problem, we consider only coloring constraint functions that associate each  $v \in V$  with a single color. In this case, the input is a graph where each vertex is assigned a color from  $C$ , and we aim to find a connected subgraph having exactly one vertex of each color.

Since every connected subgraph has a spanning tree, it suffices to look for colorful trees. This problem has been studied by Scott et al. [26] in another context, as well as by Kalaev et al. [13], Betzler et al. [4]. For completeness, we provide a dynamic programming (DP) algorithm, which is the unweighted version of the algorithm given by Scott et al. [26]. We construct a table  $B$  with rows corresponding to vertices and columns corresponding to subsets  $C' \subseteq C$ . We define  $B(v, S) = \text{True}$  if there exists in  $G$  a subtree rooted at  $v$  that is  $S$ -colorful,



**Fig. 1.** Network query problems. Left: the network, where vertex  $j$  is non-colored. Right: queries. For the basic problem disallowing indels,  $Q_1$  is solved by  $\{c, b, i\}$ , while  $Q_2$  and  $Q_4$  have no solution. When allowing a single arbitrary insertion,  $Q_2$  has solution  $\{a, d, h, i\}$  and  $Q_4$  has the solution  $\{a, b, c, d, i\}$ . When allowing a single special insertion,  $Q_3$  has the solution  $\{a, b, g, j\}$ . When allowing one deletion,  $Q_2$  has the solutions  $\{a, d\}$ ,  $\{i, f\}$ . When allowing repeated nodes and no indels,  $Q_5$  has the solution  $\{b, c, i, f, g\}$ .

and *False* otherwise. For  $S = \{\gamma\}$  and  $v \in V$  we initialize  $B(v, \gamma) = \text{True}$  iff  $F(v) = \{\gamma\}$ . Other entries of  $B$  can be computed using the following recurrence:

$$B(v, S) = \bigvee_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ \Gamma(v) \in S_1, \Gamma(u) \in S_2}} B(v, S_1) \wedge B(u, S_2), \quad (1)$$

The algorithm runs in  $O(3^k m)$  time<sup>1</sup>. One can easily generalize (I) to the weighted case, where each edge is assigned a weight, and the heaviest tree is sought.

*Insertions and Deletions.* Exact matches are often impossible due to evolutionary variation and noise in the data. Hence, we would like to allow deletions of query proteins that cannot be matched and insertions of network proteins that assist in connecting matched vertices. Deletions can be directly handled by the DP algorithm: If no  $C$ -colorful solution was found, then  $B(v, C) = \text{False}$  for all  $v$ . Allowing up to  $N_{dels}$  deletions can be done by scanning the entries of  $B$ . If there exists  $\hat{C} \subseteq C$  such that  $|\hat{C}| \geq |C| - N_{dels}$ , and  $B(v, \hat{C}) = \text{True}$  then a valid solution exists.

When allowing insertions, there are several problem variants to consider (see Figure II). In the first variant, some network vertices are not assigned a color, and only non-colored vertices can be inserted. For convenience, assign non-colored vertices the color 0. Let us call such insertions *special*.

<sup>1</sup> It can be further reduced to  $O(2^k m)$  using the techniques of Björklund et al. [5]; however, this version cannot be generalized to the weighted case, so we do not use it in the following.

**Definition 1.** An  $S$ -colorful solution allowing  $j$  special insertions is a connected subgraph  $H \subseteq G$ , where  $\exists H' \subseteq H$  such that  $V(H')$  is  $S$ -colorful and all other vertices of  $H$  are non-colored.

An obvious extension of the DP algorithm to handle up to  $N_{\text{ins}}$  special insertions is based on the color-coding paradigm of Alon et al. [1]: Randomly color the non-colored vertices with  $N_{\text{ins}}$  new colors and use DP to look for colorful trees. This procedure is repeated a sufficient number of times to ensure that every tree is colorful with high probability. However, the running time increases by a factor of  $(3e)^{N_{\text{ins}}}$ . We provide a more efficient solution below.

**Theorem 1.** Finding a  $C$ -colorful connected subgraph with up to  $N_{\text{ins}}$  special insertions can be solved in  $O(3^k m N_{\text{ins}})$  time.

*Proof.* We extend the DP table to represent also the number of special insertions used in an intermediate solution. Formally,  $B(v, S, j)$  iff there is an  $S$ -colorful subtree rooted at  $v$  that allows  $j$  special insertions, and  $j$  is the minimal number of insertions possible. Here  $j$  ranges between 0 and  $N_{\text{ins}}$ . We initialize the table by setting all entries to *False*, except: (i) For  $\gamma \neq 0$ ,  $B(v, \{\gamma\}, 0)$  iff  $\Gamma(v) = \gamma$ ; and (ii) if  $\Gamma(v) = 0$ ,  $B(v, \emptyset, 1)$ . Entries for which  $|S| \geq 1$  and  $j > 0$  are then computed using the following recurrence:

$$B(v, S, j) = \left[ \bigvee_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ j_1 + j_2 = j}} B(v, S_1, j_1) \wedge B(u, S_2, j_2) \right] \wedge \forall j' < j : \neg B(v, S, j') \quad (2)$$

We prove correctness by induction on  $|S|$  and  $j$ . The cases  $j = 0$  and  $S = \emptyset$  are immediate. Therefore, consider  $j > 0$  and as the first case  $S = \{\gamma\}$  (i.e.,  $|S| = 1$ ). By definition:

$$\begin{aligned} B(v, \{\gamma\}, j) &\iff \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ &B(v, S_1, j_1), B(u, S_2, j_2), S_1 \uplus S_2 = \{\gamma\}, j_1 + j_2 = j, \forall j' < j : \neg B(v, \{\gamma\}, j') \end{aligned}$$

Assuming there are  $u, S_1, S_2, j_1, j_2$  as above, then  $S_1$  cannot be  $\{\gamma\}$  since  $\forall j' < j : \neg B(v, \{\gamma\}, j')$ . It follows that  $S_1 = \emptyset$  and  $S_2 = \{\gamma\}$ , implying that  $\Gamma(v) = 0$ ,  $j_1 = 1$ , and  $j_2 = j - 1$  (see initialization of  $B$ ). By the induction hypothesis on  $j$ ,  $B(u, \{\gamma\}, j - 1)$  implies that there exists a tree  $T$  rooted at  $u$  having one vertex colored  $\gamma$  and a minimal number of  $j - 1$  non-colored vertices. Clearly,  $v \notin T$ . Otherwise, there will be a tree  $T'$  rooted in  $v$  having one vertex colored  $\gamma$  and  $j' < j$  special vertices, in contradiction to the minimality of  $j$ . Since  $u \in N(v)$ , then  $T \uplus \{v\}$  is a tree having one vertex colored  $\gamma$  and  $j$  non-colored vertices, as desired.

It remains to handle the case where  $|S| > 1$ . By definition:

$$\begin{aligned} B(v, S, j) &\iff \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ &B(v, S_1, j_1), B(u, S_2, j_2), S_1 \uplus S_2 = S, j_1 + j_2 = j, \forall j' < j : \neg B(v, S, j') \end{aligned}$$

Suppose such  $u, S_1, S_2, j_1, j_2$  exist. Then by the induction hypothesis, there is a tree  $T_v$  rooted at  $v$  that is  $S_1$ -colorful and contains a minimal number  $j_1$  of special vertices. Similarly, there is a tree  $T_u$  rooted at  $u$  that is  $S_2$ -colorful and contains a minimal number  $j_2$  of special vertices.  $T_u$  and  $T_v$  are clearly disjoint: Otherwise, there would be another tree  $T'$  rooted at  $v$  which is  $S$ -colorful and contains  $j' < j_1 + j_2$  special vertices, in contradiction to  $\neg B(v, S, j')$ . Since  $u \in N(v)$ , the union of these trees is  $(S_1 \uplus S_2)$ -colorful and has  $j_1 + j_2$  special vertices, as desired.

To achieve the stated running time, we maintain an auxiliary function  $t(v, S)$  which is set to  $j$  when for the first time  $B(v, S, j)$  is true for some  $j$ . In the recursion (2), we replace the condition  $j_1 + j_2 = j$  by  $t(v, S_1) + t(u, S_2) = j$ . Since the table is  $(N_{\text{ins}} + 1)$  times the size of the table in the basic case, the running time increases by a factor of  $N_{\text{ins}}$  compared to the basic case.  $\square$

In a second variant of insertion handling, any vertex can be inserted (rather than only non-colored ones). We solve this variant by using the algorithm for the problem with special insertions as a black box. Instead of running the algorithm on the input graph  $G$ , we run it on an auxiliary graph  $G' = (V', E')$ , which is constructed as follows: Add a non-colored copy  $v^0$  for each  $v \in V$ , and set  $E' = E \cup \{(v^0, u) \mid (v, u) \in E\} \cup \{(v^0, u^0) \mid (v, u) \in E\}$ . This variant has a running time of  $O(N_{\text{ins}} 3^k m)$ .

## 2.2 Multiple Color Constraints

We now turn to the more general case, where a color constraint function can associate each vertex with a set of colors and not just a single color. This problem arises when a network vertex protein is homologous to several query proteins. Betzler et al. [4] gave a fixed-parameter algorithm for the problem, where the running time is increased by a factor of  $(2e)^k$  compared to the case of single color constraints. Here we give an alternative fixed-parameter algorithm (coupled with some speedup heuristics). The basic idea is to reduce the problem to the single color case by randomly choosing a single valid color for every vertex. Our main effort is in computing an upper bound on the number of coloring iterations needed.

Define a *color graph* to be a bipartite graph  $B = (V, C, E)$  where  $V$  is the set of network vertices,  $C$  is the set of colors and  $(v, c) \in E \iff c \in \Gamma(v)$ . Consider a possible match to the query; for clarity, we assume that this match does not contain insertions or deletions. Then we can prove the following bound:

**Theorem 2.** *The probability for a subset of vertices of size  $k$  to become colorful in a random coloring is at least  $\frac{1}{k!}$ .*

The above theorem implies an overall running time of  $O(k! 3^k m N_{\text{ins}}^2)$  in the case of multiple color constraints. However, this bound is excessive in many instances, for the following reason. Let  $V'$  be a colorful set of vertices. Following [23], define the *constraint graph*  $G(V')$  as follows: the vertices are the colors, and an edge exists between two colors  $\gamma_1, \gamma_2$  if there is a vertex  $v$  in  $V'$  such

that  $\gamma_1, \gamma_2 \in \Gamma(v)$ . The resulting graph is then partitioned into connected components  $P_1, P_2, \dots, P_s$ . This partition induces a partition of the colored network vertices into sets  $Q_1, Q_2, \dots, Q_s$ , where all the vertices of  $Q_i$  can be colored only by colors from  $P_i$ . The expected number of iterations required for a  $P_i$ -sized subset of  $Q_i$  to become colorful is bounded by  $|P_i|!$ , and thus the number of iterations required for a solution of size  $k$  to become colorful is bounded by  $\prod_{i=1}^s |P_i|!$ . Therefore, since the number of iterations required by the algorithm is bounded by the number of iterations required before  $V'$  becomes colorful, the expected number of iterations of the algorithm is also bounded by the same product. Note, however, that the bound cannot be precomputed, although an upper bound can be obtained by taking  $V' = V$ .

We can reduce this upper bound using the following two rules: (i) If for some  $i$ , the product of all color degrees in  $Q_i$  is smaller than  $|P_i|!$ , then it is beneficial to exhaustively enumerate all possible colorings of  $Q_i$ . (ii) By Hall's Theorem [17], if a graph has a perfect matching and its minimum degree is  $d$ , then it has at least  $d!$  perfect matchings. Therefore, if the minimal color degree in  $Q_i$  is  $d$ ,  $\frac{|P_i|!}{d!}$  random iterations suffice.

### 2.3 An Integer Programming Formulation

In this section we provide an ILP formulation of the network querying problem, allowing us to employ industrial solvers that on certain instances are faster than DP. Formally, the problem that we aim to solve using the ILP is Problem  $\square$  (*C-COLORFUL CONNECTED SUBGRAPH*) with exactly  $N_{\text{ins}}$  arbitrary insertions. Further, we are given edge weights  $\omega : E \rightarrow \mathbb{Q}$  and wish to find a vertex subset  $K \subseteq V$  of size  $t := k + N_{\text{ins}}$  that maximizes the total edge weight  $\sum_{(v,w) \in E, v, w \in K} \omega_{vw}$ .

We declare binary variables  $\{c_v : v \in V\}$  that express whether a vertex  $v$  is selected into the complex  $K$ . It is easy to give constraints that ensure correct coloring; the difficulty is in expressing the connectivity. The idea is to find a flow<sup>2</sup> with  $t - 1$  selected vertices as sources of flow 1, and a selected sink  $r$  that drains a flow of  $t - 1$ , while disallowing flow between non-selected vertices. We use the following variables:

$$\{c_v : v \in V\}, c_v \in \{0, 1\} \quad \text{vertex } v \text{ is selected } (v \in K) \quad (3)$$

$$\{e_{vw} : (v, w) \in E, v < w\}, e_{vw} \in \{0, 1\} \quad \text{edge } (v, w) \text{ is in } G[K] \quad (4)$$

$$\{r_v : v \in V\}, r_v \in \{0, 1\} \quad \text{vertex } v \text{ is the sink} \quad (5)$$

$$\{f_{vw}, f_{wv} : (v, w) \in E\}, f_{vw}, f_{wv} \in \mathbb{Q} \quad \text{flow from } v \text{ to } w/w \text{ to } v \quad (6)$$

$$\{g_{v\gamma} : v \in V, \gamma \in \Gamma(v)\}, g_{v\gamma} \in \{0, 1\} \quad \text{vertex } v \text{ has color } \gamma \quad (7)$$

and the following constraints

---

<sup>2</sup> That is, a function  $f : V \times V \rightarrow \mathbb{Q}$  that satisfies skew symmetry ( $\forall v, w \in V : f(v, w) = -f(w, v)$ ) and flow conservation ( $\sum_{w \in V} f(v, w) = 0$ ) for all vertices  $v$  except sources and sinks; see e.g. Cormen et al. [7] for an introduction on flows.

$$\sum_{v \in V} c_v = t \quad (8)$$

$$\sum_{v \in V} r_v = 1 \quad (9)$$

$$e_{vw} \leq \frac{1}{2}c_v + \frac{1}{2}c_w \quad \forall (v, w) \in E \quad (10)$$

$$f_{vw} = -f_{wv} \quad \forall (v, w) \in E \quad (11)$$

$$\sum_{w \in N(v)} f_{vw} = c_v - tr_v \quad \forall v \in V \quad (12)$$

$$f_{vw}, f_{wv} \leq (t-1)e_{vw} \quad \forall (v, w) \in E \quad (13)$$

$$\sum_{\gamma \in \Gamma(v)} g_{v\gamma} \leq 1 \quad \forall v \in V \quad (14)$$

$$\sum_{v \in V} g_{v\gamma} = 1 \quad \forall \gamma \in C \quad (15)$$

$$g_{v\gamma} \leq c_v \quad \forall v \in V, \gamma \in \Gamma(v) \quad (16)$$

with the objective

$$\text{maximize} \sum_{(v,w) \in E} \omega_{vw} e_{vw}. \quad (17)$$

The ILP can be easily adapted to allow  $N_{\text{del}}$  deletions by changing the constraint (13).

### 3 The Implemented Algorithm

We implemented a pipeline for querying a complex given as a set of proteins from a source species, in the PPI network of a target species. For each complex, TORQUE is applied with increasing number of allowed indels until a match is found or a pre-specified bound on the number of indels is reached. We use the multiple colors per vertex model and arbitrary insertions. The possible matches for the query in each network sub-component (see below) are assigned a score based on edge weights, and the highest scoring match is finally output. We will now describe the stages of the algorithm, the scoring scheme, and the parameters we used for our testing.

*Preprocessing.* A protein complex is specified as a set of proteins. We associate a distinct color with each query protein and define a corresponding coloring constraint function. Each vertex in the target network is associated with a subset of colors corresponding to the query proteins it is sequence-similar to. In practice, only 5% of the vertices on average are associated with one or more colors. The rest are treated as non-colored.

While some of the non-colored vertices can be used as insertion vertices, many are too far from any colored vertex to be feasible insertions under the given upper bound  $N_{\text{ins}}$ . Let  $v$  be a non-colored vertex, and let  $d_0(u, v)$  be the length (number of edges) of the shortest path between  $u$  and  $v$  where every vertex on the path is required to be non-colored. We keep  $v$  if there are colored vertices  $u_1, u_2$  such that  $d_0(u_1, v) + d_0(u_2, v) \leq N_{\text{ins}} + 1$ , and the corresponding paths are vertex-disjoint. Otherwise, we remove  $v$  from the network. On the networks and complexes that we tested (see below), subnetworks containing only colored vertices are usually of size less than 50, those allowing 1–2 insertions have 200–1000 vertices, and those allowing more insertions typically cover up to 99% of the network.

After computing the current subnetwork to search in, we partition it into its connected components and search in each one independently. We call a component *feasible* if the color constraints of its vertices contain at least  $k - N_{\text{dels}}$  colors of the query. Next, we process feasible connected components of increasing size, searching for the highest scoring matched complex using the DP or the ILP methods. We increase the number of indels, generating larger connected components, until a solution is found that contains the minimal number of insertions and deletions, where insertions are preferred over deletions, as they can be better attributed to incomplete data.

*Running the query algorithms.* When querying a connected component, its unique properties dictate which of our two methods will find a match more efficiently. As a rule of thumb, when the number of vertices is very close to the number of colors  $k$ , and  $k$  is large, the ILP algorithm is preferable, since we have observed that its running time is not as sensitive to  $k$ , while the color-coding algorithm has an exponential dependency. Empirically, we apply the ILP algorithm whenever  $2^{n-k} < 3^k$ , where  $n$  is the size of the connected component. This condition was satisfied in about 2/3 of the queries we used. For the DP algorithm, we used the multiple colors per vertex model, generating coloring assignments for the vertices using the bounds described in Section 2.2, thus reducing the number of iterations required.

*Scoring.* We score a set of proteins matching a query using the approach of Sharan et al. [28]. Briefly, a match is assigned a likelihood ratio score, which measures its fit to a protein complex model (assuming that every two proteins in a complex should interact with high probability, independently of all other pairs) vs. the chance that its connections in the target network arise at random. To accommodate for information on the reliability of interactions, the interaction status of every vertex pair is treated as a noisy observation, and its reliability is combined into the likelihood score.

When applying the DP algorithm, we output the highest scoring tree rooted at each vertex. Then, for each such solution tree, we compute the score of the subgraph that is induced by its vertices, taking into account edges and non-edges, to produce a final score for this vertex set. For the ILP, we improve the running time by assuming that all negatively weighted vertex pairs  $(v, w)$  (non-edges in the network) have the same weight penalty, thus avoiding to introduce integer variables  $e_{vw}$ .

*Parameter setting.* Our tests were performed using the following set of parameters. We queried complexes of size 4–25. Query and network protein sequence similarities were evaluated using BLAST. For a vertex  $v$  and a color  $\gamma$ , we let  $\gamma \in \Gamma(v)$  if the BLAST E-value obtained by comparing the sequences of  $v$  and the query protein corresponding to  $\gamma$  was less than  $10^{-7}$ . For each complex, we allowed TORQUE to run at most 4000 seconds, and took the best solution up to that point. We set the number of allowed insertions and deletions to 2 of each for small complexes (size < 7), 3 of each for medium sized complexes (size 8–14), and 4 of each for larger complexes. We intend to explore the robustness of these parameters in the journal version.

The algorithms were implemented with Python 2.5.2; for the ILP we used CPLEX 11.0.1. The test machine was a 3 GHz Intel Xeon (only one CPU was used) with 8 G of memory, running Debian GNU/Linux 4.0.

## 4 Experimental Results

We applied TORQUE to query protein complexes within the three largest eukaryotic PPI networks available to date: yeast (5430 proteins, 39936 interactions), fly (6650 proteins, 21275 interactions) and human (7915 proteins, 28972 interactions). As queries, we used six collections of protein complexes from different species: yeast, fly, human, bovine, mouse, and rat. The first three served us to validate our algorithm and compare it to the state-of-the-art QNet algorithm [23]<sup>3</sup>. The last three, for which no large-scale PPI information exists, allowed us to explore the power of the algorithm in querying protein complexes for which no topology information is available. In the following we describe the data, evaluation measures and the results obtained.

*Data acquisition.* For yeast, fly and human we obtained up-to-date PPI data, gathered from recently published papers [31, 24, 32, 11, 15, 22] and from public databases [34, 10, 20]. High-throughput mass spectrometry data [11, 15] was translated into binary PPIs using the spoke model [2]. Yeast complexes were downloaded from SGD [27] (Macromolecular Complex GO-Slim category). Fly complexes were obtained using the AmiGo [12] browser to collect all proteins annotated with GO:0043234 (protein complex). The complexes for all mammals (human, mouse, rat, bovine) were downloaded from the CORUM website [25].

*Quality evaluation.* To evaluate the quality of the matches, we used two measures: *functional coherence* and *specificity*. The first measure reports the percent of matches that are significantly functionally coherent with respect to the Gene Ontology (GO) [33] annotation. Note that while the query is functionally coherent, the reported matches may not be so due to permissive homology matching and the noise in the PPI data. To compute the functional coherence of a match,

---

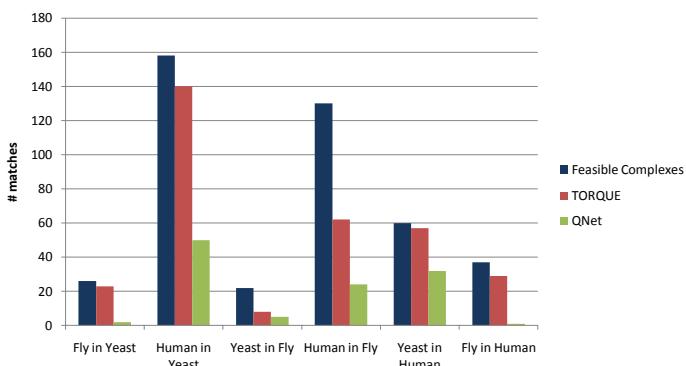
<sup>3</sup> A comparison to GraphFind [9] was not feasible, since its interface does not allow automated execution of the more than 600 queries.

represented as a set of proteins, we used the GO TermFinder [6] tool. The  $p$ -values returned by the tool were further corrected for multiple match testing using the false discovery rate (FDR) procedure [3].

The second measure reports the specificity of the suggested solution, i. e., the fraction of matches that significantly overlap with a known protein complex. The significance of the overlap was evaluated using the hypergeometric distribution. The resulting  $p$ -value was compared to those obtained on 100 random sets of proteins of the same size to produce an empirical  $p$ -value. Those  $p$ -values were FDR-corrected for multiple testing. In the specificity computation we focused on matches that had a non-zero overlap with the collection of complexes to which they were compared. We also report separately those *novel matches* that had no overlap with known complexes. Although it is possible that some of these non-overlapping matches are false positives, we believe that the high percentage of specific matches indicate that some - or most - of these are indeed novel complexes.

*Comparison to QNet.* Our first set of experiments focused on the yeast, fly and human networks and protein complex collections. For each of the three species, we queried its complexes in the networks of the other two species. As large-scale networks are available in this setting, we could compare ourselves to the QNet algorithm [23], which was designed to tackle topology-based queries. While exact topology for the query complexes is mostly unknown, QNet infers it by projecting the complexes onto the corresponding network. This results in a set of possible spanning trees for the complex that are hence provided to QNet as inputs. This makes QNet very dependent on the quality of the source network, in addition to the usual dependence on the quality and completeness of the target network. We used the original QNet code with the same machine setup and parameters as our algorithm: sequence similarity, insertions and deletions, and time limits.

A striking difference between TORQUE and QNet can be seen from the results in Figure 2—out of 433 feasible queries overall, TORQUE detected matches for 311, while QNet found matches for 114 only. As we show below, this 45% gain in sensitivity did not harm the specificity of the results.



**Fig. 2.** Comparison of number of matches for TORQUE and QNet

**Table 1.** Number and percentage of matches found that pass a quality significance threshold of 0.05

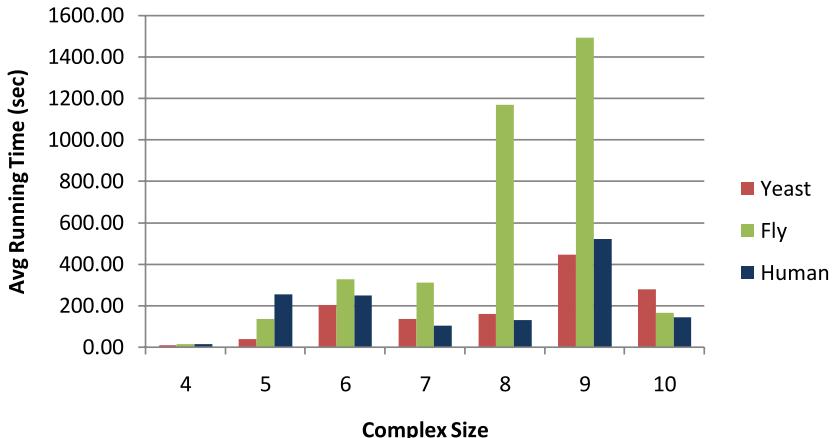
Network	Complex	Functional coherence		Specificity		Novel matches	
		TORQUE	QNet	TORQUE	QNet	TORQUE	QNet
Yeast	Fly	23 (100%)	2 (100%)	19 (82%)	2 (100%)	7	0
	Human	134 (95%)	49 (98%)	119 (85%)	47 (94%)	8	2
Fly	Yeast	8 (100%)	3 (60%)	8 (100%)	4 (80%)	1	0
	Human	56 (90%)	21 (87%)	62 (100%)	23 (95%)	22	5
Human	Yeast	48 (84%)	25 (78%)	43 (75%)	23 (71%)	8	6
	Fly	21 (72%)	0 (—)	21 (72%)	0 (—)	7	0
Total		290	100	272	99	46	13

**Table 2.** Statistics of querying protein complexes for which no topology information is available

Network Complex		#Feasible	#Matches	Functional coherence	Specificity	Novel matches
Yeast	Bovine	4	4	4	4	0
	Mouse	17	17	16	13	1
	Rat	23	20	19	9	6
Fly	Bovine	3	0	-	-	-
	Mouse	14	7	0	1	6
	Rat	34	21	17	7	14
Human	Bovine	4	4	2	1	0
	Mouse	48	46	32	24	6
	Rat	44	43	32	24	4
Total		168	162	122	83	37

Next, we turned to evaluate the results using the functional coherence and specificity measures described above. The results for the three data sets are summarized in Table 2. As evident from the table, even though TORQUE matched many more queries, its results exhibit higher functional coherence and similar specificity levels.

*Topology-free queries.* A unique characteristic of TORQUE is its ability to query protein complexes for which a topology is not known. Here we apply our algorithm to query, for the first time, sets of protein complexes of mouse (59 complexes), rat (55) and bovine (10) – species for which no large scale PPI data are currently available. In Table 2, we present the results of querying these complexes within the networks of yeast, fly, and human. As evident from the table, more than 95% of the feasible queries had a match, and the majority of the matches were functionally enriched or matched a known complex.



**Fig. 3.** Running time as a function of complex size

*Running time.* The running time of TORQUE depends on many factors: complex size, number of homologs for each query protein, and the size of the connected component being tested. Figure 3 shows the running time of TORQUE for complexes of size up to 10, for those instances that the algorithm finished its processing on within the 1 hour time limit. Queries of size > 10 were handled by the ILP algorithm, whose average running time was 0.29 seconds for the queries whose processing was completed. Out of the total 624 feasible queries of all sizes, the processing of 122 was not completed in time.

## 5 Conclusions

We presented a tool for querying protein complexes for which no topology information is available within a PPI network. Compared to a topology-based approach, our program produces many more matches that are highly functionally enriched. Thus, our tool seems practical for a wide range of network query tasks, in particular involving species with sparse data. It would be interesting to examine matches for biological significance, in particular those that do not overlap any known complex and may suggest unknown complexes of the target species. This may also allow us to design a fine-tuned cost model, that takes costs of insertions, deletions, and particular protein mappings into account.

**Acknowledgments.** We thank Noga Alon for his help in analyzing the case of multiple color constraints. We thank Banu Dost for providing us with the QNet code, and Nir Yosef for providing the PPI networks. R. Shamir and R. Sharan were supported in part by the Israel Science Foundation (grant no. 385/06). F. Hüffner was supported by a postdoctoral fellowship from the Edmond J. Safra Bioinformatics Program at Tel Aviv University.

## References

- [1] Alon, N., Yuster, R., Zwick, U.: Color coding. *Journal of the ACM* 42, 844–856 (1995)
- [2] Bader, G.D., Hogue, C.W.: Analyzing yeast protein-protein interaction data obtained from different sources. *Nature Biotechnology* 20(10), 991–997 (2002)
- [3] Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)* 57(1), 289–300 (1995)
- [4] Betzler, N., Fellows, M.R., Komusiewicz, C., Niedermeier, R.: Parameterized algorithms and hardness results for some graph motif problems. In: Ferragina, P., Landau, G.M. (eds.) *CPM 2008. LNCS*, vol. 5029, pp. 31–43. Springer, Heidelberg (2008)
- [5] Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: Proc. 39th STOC, New York, pp. 67–74 (2007)
- [6] Boyle, E.I., Weng, S., Gollub, J., Jin, H., Botstein, D., Cherry, J.M., Sherlock, G.: GO::TermFinder—open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics* 20(18), 3710–3715 (2004)
- [7] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn. MIT Press, Cambridge (2001)
- [8] Fellows, M.R., Fertin, G., Hermelin, D., Vialette, S.: Borderlines for finding connected motifs in vertex-colored graphs. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) *ICALP 2007. LNCS*, vol. 4596, pp. 340–351. Springer, Heidelberg (2007)
- [9] Ferro, A., Giugno, R., Mongiovì, M., Pulvirenti, A., Skripin, D., Shasha, D.: Graphfind: enhancing graph searching by low support data mining techniques. *BMC Bioinformatics* 9(suppl. 4), 1471–2105 (2008)
- [10] FlyBase-Consortium. The FlyBase database of the drosophila genome projects and community literature. *Nucleic Acids Research*, 31(1):172–175 (2003)
- [11] Gavin, A.C., Aloy, P., Grandi, P., Krause, R., Boesche, M., Marzioch, M., Rau, C., Jensen, L.J., Bastuck, S., Dompelfeld, B., et al.: Proteome survey reveals modularity of the yeast cell machinery. *Nature* 440(7084), 631–636 (2006)
- [12] GO Consortium. Amigo (September 2008), <http://amigo.geneontology.org/>
- [13] Kalaev, M., Bafna, V., Sharan, R.: Fast and accurate alignment of multiple protein networks. In: Vingron, M., Wong, L. (eds.) *RECOMB 2008. LNCS (LNBI)*, vol. 4955, pp. 246–256. Springer, Heidelberg (2008)
- [14] Kelley, B.P., Yuan, B., Lewitter, F., Sharan, R., Stockwell, B.R., Ideker, T.: PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Research* 32(Web Server issue) (July 2004)
- [15] Krogan, N.J., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., Li, J., Pu, S., Datta, N., Tikuisis, A.P., et al.: Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. *Nature* 440(7084), 637–643 (2006)
- [16] Lacroix, V., Fernandes, C., Sagot, M.: Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3(4), 360–368 (2006)
- [17] Lovász, L., Plummer, M.D.: *Matching Theory*. Annals of Discrete Mathematics, vol. 29. North-Holland, Amsterdam (1986)

- [18] Narayanan, M., Karp, R.M.: Comparing protein interaction networks via a graph match-and-split algorithm. *Journal of Computational Biology* 14(7), 892–907 (2007)
- [19] Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications, vol. 31. Oxford University Press, Oxford (2006)
- [20] Peri, S., Navarro, J.D., Amanchy, R., Kristiansen, T.Z., Jonnalagadda, C.K., Surendranath, V., Nirajan, V., Muthusamy, B., Gandhi, T.K., Gronborg, M., et al.: Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Research* 13(10), 2363–2371 (2003)
- [21] Pinter, R.Y., Rokhlenko, O., Yeger-Lotem, E., Ziv-Ukelson, M.: Alignment of metabolic pathways. *Bioinformatics* 21(16), 3401–3408 (2005)
- [22] Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B.J., Hon, G.C., Myers, C.L., Parsons, A., Friesen, H., Oughtred, R., Tong, A., et al.: Comprehensive curation and analysis of global interaction networks in *Saccharomyces cerevisiae*. *Journal of Biology* 5(4), 11 (2006)
- [23] Sharan, R., Dost, B., Shlomi, T., Gupta, N., Ruppin, E., Bafna, V.: Qnet: A tool for querying protein interaction networks. *Journal of Computational Biology* 15(7), 913–925 (2008)
- [24] Rual, J.F., Venkatesan, K., Hao, T., Hirozane-Kishikawa, T., Dricot, A., Li, N., Berriz, G.F., Gibbons, F.D., Dreze, M., Ayivi-Guedehoussou, N., et al.: Towards a proteome-scale map of the human protein-protein interaction network. *Nature* 437(7062), 1173–1178 (2005)
- [25] Ruepp, A., Brauner, B., Dunger-Kaltenbach, I., Frishman, G., Montrone, C., Stransky, M., Waegele, B., Schmidt, T., Doudieu, O.N., Stümpflen, V., Mewes, H.W.: Corum: the comprehensive resource of mammalian protein complexes. *Nucleic Acids Research* 36(Database issue), 646–650 (2008)
- [26] Scott, J., Ideker, T., Karp, R.M., Sharan, R.: Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology* 13(2), 133–144 (2006)
- [27] SGD project. *Saccharomyces* genome database (September 2008), <http://www.yeastgenome.org/>
- [28] Sharan, R., Ideker, T., Kelley, B.P., Shamir, R., Karp, R.M.: Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology* 12(6), 835–846 (2005)
- [29] Shlomi, T., Segal, D., Ruppin, E., Sharan, R.: QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics* 7, 199 (2006)
- [30] Sohler, F., Zimmer, R.: Identifying active transcription factors and kinases from expression data using pathway queries. *Bioinformatics* 21(suppl. 2), ii115–ii122 (2005)
- [31] Stanyon, C.A., Liu, G., Mangiola, B.A., Patel, N., Giot, L., Kuang, B., Zhang, H., Zhong, J., Finley Jr., R.L.: A drosophila protein-interaction map centered on cell-cycle regulators. *Genome Biol.* 5(12), R96 (2004)
- [32] Stelzl, U., Worm, U., Lalowski, M., Haenig, C., Brembeck, F.H., Goehler, H., Stroedicke, M., Zenkner, M., Schoenherr, A., Koeppen, S., et al.: A human protein-protein interaction network: a resource for annotating the proteome. *Cell* 122(6), 957–968 (2005)

- [33] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics* 25, 25–29 (2000)
- [34] Xenarios, I., Salwinski, L., Joyce, X., Higney, P., Kim, S., Eisenberg, D.: Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research* 30, 303–305 (2002)
- [35] Yang, Q., Sze, S.-H.: Path matching and graph matching in biological networks. *Journal of Computational Biology* 14(1), 56–67 (2007)
- [36] Yu, et al.: High-quality binary protein interaction map of the yeast interactome network. *Science* 322(5898), 104–110 (2008)
- [37] Zheng, Y., Szustakowski, J.D., Fortnow, L., Roberts, R.J., Kasif, S.: Computational identification of operons in microbial genomes. *Genome Research* 12(8), 1221–1230 (2002)

# Cross Species Expression Analysis of Innate Immune Response

Yong Lu<sup>1,3</sup>, Roni Rosenfeld<sup>1</sup>, Gerard J. Nau<sup>2</sup>, and Ziv Bar-Joseph<sup>1,\*</sup>

<sup>1</sup> School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA  
zivbj@cs.cmu.edu

<sup>2</sup> Department of Molecular Genetics and Biochemistry, University of Pittsburgh Medical School, Pittsburgh, PA 15213, USA

<sup>3</sup> Present address: Department of Biological Chemistry and Molecular Pharmacology, Harvard Medical School, Boston, MA 02115, USA

**Abstract.** The innate immune response is the first line of host defense against infections. This system employs a number of different types of cells which in turn activate different sets of genes. Microarray studies of human and mouse cells infected with various pathogens identified hundreds of differentially expressed genes. However, combining these datasets to identify common and unique response patterns remained a challenge. We developed methods based on probabilistic graphical models to combine expression experiments across species, cells and pathogens. Our method analyzes homologous genes in different species concurrently overcoming problems related to noise and orthology assignments. Using our method we identified both core immune response genes and genes that are activated in macrophages in both human and mouse but not in dendritic cells, and vice versa. Our results shed light on immune response mechanisms and on the differences between various types of cells that are used to fight infecting bacteria.

**Supporting website:** <http://www.cs.cmu.edu/~lyongu/pub/immune/>

## 1 Introduction

Innate immunity is the first line of antimicrobial host defense in most multi-cellular organisms, and is instructive to adaptive immunity in higher organisms [12]. There are multiple types of immune cells, including macrophages, dendritic cells, and others. Depending on their role, each type of cell may respond by activating a different set of genes, even to the same bacteria [5]. In addition to the cell type, innate immune response differs based on the specific pathogen in question [32]. To date, gene expression profiling has been used to investigate transcriptional changes in human and mouse macrophages and dendritic cells during infection with several different pathogens [5,7,8,13,16,17,22,29,36]. In each of these studies, a list of genes involved in the response is determined by first ranking the genes based on their expression changes and then selecting the top-ranked genes based on a score or p-value cutoff. While some papers analyze data from multiple cell types or multiple pathogens, a large scale

---

\* Corresponding author.

comparison of these datasets across cells, pathogens and different species has not yet been performed.

Microarray expression experiments that study immune response to bacteria infection can be divided along several lines. Here we focus on three such divisions: cell types, bacteria types, and host species.

Innate immunity is the result of the collective responses of different immune cells, which are differentiated from multipotential hematopoietic stem cells [19]. To understand the roles of and possible interplays between different types of immune cells, it is important to identify both the common responses of different immune cells, as well as responses unique to a certain cell type. Identification of genes differentially expressed in macrophages but not in dendritic cells, and vice versa, may highlight their specific functions and help us understand mechanisms leading to their different immune response roles. In addition to the different cells, specific bacteria types are known to trigger very different innate immune responses [32]. Specifically, response to Gram-positive and Gram-negative bacteria is activated by different membrane receptors that recognize molecules associated with these bacteria. Finally, many of the key components in the innate immune system are highly conserved [15]. For example, the structure of Toll-like receptors (TLRs), a class of membrane receptors that recognizes molecules associated with bacteria, is highly conserved from *Drosophila* to mammals. It is less known though to what extent the immune response program is conserved and what other genes play a role in this conserved response.

While each of these subsets of experiments (macrophages vs. dendritic, human vs. mouse etc.) can be analyzed separately using ranking methods and then compared, due to noise in gene expression data methods that rely on a score cutoff become much less reliable for genes closer to the threshold [27]. Thus, analyzing responses to different pathogens and then examining the overlap between the lists derived for each experiment may not identify a comprehensive list of immune response genes. Similarly, while comparing the expression changes triggered by similar bacteria in human and mouse may lead to the identification of conserved immune response patterns, direct comparison of these profiles across experiments is sensitive to noise and orthology assignments, leading to unreliable results and underestimation of conservation [25].

In previous work [26,27] we combined expression datasets from several species to identify conserved cell cycle genes. The underlying idea is that pairs of orthologous genes are more likely than random pairs to be involved in the same cellular system. Thus, if one of the genes in the pair has a high microarray expression score while the other has a medium score, we can use the high scoring gene to elevate our belief in its ortholog, and vice versa. We used discrete Markov random fields (MRFs) to construct a homology graph between genes in different species. We developed a belief propagation algorithm to propagate information across species allowing orthologous genes to be analyzed concurrently.

Here we extend this method in several ways so that it can be applied to analyzing immune response data. Unlike the cell cycle, which we assumed worked in a similar way in all cell types of a specific species, here we are interested in both common responses and distinguishing responses for each dividing factor. This requires a different analysis of the posterior values assigned to nodes in the graph. In addition, for the immune response analysis, genes are represented multiple times in the graph (once for each cell and bacteria type) leading to a new graph topology. We are also

interested in multiple labels for immune response (up, down, not changing) compared to the binary labels we used for cell cycle analysis. Finally, in this paper we use a Gaussian random field instead of a discrete Markov random field leading to faster updates and improved analysis. Instead of simply connecting genes with high protein sequence similarity, the edges in the graph are determined in a novel way that enables us to utilize the information contained in sequence homology in a global manner, leading to improved prediction performance.

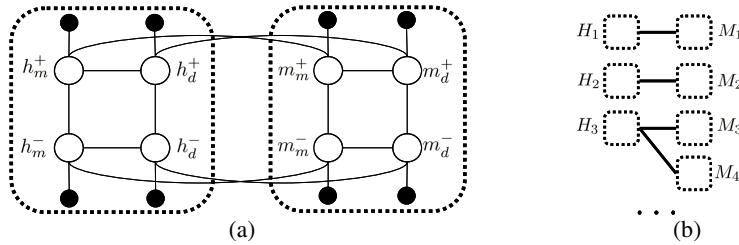
We have used our method to combine data from expression experiments across all three dividing factors. Our method identified a core set of genes containing many of the known immune response genes and a number of new predictions. In addition, our method successfully highlighted differences between conserved responses in macrophages and dendritic cells, shedding new light on the functions of these types of cells.

A number of papers used Markov random field models to integrate biological data sources. These include work on protein function prediction [6,24] and functional orthology prediction [2]. Our method has different goals and uses different data sources. In addition our work differs from these previous papers in several important aspects. Our method propagates information from different cell types and species to improve gene function prediction, while previous work either did not use cross-species information, or only used it to align networks from different species. We do so by defining our model on a network that explicitly represents genes from different species and cell types. In contrast, previous work either focused on a single species [6,24], or on an aligned network where each node represents an orthologous group [2]. Finally, our model is defined on continuous random variables instead of discrete variables, which enables us to predict three-class labels (up/unchanged/down), while most previous models only handle two-class labels.

## 2 Computational Model: Gaussian Random Field

We formulate the problem of identifying immune response genes using a probabilistic graphical model. In a probabilistic graphical model, random variables are represented by nodes in a graph, and conditional dependency relations are represented by edges. Probabilistic graphical models can be based on directed graphs or on undirected graphs. The model we use here is based on undirected graphs, where special functions (termed “potential functions”) are defined on nodes and edges of the graph, and the joint probability distribution is represented by the product of these potential functions. The form of the potential functions encodes our prior knowledge as well as modeling preferences.

We use Gaussian random fields (GRFs) to model the assignment of gene labels. Gaussian random fields are a special type of Markov random fields. In a GRF, every node follows a normal distribution, and all nodes jointly follow a multivariate normal distribution. There are two types of nodes in our graphical model (Fig. 1). The first type is a gene node; it represents the status of a gene in a certain cell type, from a certain host species, in response to a certain type of pathogen. Here we consider two cell types (macrophages and dendritic cells), two host species (humans and mice), and two pathogen types (Gram-negative and Gram-positive bacteria) although the model is general and can accommodate other types as well. The set of possible labels for



**Fig. 1.** Diagram of the Gaussian random field (GRF) model. (a) A subgraph in the GRF containing homologous human and mouse genes. The white node  $h_m^+$  represents the (latent) label of the human gene  $h$  in macrophages under infection of Gram-positive bacteria.  $h_m^-$  represents the gene's label in macrophages under infection of Gram-negative bacteria.  $h_d^+$  and  $h_d^-$  represent the labels of the same genes in dendritic cells under the infection of Gram-positive or Gram-negative bacteria.  $m_m^+$ ,  $m_m^-$ ,  $m_d^+$ , and  $m_d^-$  are similarly defined for the homologous mouse gene  $m$ . Two white nodes are connected by an edge if they represent the same gene in two experiments, either on the same cell type or under the infection of the same type of bacteria. We also connect two white nodes if they represent homologous genes in the same cell type and under the infection of the same type of bacteria. The black nodes represent the observation from the expression data in a certain cell type and under the infection of the appropriate bacteria. They are connected with the white nodes representing the corresponding genes under the same condition. (b) A high level diagram of the GRF model. Each dotted box represents a subgraph of four nodes related to the same gene as those shown in (a), and each “edge” represents four edges connecting the nodes of homologous genes in the two dotted boxes, in the same way as shown in (a).

each gene can be either two (involved in immune response or not), or three (suppressed, induced, or unchanged during immune response). For simplicity, we will describe our model using binary labels, but will present the results based on both sets of possible labels.

Corresponding to each gene node is a score node, representing the observed expression profile of the corresponding gene. Together, the GRF jointly models the labels of all genes in all cell types, all species, and under both types of infection conditions. The edges in the GRF represent the conditional dependencies between gene labels. We put an edge between two gene nodes when they are a priori more likely to have the same label. Specifically, there are two cases where we add an edge. In the first case, for each gene node in the graph, we connect it with another gene node if the protein sequence similarity between these two genes is high and the experiments related to both nodes are in the same cell and bacteria types. The assumption is that genes with similar sequence are more likely to have similar function in the same type of cell and for the same bacteria. The edge potential function defined on these edges (presented below) introduces a penalty when two genes with high sequence similarity are assigned different labels. In the second case, we connect a gene node with another gene node if the two nodes represent the same gene in the same type of cell or bacteria. Here we assume the genes are likely to function similarly in the same type of cell, or under the same type of infection. Again, the potential function penalizes the case where a gene is assigned different label under different conditions for the same cell. The size of the penalty depends on the strength or weight attached to the edge. Different edges may have different weights (see below). The joint probability is defined as

the product of the node potential functions and edge potential functions, divided by a normalization function. We can infer the label of individual genes by estimating the joint maximum *a posteriori* (MAP) assignment of all nodes.

## 2.1 Computing the Weight Matrix

An important issue in random field models is the assignment of edge weights. Employing a similar approach but in a simpler setting, Lu et al. [26] use a Markov random field to jointly model gene statuses in multiple species, where edges in the graph are weighted by BLASTP [1] scores between pairs of genes. Given two genes connected in the graph, the edge weight (BLASTP bit score) represents the sequence similarity between the two genes, which in turn captures the a priori dependency between their labels. While this is a useful strategy, in a Markov random field model edges represent the dependency between the two nodes conditioned on the labels of all other nodes [3]. In contrast, sequence similarity is computed for a pair of genes regardless of other genes. In other words, what a BLASTP score captures is the marginal dependency between the two genes' labels rather than the conditional dependency.

To address this issue we compute new edge weights using the BLASTP score matrix, which captures the marginal covariance of the Gaussian random field. It has been shown that for GRFs the appropriate weight matrix is equal to the inverse of the marginal covariance matrix [39].

Using this observation, we can build a similarity matrix based on BLASTP scores, and use its inverse as the weight matrix for the GRF. Each row (and each column) in the similarity matrix corresponds to a gene. If the BLASTP bit score between two genes is above a cutoff, we set the corresponding elements in the similarity matrix to that score. Otherwise, it's set to zero. We use a stringent cutoff (Results) so that we are fairly confident of the functional conservation when we add a non-zero element. Because the similarity matrix contains scores for all genes in two species, the computational cost to invert it is very high. We thus compute an approximate inverse. We first convert the matrix into a diagonal block matrix by Markov clustering algorithm [9], then compute the approximate inverse by inverting each block independently. The matrix inversion is done using the Sparse Approximate Inverse Preconditioner [14].

Finally, we assign edge weights based on this inverse matrix. Note that each gene is represented by four nodes in the graph, because it is present in different experiments on two cell types and two types of pathogens. For edges connecting gene nodes in different species we set the weight according to the inverse similarity matrix. For edges connecting the same gene in different types of cells and bacteria we use a single hyperparameter as their edge weight for cell and bacteria relationships.

## 2.2 Expression Score Distribution

The gene expression score is a numeric summary computed from the gene's microarray time series, which we will define in Results. We assume that the scores of genes with the same label follow a Gaussian distribution with an experiment specific mean and variance. Due to noise in microarray experiments, these distributions are highly overlapping, making it hard to separate labels by expression score alone.

### 2.3 Node Potential Function

The node potential functions capture information from gene expression data. For each gene  $i$ , let  $C_i$  denote its (hidden) label,  $S_i$  denote its expression score,  $y_i$  denote the random variable in the GRF associated with this gene. As mentioned above  $C_i$  can be a binary variable or a ternary variable if we consider three gene labels.  $S_i$  and  $y_i$  are both real variables. Because each  $y_i$  follows a normal distribution, we need to have a way to link a gene's probability of belonging to each class with the corresponding normal distribution. This is achieved by the probit link function. In the binary labels case let  $p_i$  be the probability of gene  $i$  being an immune response gene conditioned on its expression score  $S_i$ ,

$$p_i = \Pr(C_i = 1 | S_i) = \frac{\Pr(S_i | C_i = 1) \Pr(C_i = 1)}{\Pr(S_i | C_i = 1) \Pr(C_i = 1) + \Pr(S_i | C_i = 0) \Pr(C_i = 0)}$$

For the GRF the node potential function is defined as

$$\psi_i(y_i) = \phi(y_i | \mu = \Phi^{-1}(p_i), \sigma^2 = 1) \quad (1)$$

where  $\phi(y_i | \mu, \sigma^2)$  is the probability density function for the normal distribution with mean  $\mu$  and variance  $\sigma^2$ , and  $\Phi^{-1}(x)$  is the probit function, i.e. the inverse cumulative distribution function for the standard normal distribution. In other words, the information from a gene's expression score is encoded by a normal distribution of  $y_i$  such that  $p_i = \Pr(y_i > 0)$ .

In the case of three labels for genes ( $C_i \in \{-1, 0, +1\}$ ), we can use the following formulas to link the probabilities of  $C_i$  and  $y_i$ :

$$\begin{aligned} \Pr(C_i = 1 | S_i) &= \Pr(y_i > 1), \quad \Pr(C_i = -1 | S_i) = \Pr(y_i \leq -1) \\ \Pr(C_i = 0 | S_i) &= \Pr(-1 < y_i \leq 1) \end{aligned} \quad (2)$$

It can be proven (see appendix) that given any (non-zero) probability mass function on  $C_i$ , we can find a normal distribution  $N(\mu, \sigma^2)$  such that these formulas are satisfied when  $y_i \sim N(\mu, \sigma^2)$ .

### 2.4 Edge Potential Function

The edge potential functions capture the conditional dependencies between pairs of gene nodes. The assumptions here are that (1) genes with higher sequence similarity are more likely than otherwise to have the same or similar functions; and (2) a given gene is likely to have the same function across cell types and across pathogens.

First we will define the edge potential functions for edges connecting homologous genes in the same cell type and under infection of the same type of bacteria. In this case, the edge potential function depends on the weight matrix we introduced above. Note that although all elements in the BLAST score matrix are non-negative (sequence similarities are non-negative), its inverse matrix may have negative elements. As a consequence, edge weights can be either positive or negative. A positive edge weight indicates that the labels of the two gene are positively correlated, *conditioned* on the labels of all other gene nodes. A negative edge weight means that they are negatively correlated, conditioned on the other gene nodes.

The following edge potential function captures this dependency ( $\lambda_0$  is a positive hyperparameter):

$$\psi_{ij}(y_i, y_j) = \begin{cases} \exp\{-\lambda_0 |w_{ij}|(y_i - y_j)^2\} & \text{if } w_{ij} \geq 0 \\ \exp\{-\lambda_0 |w_{ij}|(y_i + y_j)^2\} & \text{if } w_{ij} < 0 \end{cases}$$

When the edge weight  $w_{ij}$  is positive, the edge potential function places a penalty if  $y_i$  and  $y_j$  are different. The larger the difference, the higher the penalty. Likewise, when  $w_{ij}$  is negative, the edge potential function introduces a penalty based on how close  $y_i$  and  $y_j$  are to each other. The penalty becomes higher when we become more confident in  $y_i$  and  $y_j$  and the two are close.

For edges connecting the same gene in the same cell type but under infection of different types of bacteria, the edge potential function is defined as

$$\psi_1(y_i, y_j) = \exp\{-\lambda_1(y_i - y_j)^2\}$$

where  $\lambda_1$  is a positive hyperparameter. Similarly for edges connecting the same gene under the infection of the same type of bacteria but in different cell types, the edge potential is defined as

$$\psi_2(y_i, y_j) = \exp\{-\lambda_2(y_i - y_j)^2\}$$

where  $\lambda_2$  is a positive hyperparameter. Together, the joint likelihood function is defined as

$$L = \frac{1}{Z} \prod \psi_i(y_i) \prod \psi_{ij}(y_i, y_j) \prod \psi_1(y_i, y_j) \prod \psi_2(y_i, y_j).$$

### 3 Learning the Model Parameters

In this section we will present our algorithm based on two gene classes. The algorithm can be extended to three gene classes by using different node potential functions (See discussion in Section 2.3). For our model we need to learn the hyperparameters  $\lambda$ . We also need to learn the parameters of the expression score distributions for each combination of cell types, host species, and pathogen types. In each case, there are four parameters  $(\mu_0, \sigma_0^2, \mu_1, \sigma_1^2)$ , i.e. the means and variances of the two different Gaussian distributions, one corresponding to the scores of immune response genes, the other corresponding to the scores of the remaining genes.

We learn these parameters in an iterative manner, by an EM-style algorithm. We start from an initial guess of the parameters. Based on these parameters, we infer “soft” posterior assignments of labels to the genes using a version of the belief propagation algorithm on the GRF. The posterior assignments are in turn used to update the score distribution parameters. We repeat the belief propagation algorithm based on the new parameters to infer updated assignments of labels. This procedure goes on iteratively until the parameters and the assignments do not change anymore. Below we discuss these steps in detail.

**Table 1.** Algorithm for combining immune response gene expression data**Input**

1. expression score  $S_i$  for each gene in each cell type, host species, and pathogen type
2. graph structure (edge weights)

**Output**

For each gene node, its posterior probability of belonging to each class

**Initialization**

For each combination of host species, cell type, and pathogen type, compute estimates for  $\mu_0$ ,  $\sigma_0$ ,  $\mu_1$ , and  $\sigma_1$  using permutation analysis

**Iterate until convergence**

1. Use Belief Propagation to infer a posterior for each gene node
2. Use the estimated posterior to re-estimate the Gaussian expression score distributions

### 3.1 Iterative Step 1: Inference by Belief Propagation

Given the model parameters, we want to compute the posterior marginal distribution for each latent variable  $y_i$ , from which we can derive for each gene node the posterior probability of being involved in immune response. It is hard to compute the posteriors directly because the computational complexity of the normalization function in the joint likelihood function scales exponentially. However, due to the dependency structure in the GRF, we can adapt the standard Belief Propagation algorithm [38] for GRF, and use it to compute all the posteriors efficiently.

Unlike MRFs defined on discrete variables, variables in GRFs are continuous and follow normal distributions. The current estimation of the marginal posterior (“belief”) of every latent variable  $y_i$  in the GRF is a normal distribution. Similarly, the “messages” passed between nodes are also normal distributions.

The Belief Propagation algorithm consists of the following two steps: “message passing”, where every node in the GRF passes its current belief to all its neighbors, and “belief update”, where every node updates its belief based on all incoming messages. The algorithm starts from a random guess of the beliefs and messages, and then repeats these two steps until the beliefs converge.

- (1) Message passing. In this step, every node  $y_i$  computes a message for each of its neighbors  $y_j$ , sending  $y_i$ ’s belief of  $y_j$ ’s distribution. The message is based on the potential functions, which represent local information (node potential) and pairwise constraints (edge potential), as well as incoming messages from all  $y_i$ ’s neighbors *except*  $y_j$ .

$$m_{ij}(y_j) \leftarrow \int \psi_{ij}(y_i, y_j) \psi_i(y_i) \prod_{k \in N(i) \setminus j} m_{ki}(y_i) \cdot dy_i \quad (3)$$

- (2) Belief update. Once node  $y_i$  has received messages from all its neighbors, it updates the current belief incorporating all these messages and the local information from the node potential. The update rule is as follows

$$b_i(y_i) \leftarrow (1/v_i) \psi_i(y_i) \prod_{k \in N(i)} m_{ki}(y_i) \quad (4)$$

where  $v_i$  is a normalization constant to make  $b_i(y_i)$  a proper distribution.

Because all the messages and beliefs come from a normal distribution, they can be represented by the corresponding means and variances. Thus, in this case the message update rule and belief update rule above can be formulated into rules updating the means and variances directly, completely avoiding these computationally expensive integration operations. The exact update rules are given in the appendix.

### 3.2 Iterative Step 2: Updating the Score Distribution

The posterior computed in step 1 is based on the current (the  $g$ 'th iteration) estimation of parameters, collectively denoted by  $\Theta^{(g)}$ . The goal now is to determine the parameters that maximize the expected log-likelihood of the complete data over the observed expression scores given the parameters  $\Theta^{(g)} = (\mu_0^{(g)}, \sigma_0^{(g)}, \mu_1^{(g)}, \sigma_1^{(g)})$ .

To update the parameters of the score distributions, we first compute the posterior probability of a gene being involved in immune response, based on the posterior of  $y_i$ . This is the same as applying the reverse probit function:

$$\Pr(C_i = 1 | \Theta^{(g)}) = \int_0^{+\infty} b_i(y_i) dy_i$$

For simplicity, we use the following notations

$$p_i^{(g)} = \Pr(C_i = 1 | \Theta^{(g)}) \quad q_i^{(g)} = \Pr(C_i = 0 | \Theta^{(g)})$$

The updated distribution parameters for a Gaussian mixture are computed by standard rules

$$\begin{aligned} \mu_0^{(g+1)} &= \sum_i q_i^{(g)} S_i / \sum_i q_i^{(g)} & \mu_1^{(g+1)} &= \sum_i p_i^{(g)} S_i / \sum_i p_i^{(g)} \\ \sigma_0^{(g+1)} &= \sqrt{\frac{\sum_i q_i^{(g)} (S_i - \mu_0^{(g+1)})^2}{\sum_i q_i^{(g)}}} & \sigma_1^{(g+1)} &= \sqrt{\frac{\sum_i p_i^{(g)} (S_i - \mu_1^{(g+1)})^2}{\sum_i p_i^{(g)}}} \end{aligned}$$

Our algorithm is summarized in Table 1.

### 3.3 Learning the Hyperparameters

To learn the hyperparameters  $\lambda_0$ ,  $\lambda_1$  and  $\lambda_2$ , we use a list of known immune genes. These serve as training data for our algorithm. Following convergence of the belief propagation algorithm we optimize the prediction accuracy using the Nelder-Mead algorithm [33]. Note that this list is not used for the Results below. We divided our list of known immune genes and only used a third to learn the parameters. The other two thirds were used for the comparisons discussed below.

**Table 2.** Summary of datasets used

Host/Cell Type	Gram- Datasets	Gram+ Datasets
Human Macrophages	5	2
Human Dendritic Cells	9	2
Mouse Macrophages	7	7
Mouse Dendritic Cells	7	0

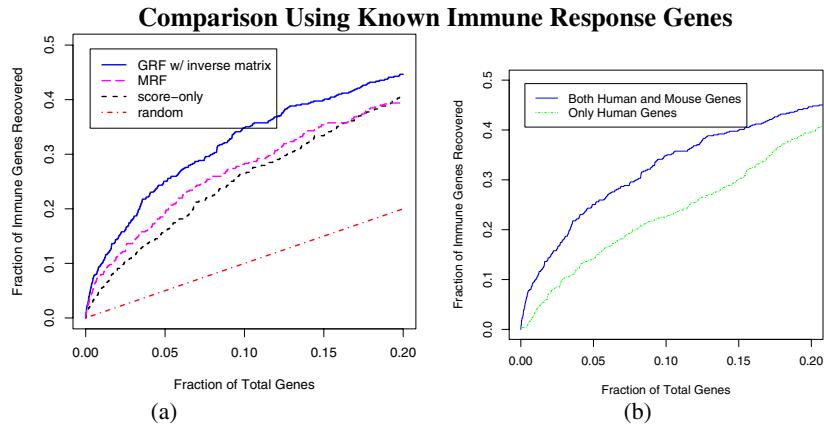
## 4 Results

**Immune response data.** Immune response microarray experiments were retrieved from supporting websites of [5,7,8,13,16,17,22,29,36], totaling 39 data sets. The data sets include experiments on macrophages and dendritic cells in humans and mice. For each cell type we have included experiments using Gram-positive and Gram-negative bacteria, except for mouse dendritic cells, for which we only found Gram-negative bacteria datasets. Human and mouse orthologs were downloaded from Mouse Genome Database [10]. Table 2 summarizes the datasets used in this paper.

**Computing expression scores and edge weights.** For each gene in each experiment, an expression score is computed from the gene expression time series data. The score is based on the slope of the time series to capture both the change in expression levels and the time between infection and response. Specifically, we first compare the absolute values of the highest and the lowest expression levels. The score is positive if the former is higher, or negative if the latter is higher. Denote the time point that corresponds to the highest absolute value of the expression level as  $t_i$ . The score is computed as follows:  $S_i = \text{expression}(t_i) / t_i$ . The score is positively correlated with the height of the peak expression value and increases the earlier this value is reached.

To compute the edge weights we first computed the BLASTP bit score between each pair of protein sequences. We turned the bit scores into a matrix, and set to zero those elements smaller than the 100 (our cutoff). We next computed an approximate sparse inverse of this matrix [14] and used it as the weight matrix for the graph.

**Recovering known human immune response genes.** To evaluate the performance of our model, we retrieved 642 known human innate immune response genes from [20], and used them as our labeled data. We learned the model parameters by three-fold cross validation using the labeled data. We compared the performance of GRF, MRF, and the baseline model where genes are ranked by their expression score alone. The MRF model is discussed in detail in [26]. We use the fraction of known immune response genes recovered by a model as the performance measure. Because the set of immune response genes we used does not have labels indicating the cell types or infection conditions, we treat a gene as “positive” regardless of the cell type and bacteria type. For GRF and MRF models, the genes were ranked by their highest posterior probability (in any of the cell or bacteria types). For the baseline model, the genes are ranked by their expression scores. As we show in Fig. 2 (a), both GRF and MRF models outperform the baseline model. These models are able to infer a better gene’s posterior probability by transferring information between the same gene across cell types or from homologous genes across species. For example, for the top 10% ranked genes, MRF is able to recover 28% of known immune response genes, compared with 26% by the baseline model. Encouragingly, GRF leads to the biggest improvement in performance. Of the top 10% high scoring genes based on the posterior computed by GRF, 35% are known immune response genes, a 35% increase compared to the baseline (score only) model.

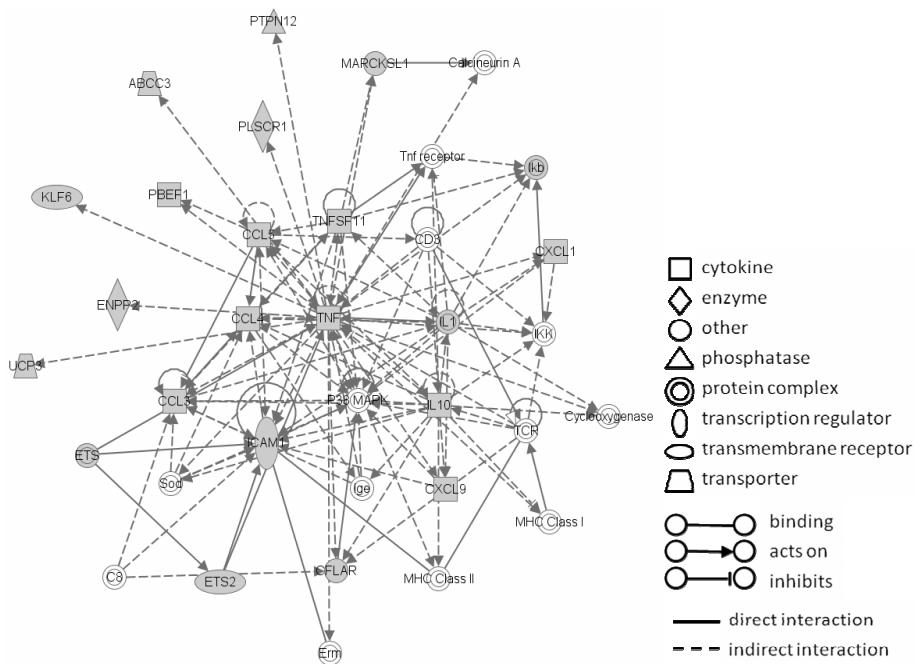


**Fig. 2.** (a) Performance comparison of the Gaussian random field (GRF) with improved weights, the Markov random field (MRF), and a baseline model ranking genes by their expression scores. Using MRF we were able to recover 18% of the known immune genes in the top 5% of ranked genes. This is a 28% improvement compared with the baseline model (which recovers 14% of the immune genes). The GRF model is able to recover 25% known immune genes at the same threshold, a 79% improvement over the baseline method and a 38% improvement over the MRF. (b) Performance comparison of GRF on two different graphs. The first graph contains genes from macrophages and dendritic cells in both human and mouse. The second graph contains genes from human macrophages and dendritic cells, but not from those in mouse. It can be seen that using homology information leads to large improvements.

To study the gain obtained by using cross species analysis we tested the performance of the GRF model when using only the human genes (removing the mouse genes from the graph). As we can be seen in Fig. 2(b), the performance of the GRF when only human genes are included is drastically reduced. The ROC curve when using this data is completely dominated by the curve of the results when using both species, even though the comparison is for recovering known human genes. This indicates that by combining data from both species we can improve the assignment of each species as well.

To determine how sensitive the computed expression cores are to experimental noise, we carried out similar analysis on data that was generated by adding a small Gaussian noise term ( $\text{mean}=0, \text{variance}=m^2$ , where  $m$  is the median expression difference between the first two time points) to each time point of the immune response data. For each of the noise added datasets we determined the score's performance for recovering known human immune response genes as discussed above. We repeated this process 50 times and found that the precision varies by 10% compared to the real data indicating the robustness of the computed scores (see Supporting website for details).

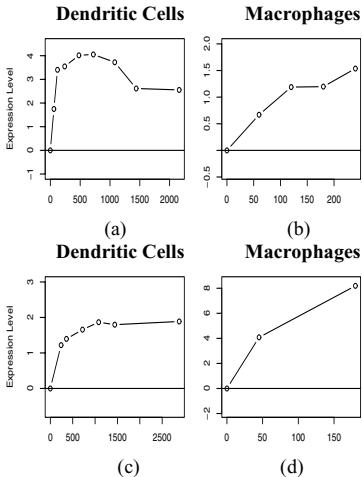
**Identification of common response genes by combined analysis.** Based on the learned posterior probabilities, we ranked the genes for each cell type in each species, for both Gram-positive and Gram-negative infections. We identified 57 ortholog pairs for which all nodes for both genes are assigned high posterior (see appendix). These genes are commonly induced by all bacteria in both macrophages and dendritic cells



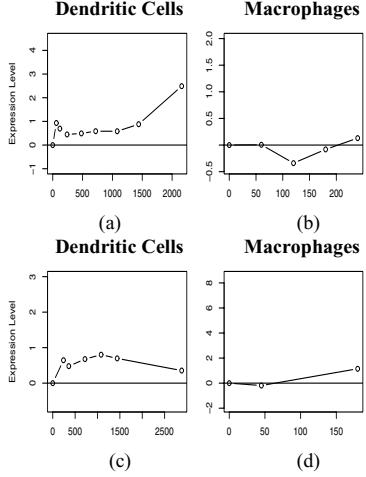
**Fig. 3.** One of the networks of genes commonly induced in both dendritic cells and macrophages when infected by bacteria, in both human and mouse. The network was constructed using Ingenuity Pathway Analysis ([www.ingenuity.com](http://www.ingenuity.com)). The gray-colored nodes are genes identified by our method. White-colored nodes are genes interacting with commonly induced genes. Note the large fraction of the pathway recovered by our method. Many known immune response genes are present in this network. IL1 is an important mediator of inflammatory response and involved in cell proliferation, differentiation, and apoptosis (Mizutani et al., 1991; Bratt and Palmblad, 1997). ETS2 is an important transcription factor for inflammation. CCL3, CCL4, and CCL5 are chemokines that recruit and activate leucocytes (Wolpe et al., 1988). The profiles for one of these genes, CCL5, are shown in Fig. 4.

across the two species (Fig. 4). As a sanity check, we first compared our list with a separate list of genes commonly induced in human *macrophages* by various bacteria. This latter list was derived from expression experiments that were not included in our analysis [32]. The results confirmed the lists we identified. The overlap between the two lists was highly significant with a  $p$ -value =  $1.70 \times 10^{-25}$  ( $p$ -value computed using hypergeometric distribution).

To reveal the functions of the common response genes we carried out GO enrichment analysis using STEM [11]. The enriched GO categories include many common categories involved in immune responses, including “immune response” ( $p$ -value= $3.9 \times 10^{-8}$ , all  $p$ -values corrected using Bonferroni), “inflammatory response” ( $p$ -value= $2.5 \times 10^{-7}$ ), “cell-cell signaling” ( $p$ -value= $1.1 \times 10^{-6}$ ), “defense response” ( $p$ -value= $1.5 \times 10^{-6}$ ), and “response to stress” ( $p$ -value= $2.4 \times 10^{-5}$ ).



**Fig. 4.** Expression profiles of CCL5 identified by our method as a common immune response gene. (a) and (b) expression profiles for human CCL5 in dendritic cells and macrophages. (c), (d) expression profiles for mouse CCL5 in dendritic cells and macrophages. Expression of both genes is strongly induced following infection.



**Fig. 5.** Expression profiles of CD86 identified to be activated only in dendritic cells. (a) and (b), expression profiles for human CD86 in dendritic cells and macrophages. (c), (d) expression profiles for mouse CD86 in dendritic cells and macrophages. For both species, the expression of CD86 is induced in dendritic cells, but unchanged following infection in macrophages (and only mildly induced at the end of the time course).

Our list recovered many of the classic players of innate immune activation and inflammation. For example, TNF is a pro inflammatory cytokine and stimulates the acute phase reaction [28]. IL1 is an important mediator of inflammatory response and involved in cell proliferation, differentiation, and apoptosis [4,31]. The list also includes chemokines that recruit and activate leucocytes (CCL3, CCL4, CCL5, CXCL1) [37] or attracts T-cells (CXCL9) [35]. Also important to the regulation of inflammation response is IL10, a well-known anti-inflammatory molecule [21]. Additionally, ETS2, NFkB, and JUNB are all very important transcription factors that are activated in inflammation [34]. In addition to recovering genes labeled in the IRIS database [20], which accounts for 21% of our predictions, we also successfully identified many immune response genes that were not included in the labeled dataset. Six out of the top 10 such genes are known to be commonly induced in host response in macrophages and dendritic cells [18], including PBEF1, an inhibitor of neutrophil apoptosis [23], and MMP14, an endopeptidase that degrades various components of the extracellular matrix [30]. See supporting website for complete list.

To identify the pathways involved in common immune response, we searched for networks enriched by common response genes using Ingenuity Pathway Analysis. One of these networks is shown in Fig. 3.

**Immune responses conserved in specific cell types.** In addition to genes commonly induced across all dividing factors, we also identified genes that are differentially

expressed between the two cell types. We identified 127 genes that are highly induced in dendritic cells in both bacteria types across human and mouse, but are not induced in macrophages (Fig. 5). GO enrichment analysis highlights some of the important characteristics of this set of genes, including “cell communication” ( $p\text{-value}=1.7\times10^{-10}$ ) and “signal transduction” ( $p\text{-value}=1.1\times10^{-9}$ ). (See supporting website for the complete lists.) Many of the genes are known to be associated with functions of dendritic cells, especially antigen processing and presentation. For example, components of the proteosome are prominently represented in the genes determined to be induced in dendritic cells. The proteosome is a necessary first step in MHC class I antigen presentation, a major function of dendritic cells. Peptides generated by the proteosome are then transported from the cytosol to endoplasmic reticulum by TAP, also represented in the gene list, where they are loaded on to MHC I molecules. Antigen presentation by DC is also accomplished through the class II pathway and the DC-specific gene list includes HLA-DRA, a human MHC II (class II) surface molecule. In addition to peptide-MHC complexes, T cell activation during antigen presentation requires a second signal. CD86, identified as a dendritic cell gene by our algorithm is an essential co-stimulatory molecule that delivers this second signal and is also a marker of dendritic cell maturation. Also in this list are TNFSF9 and TNFSF4, two cytokines that play a role in antigen presentation between dendritic cells and T lymphocytes.

We have also identified 157 genes that are more likely to be induced in macrophages than in dendritic cells. Among these genes, FNGR1 is important for macrophages to detect interferon-gamma (also known as type II interferon), a key activating cytokine of macrophages. HMGB1, a chromatin structural protein, is believed to be involved in inflammation and sepsis. Another interesting gene is ADAM12, which is from a family of proteinases that are likely involved in tissue remodeling/wound healing by macrophages.

## 5 Conclusions and Future Work

By combining expression experiments across species, cell types and bacteria type we were able to obtain a core set of innate immune response genes. The set we identified contained many of the known key players in this response and also included novel predictions. We have also identified unique signatures for macrophages and dendritic cells leading to insights regarding the set of processes activated in each of these cells types as part of the response.

While our method assumes that homologous genes share similar functions, it is still sensitive to the observed expression profiles. Thus, if two homologs display different expression patterns they would be assigned different labels. Still, homology information is a very useful feature for most genes. Relying on homology information we were able to drastically improve the recovery of the correct set of genes.

While we have focused here on immune response, our method is general and can be applied to other diseases or conditions. We would like to further explore the lists derived by our method to determine the interactions and mechanisms leading to the activation of these genes in the cells they were assigned to. We would also like to expand our method so that it can better utilize the temporal information available in the microarray data. An additional area to explore is to incorporate other sources of

information in the construction of the weight matrix. For example, it would be interesting to consider protein domains in addition to sequence similarity when creating the weight matrix.

## References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410 (1990)
2. Bandyopadhyay, S., Sharan, R., Ideker, T.: Systematic identification of functional orthologs based on protein network comparison. *Genome Res.* 16, 428–435 (2006)
3. Bishop, C.M.: *Pattern Recognition and Machine Learning*, pp. 383–392. Springer, Heidelberg (2006)
4. Bratt, J., Palmblad, J.: Cytokine-induced neutrophil-mediated injury of human endothelial cells. *J. Immunol.* 159, 912–918 (1997)
5. Chaussabel, D., Semnani, R.T., McDowell, M.A., Sacks, D., Sher, A., Nutman, T.B.: Unique gene expression profiles of human macrophages and dendritic cells to phylogenetically distinct parasites. *Blood*. 202, 672–681 (2003)
6. Deng, M., Chen, T., Sun, F.: Integrated probabilistic model for functional prediction proteins. *J. Comput. Biol.* 11, 435–465 (2004)
7. Detweiler, C.S., Cunanan, D.B., Falkow, S.: Host microarray analysis reveals a role for the *Salmonella* response regulator *phoP* in human macrophage cell death. *Proc. Natl. Acad. Sci., USA* 98, 5850–5855 (2001)
8. Draper, D.W., Bethea, H.N., He, Y.W.: Toll-like receptor 2-dependent and -independent activation of macrophages by group B streptococci. *Immunol. Lett.* 102, 202–214 (2006)
9. Enright, A.J., van Dongen, S., Ouzounis, C.A.: An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.* 30, 1575–1584 (2002)
10. Eppig, J.T., Bult, C.J., Kadin, J.A., Richardson, J.E., Blake, J.A., the members of the Mouse Genome Database Group: The Mouse Genome Database (MGD): from genes to mice—a community resource for mouse biology. *Nucleic Acids Res.* 33, D471–D475 (2005)
11. Ernst, J., Bar-Joseph, Z.: STEM: a tool for the analysis of short time series gene expression data. *BMC Bioinformatics* 7, 191 (2006)
12. Fearon, D.T., Locksley, R.M.: The instructive role of innate immunity in the acquired immune response. *Science* 272, 50–54 (1996)
13. Granucci, F., Vizzardelli, C., Pavelka, N., Feau, S., Persico, M., Virzi, E., Rescigno, M., Moro, G., Ricciardi-Castagnoli, P.: Inducible *IL-2* production by dendritic cells revealed by global gene expression analysis. *Nat. Immunol.* 2, 882–888 (2001)
14. Grote, M.J., Huckle, T.: Parallel Preconditioning with Sparse Approximate Inverses. *SIAM J. Sci. Comput.* 18, 838–853 (1997)
15. Hoffmann, J.A., Kafatos, F.C., Janeway Jr., C.A., Ezekowitz, R.A.B.: Phylogenetic perspectives in innate immunity. *Science* 284, 1313–1318 (1999)
16. Hoffmann, R., van Erp, K., Trulzsch, K., Heesemann, J.: Transcriptional responses of murine macrophages to infection with *Yersinia enterocolitica*. *Cell Microbiol.* 6, 377–390 (2004)
17. Huang, Q., Liu, D., Majewski, P., Schulte, L.C., Korn, J.M., Young, R.A., Lander, E.S., Hacohen, N.: The Plasticity of Dendritic Cell Responses to Pathogens and Their Components. *Science* 294, 870–875 (2001)

18. Jenner, R.G., Young, R.A.: Insights into host responses against pathogens from transcriptional profiling. *Nat. Rev. Microbiol.* 3, 281–294 (2005)
19. Keller, G., Snodgrass, R.: Life span of multipotential hematopoietic stem cells in vivo. *J. Exp. Med.* 171, 1407–1418 (1990)
20. Kelley, J., Bono, B.D., Trowsdale, J.: IRIS: a database surveying known human immune system genes. *Genomics* 85, 503–511 (2005)
21. Lammers, K.M., Brigidi, P., Vitali, B., Gionchetti, P., Rizzello, F., Caramelli, E., Matteuzzi, D., Campieri, M.: Immunomodulatory effects of probiotic bacteria DNA: IL-1 and IL-10 response in human peripheral blood mononuclear cells. *FEMS Immunol Med. Microbiol.* 22, 165–172 (2003)
22. Lang, R., Patel, D., Morris, J.J., Rutschman, R.L., Murray, P.J.: Shaping Gene Expression in Activated and Resting Primary Macrophages by IL-10. *J. Immunol.* 169, 2253–2263 (2002)
23. Lee, H.C., Goodman, J.L.: Anaplasma phagocytophilum causes global induction of antiapoptosis in human neutrophils. *Genomics* 88, 496–503 (2006)
24. Letovsky, S., Kasif, S.: Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics* 19(suppl. 1), i197–i204 (2003)
25. Liu, M., Liberzon, A., Kong, S.W., Lai, W.R., Park, P.J., Kohane, I.S., Kasif, S.: Network-Based Analysis of Affected Biological Processes in Type 2 Diabetes Models. *PLoS Genet.* 3, e96 (2007)
26. Lu, Y., Rosenfeld, R., Bar-Joseph, Z.: Identifying Cycling Genes by Combining Sequence Homology and Expression Data. *Bioinformatics* 22, e314–e322 (2006)
27. Lu, Y., Mahony, S., Benos, P.V., Rosenfeld, R., Simon, I., Breeden, L.L., Bar-Joseph, Z.: Combined Analysis Reveals a Core Set of Cycling Genes. *Genome Biol.* 8, R146 (2007)
28. Lukacs, N.W., Strieter, R.M., Chensue, S.W., Widmer, M., Kunkel, S.L.: TNF-alpha mediates recruitment of neutrophils and eosinophils during airway inflammation. *J. Immunol.* 154, 5411–5417 (1995)
29. McCaffrey, R.L., Fawcett, P., O’Riordan, M., Lee, K.-D., Havell, E.A., Brown, P.O., Portnoy, D.A.: From the Cover: A specific gene expression program triggered by Grampositive bacteria in the cytosol. *Proc. Natl. Acad. Sci. USA* 101, 11386–11391 (2004)
30. Mignon, C., Okada, A., Mattei, M.G., Basset, P.: Assignment of the human membrane-type matrix metalloproteinase (MMP14) gene to 14q11-q12 by in situ hybridization. *Genomics* 28, 360–361 (1995)
31. Mizutani, H., Schechter, N., Lazarus, G., Black, R.A., Kupper, T.S.: Rapid and specific conversion of precursor interleukin 1 beta (IL-1 beta) to an active IL-1 species by human mast cell chymase. *J. Exp. Med.* 174, 821–825 (1991)
32. Nau, G.J., Richmond, J.F.L., Schlesinger, A., Jennings, E.G., Lander, E.S., Young, R.A.: Human macrophage activation programs induced by bacterial pathogens. *Proc. Natl. Acad. Sci. USA* 99, 1503–1508 (2002)
33. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. *Comput. J.* 7, 308–313 (1965)
34. Sun, Z., Andersson, R.: NF-kappaB activation and inhibition: a review. *Shock* 18, 99–106 (2002)
35. Valbuena, G., Bradford, W., Walker, D.H.: Expression analysis of the T-cell-targeting chemokines CXCL9 and CXCL10 in mice and humans with endothelial infections caused by rickettsiae of the spotted fever group. *Am. J. Pathol.* 163, 1357–1369 (2003)
36. Van Erp, K., Dach, K., Koch, I., Heesemann, J., Hoffmann, R.: Role of strain differences on host resistance and the transcriptional response of macrophages to infection with *Yersinia enterocolitica*. *Physiol. Genomics* 25, 75–84 (2006)

37. Wolpe, S.D., Davatelas, G., Sherry, B., Beutler, B., Hesse, D.G., Nguyen, H.T., Moldawer, L.L., Nathan, C.F., Lowry, S.F., Cerami, A.: Macrophages secrete a novel heparin-binding protein with inflammatory and neutrophil chemokinetic properties. *J. Exp. Med.* 167, 570–581 (1988)
38. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. In: Exploring Artificial Intelligence in the New Millennium, pp. 236–239. Morgan Kaufmann Publishers Inc., San Francisco (2003)
39. Zhu, X.: Semi-Supervised Learning with Graphs. Ph.D. Thesis, Carnegie Mellon University, CMU-LTI-05-192 (2005)

## Appendix

### 1. Proof of Eq (2)

In this section we prove that for any positive numbers  $a$ ,  $b$ , and  $c$  satisfying  $a + b + c = 1$ , there exist real numbers  $\mu$  and  $\sigma^2$  such that if  $y \sim N(\mu, \sigma^2)$ , then

$$\Pr(y \leq -1) = a, \Pr(-1 < y \leq 1) = b, \Pr(y > 1) = c. \quad (5)$$

Here  $N(\mu, \sigma^2)$  denotes the Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ .

**Proof:** Let  $\Phi$  denote the cumulative density function of the standard Gaussian distribution  $N(0,1)$ . Let  $u = 2 / (\Phi^{-1}(a+b) - \Phi^{-1}(a))$ ,  $v = 1 + u \cdot \Phi^{-1}(a)$ . Then  $N(v, u^2)$  satisfies the conditions in Eq (5) as required to prove the claim we made for Eq. (2).

### 2. Belief propagation for Gaussian Random Fields

In Section 3.1, we describe the belief propagation algorithm on a Gaussian random field. We give the message passing and belief update rules in Eq (3) and (4). Because each variable in a GRF follows a Gaussian distribution, these equations can be simplified and lead to very efficient update rules.

Note that the operations carried out in Eq (3) and (4) are multiplication of univariate Gaussian distributions and marginalization of bivariate Gaussian distributions. For multiplication of univariate Gaussian distributions with mean  $\mu_i$  and variance  $\sigma_i^2$ , we have

$$\prod_i \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\} \propto \exp\left\{-\frac{(x-(\sum q_i \mu_i)/\sum q_i)^2}{2(\sum q_i)^{-1}}\right\}$$

where  $q_i = 1/\sigma_i^2$ . The resulting product is a Gaussian distribution with the following mean and variance

$$\begin{aligned} \mu &\leftarrow (\sum q_i \mu_i) / \sum q_i \\ \sigma^2 &\leftarrow (\sum q_i)^{-1} \end{aligned}$$

We can get belief update rules for Eq (4) by substituting  $\mu_i$  and  $\sigma_i^2$  with the mean and variance of  $m_{ki}(y_i)$  and  $\psi_k(y_i)$ , where  $k$  belongs to the set of the neighbors of  $i$  excluding  $j$ .

Next we derive the rules for marginalization of bivariate Gaussian distributions in Eq (3). Let

$$\begin{aligned} f_{ij}(y_i) &\stackrel{\text{def}}{=} \psi_i(y_i) \prod_{k \in N(i) \setminus j} m_{ki}(y_i) \sim N(\nu_{ij}, \rho_{ij}^2) \\ \psi_{ij}(y_i, y_j) \cdot f_{ij}(y_i) &\sim N((\mu_i, \mu_j), \Sigma_{ij}) \end{aligned} \quad (6)$$

and

$$\Sigma_{ij}^{-1} = \begin{pmatrix} p_{ii} & p_{ij} \\ p_{ij} & p_{jj} \end{pmatrix}$$

We can compute the mean and variance of message  $m_{ij}(y_j)$ , which is the result of marginalization of the bivariate Gaussian distribution in Eq (3), by matching the left-hand side (LHS) and right-hand side (RHS) of Eq (6). By expanding the exponent of the RHS of Eq (6), we get

$$\begin{aligned} & -\frac{1}{2} \begin{pmatrix} y_i - \mu_i & y_j - \mu_j \end{pmatrix} \begin{pmatrix} p_{ii} & p_{ij} \\ p_{ij} & p_{jj} \end{pmatrix} \begin{pmatrix} y_i - \mu_i \\ y_j - \mu_j \end{pmatrix} \\ &= -\frac{1}{2} [p_{ii}y_i^2 + p_{jj}y_j^2 + 2p_{ij}y_iy_j + \dots] \end{aligned} \quad (7)$$

Substituting and expanding the exponent of the LHS of Eq (6), we get

$$-\frac{1}{2} [\alpha_{ij} + r_{ij}] y_i^2 + \alpha_{ij} y_j^2 + \text{sign}(w_{ij}^*) \cdot 2\alpha_{ij} y_i y_j + \dots \quad (8)$$

where

$$\alpha_{ij} = 2\lambda |w_{ij}^*| \quad \text{and} \quad r_{ij} = 1/\rho_{ij}^2$$

Equating (7) and (8), we can get the following update rules for computing the mean and variance of message  $m_{ij}(y_j)$

$$\begin{aligned} \mu_j &= \text{sign}(w_{ij}^*) \cdot \nu_{ij} \\ \sigma_j^2 &= \frac{\alpha_{ij} + r_{ij}}{\alpha_{ij}r_{ij}} = \frac{1}{r_{ij}} + \frac{1}{\alpha_{ij}} \end{aligned}$$

### 3. Identification of Common Response Genes

Following convergence of our inference algorithm for the GRF model, we obtained the posterior probability of a gene participating in immune response, for each cell type in each species, for both Gram-positive and Gram-negative infections. Using these posteriors we constructed the list of common response genes by selecting ortholog pairs whose posterior probabilities are higher than 0.5 in all cells, bacteria and species. These genes are up-regulated in response to bacterial infections in all types of experiments we looked at.

# Haplotype Inference in Complex Pedigrees

Bonnie Kirkpatrick<sup>1</sup>, Javier Rosa<sup>2</sup>, Eran Halperin<sup>3</sup>, and Richard M. Karp<sup>4</sup>

<sup>1</sup> Computer Science Dept, University of California, Berkeley  
`bbkirk@eecs.berkeley.edu`

<sup>2</sup> Computer Science Dept, Rutgers, The State University of New Jersey,  
New Brunswick

<sup>3</sup> School of Computer Science and the Dept. of Biotechnology, Tel-Aviv University,  
and the International Computer Science Institute, Berkeley  
`heran@icsi.berkeley.edu`

<sup>4</sup> Computer Science Dept, University of California, Berkeley and the International  
Computer Science Institute  
`karp@icsi.berkeley.edu`

**Abstract.** Despite the desirable information contained in complex pedigree datasets, analysis methods struggle to efficiently process these datasets. The attractiveness of pedigree data sets is their power for detecting rare variants, particularly in comparison with studies of unrelated individuals. In addition, rather than assuming individuals in a study are unrelated, knowledge of their relationships can avoid spurious results due to confounding population structure effects. However, a major challenge for the applicability of pedigree methods is the ability handle complex pedigrees, having multiple founding lineages, inbreeding, and half-sibling relationships.

A key ingredient in association studies is imputation and inference of haplotypes from genotype data. Existing haplotype inference methods either do not efficiently scale to complex pedigrees or their accuracy is limited. In this paper, we present algorithms for efficient haplotype inference and imputation in complex pedigrees. Our method, PhyloPed, leverages the perfect phylogeny model, resulting in an efficient method with high accuracy. In addition, PhyloPed effectively combines the founder haplotype information from different lineages and is immune to inaccuracies in prior information about the founders.

## 1 Introduction

Both genetic and environmental factors affect the etiology of complex conditions such as cancer or Alzheimer's disease. In an attempt to reveal the genetic factors of these conditions, many researchers use pedigree studies, in which the genomes of a set of related cases and controls are compared. Regions in which the allelic distribution of the cases differs from the expected distribution given the pedigree relationships are suspect for direct or indirect involvement in the disease mechanism.

Current studies focus on analyzing *single nucleotide polymorphisms* (SNPs), which are mutations that occurred once in history and propagated through the

population. Common SNPs are usually bi-allelic with both of the alleles appearing in at least 5% of the population. Current genotyping technologies allow us to genotype a set of a million SNPs spread across the whole genome for less than a thousand dollars per person. Thus, large scale studies, involving hundreds of thousands of SNPs and thousands of individuals, are feasible. Indeed, if the genealogical data exists, it is possible to obtain a pedigree for thousands of individuals (e.g. the Huddertite data with more than 1600 individuals [1], and animal breeding data [17]).

Although many current study designs employ population case-control designs, with unrelated individuals, there are substantial advantages to using pedigree study designs. Designing population-based studies may be problematic due to confounding effects such as population substructure and heterogeneity within the case population, (i.e., the set of cases consists of a few subsets, where each subset has a different disease that is manifested in the same way, but is genetically and probably biologically different). These phenomena may reduce power or lead to false discoveries. However, incorporating related individuals into association studies bypasses these problems by correcting for sources of heterogeneity and population substructure. Knowing relationships between family members, as in a pedigree, can assist in obtaining a more accurate estimate of each individual's haplotypes, which are sequences of alleles on a chromosome that were inherited from the same ancestor. Haplotypes can be used for imputation of unobserved genotypes or alleles, which have been useful in finding new associations [2]. Furthermore, imputation in family-based association studies has been shown to increase power [53].

The theoretical usefulness of large pedigree datasets is diminished in practice by computational issues. Pedigree analysis is known to be NP-hard [16], and all known algorithms have exponential running time. The classical trade-off for running time is between being exponential in the number of loci (and linear in the number of individuals) or exponential in the number of individual (and linear in the number of loci), for example, the Elston-Stewart and Lander-Green algorithms respectively [8][14]. More recent work, in the form of Superlink [10], heuristically optimizes this trade-off. Among the MCMC approaches to pedigree analysis, blocked Gibbs sampling has been successfully applied to large pedigrees [13][18]. Blocked Gibbs samplers are a generalization of Gibbs samplers where a set of variables is updated at each step rather than a single variable. These samplers elegantly deal with inbreeding by conditioning and rely on the random steps of the Markov Chain to propagate the effect of inbreeding through the graphical model. Convergence occurs quickly in practice. However correctness is typically proved via irreducibility of the state space for a particular problem instance, and these proofs themselves may correspond to difficult computational problems [4][13].

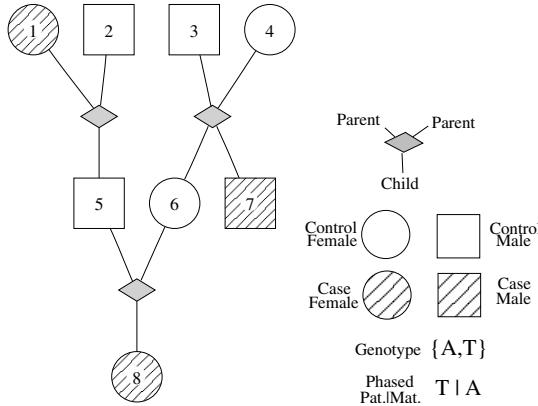
In this paper, we consider a special case of the pedigree haplotyping problem for complex pedigrees, having multiple lineages. Specifically, we are interested in regions of the genome that are sufficiently linked that there is little evidence of recombination during pedigree meiosis. Further, there are two cases for these regions. First, if there is little evidence of ancestral recombinations or recurrent

mutations in the founding haplotypes, the perfect phylogeny model [1] would apply to the pedigree haplotypes. The perfect phylogeny model has been shown to be realistic as long as the studied region is physically short [6,7,9]. Second, if there is evidence of ancestral recombinations in the region, then ancestral recombinations must be allowed, and the founding haplotypes are not restricted to a perfect phylogeny. We make no other assumptions about recombination rates or founder allele frequencies. These two cases allow us to make simplifying assumptions and allow efficient computation over large and complex pedigrees without compromising accuracy.

To solve the first case of the problem, we propose a blocked Gibbs sampler with running time polynomial in the number of SNPs and linear in the number of individuals. Roughly, PhyloPed, our method, chooses overlapping blocks of individuals that correspond to lineages in the pedigree. A single sampling step updates the haplotype assignments for all the individuals in the lineage of interest. The algorithm considers each lineage in turn, updates that lineage, and continues until convergence. PhyloPed begins the blocked Gibbs sampler at an initial state that is a feasible haplotype configuration that is compatible with the perfect phylogeny. In practice, the initial haplotype state can often be obtained quickly, though in the worst case, due to disallowing recombination, the running time may be exponential in the number of individuals. In the case that the founder haplotypes could not have come from a perfect phylogeny, PhyloPed reverts to the second case, without the perfect phylogeny, and runs the same blocked Gibbs sampler from an initial haplotype configuration with unrestricted founder haplotypes (with running time exponential in the number of SNPs). Furthermore, PhyloPed does not require knowledge of recombination rates or founder-allele frequencies. The perfect phylogeny allows more accurate haplotype inference, for a small number of SNPs.

## 2 Methods

We represent a pedigree on a set of individuals  $I$  as a directed graph having individuals as nodes (either circles or squares) and relationships indicated by edges and marriage nodes (solid diamonds, see Figure 1). The pedigree edges are usually *implicitly* directed, with the edges being directed downwards. Parent-child relationships are drawn with a vertical arrangement of nodes and edges, with edges from the parent down to a marriage node and from the marriage node down to the child (Fig. 1). The *founders* of the pedigree are individuals  $F \subset I$  whose parents are not represented in the graph. According to convention, assume that every non-founder has both their parents represented in the pedigree, so that every marriage node has two parents above and adjacent to it. For each of the  $M$  bi-allelic SNPs, every individual  $w$  has an unordered single-locus genotype  $g_w^m$  at SNP  $m$ . An individual with a fully observed genotype has a single set of possible alleles  $g_w^m \in \{\{0, 0\}, \{0, 1\}, \{1, 1\}\}$ . An individual with an unobserved or partially observed genotype has several possible sets of alleles  $g_w^m \in \{\{\{0, 0\}, \{0, 1\}, \{1, 1\}\}, \{\{0, 0\}, \{0, 1\}\}, \{\{0, 1\}, \{1, 1\}\}\}$ .



**Fig. 1.** Pedigree Example and Key

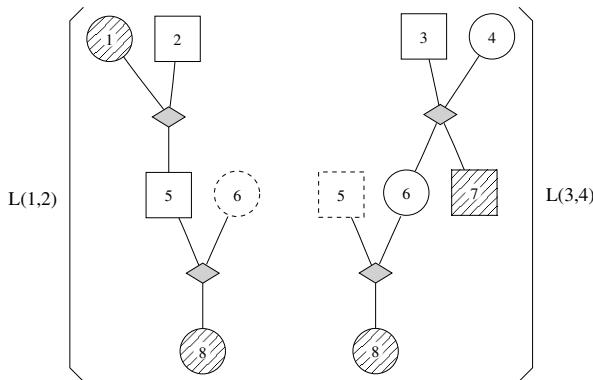
We denote a haplotype by a sequence of binary alleles,  $h \in \{0, 1\}^M$ , where  $M$  is the number of SNPs in a region of the genome. Let the  $m$ 'th allele of haplotype  $h$  be denoted  $h(m)$ . A pedigree state associates an ordered pair of haplotypes (or multi-locus genotype),  $s_w = (h_w, h'_w)$ , with each individual  $w \in I$ . The haplotypes of an individual are *consistent* with the observed genotypes provided that at each SNP  $m$ , all known alleles are represented in the haplotypes, meaning that  $\{h_w(m), h'_w(m)\} \subset g_w^m$ . Let  $C(w, s_w)$  be an indicator variable that is 1 when the haplotype state  $s_w$  is consistent with the observed genotypes. Let  $S(w)$  be the set of all haplotype states that are consistent with  $w$ 's observed genotype.

We assume that the  $M$  SNPs are tightly linked and are effectively unable to recombine when passed from one generation to the next. In other words, haplotypes are passed according to Mendelian inheritance. More precisely, we define an individual's haplotype state  $s_w$  as *non-recombinant* when the haplotype  $h_w$  inherited from the father,  $f(w)$ , exactly matches one of the father's haplotypes,  $h_w = h_{f(w)}$  or  $h_w = h'_{f(w)}$ , and similarly  $h'_w$ , inherited from the mother,  $m(w)$ , matches one of the mother's haplotypes.

Mendelian inheritance gives the probability that each of the father's (or mother's) haplotypes are inherited by the child:  $Pr[h_w = h_{f(w)} | h_{f(w)} \neq h'_{f(w)}] = 1/2$  and  $Pr[h_w = h_{f(w)} | h_{f(w)} = h'_{f(w)}] = 1$ . This assumption determines a family of probability distributions over states of the pedigree, given genotype data for some of the individuals. Our goal is to find the haplotypes that maximize the conditional distribution of pedigree states given genotype data for some individuals.

*Lineage Decomposition.* Rather than computing the joint distribution of haplotype assignments to all the founders, we decompose a complex pedigree into tree-like lineages. Roughly speaking, each lineage is a block of variable that will be updated in a single iteration of the blocked Gibbs sampler, and the lineages are not necessarily disjoint from each other.

A lineage is defined as follows. Let  $H(w)$  denote the set of children of node  $w$ . Founders  $p$  and  $q$  are called a *monogamous founding pair* (MFP) if and only



**Fig. 2.** These are the lineages for the pedigree in Fig. 1. The non-lineage parents are dashed, and individuals 6 and 5, respectively, are parents of individuals in the lineages  $L(1,2)$  and  $L(3,4)$ .

if  $H(p) = H(q)$  (i.e. there are no half-siblings of the founders' children). Assume that the pedigree contains only monogamous married pairs of founders. The *lineage of the monogamous founding pair*  $(p, q)$  is the induced subgraph of the pedigree that contains all the descendants of  $p$  and  $q$ . Formally, the lineage  $L(p, q)$  is a directed, acyclic graph that contains a source node for each  $p, q$  and a node for each descendant of  $p, q$ . This means that  $L(p, q)$  is the smallest subgraph of the given pedigree such that  $L$  contains both founders  $p, q$  and, if  $L$  contains a node  $w$ , then  $L$  contains the children of  $w$ . If a parent of  $w$  is not in  $L$  then that parent is called the *non-lineage* parent.

For example the lineages of the pedigree in Fig. 1 are shown in Fig. 2 as two distinct pedigree sub-graphs. Notice that parents of lineage members fall into four categories: 1) founders participating in the MFP, and 2) non-lineage founding parents, 3) non-lineage parents (non-founders who are descendants of another lineage) and 4) lineage descendants (descended from the MFP).

Our goal is to choose a haplotype state for the pedigree from the posterior distribution of haplotype states given the genotype data of the pedigree and assumptions about haplotype sharing between lineages. A side effect of this is that PhyloPed infers all missing alleles, including haplotypes (and genotypes) for ungenotyped individuals. To find a haplotype state, we consider each lineage separately and calculate the distribution of haplotype states for the monogamous pairs of founders, given the genotype data of their descendants and probabilistic assumptions about the states of the non-lineage founders and non-lineage parents. The calculation proceeds in three phases:

1. Find a consistent, non-recombinant state for the pedigree. If there is such a state that is also compatible with some perfect phylogeny on all of the observed genotypes [119], choose that state (see Supplement). Otherwise, when the founder haplotypes require ancestral recombinations or back mutations, choose any consistent, non-recombinant state for the pedigree haplotypes.

2. Decompose the pedigree into lineages, as described above.
3. Iterate over the collection of lineages: first, compute the distribution for the MFP haplotypes conditioning on the genotypes in the lineage and conditioning on the current states of the non-lineage parents, and second, sample new haplotype states for the lineage descendants from the computed distribution.

*Inference for a Single Lineage.* To describe step 3 in detail, we need to establish a few assumptions. First, assume that we have a consistent, non-recombinant state for the pedigree (for details, see Supplement). Also, assume for the moment that there is a known prior probability for founder haplotypes,  $\alpha(h)$  for the  $2^m$  possible haplotypes. Each time inference is performed on a lineage, the algorithm removes inbreeding loops by randomly choosing an individual to condition on. This is done by successively finding the oldest inbred descendant (whose parents are not inbred) and flipping a coin to choose which parent will be designated the non-lineage parent for the duration of the iteration.

For a single, non-inbred lineage, we can compute the probability of the MFP haplotypes by conditioning on 1) the haplotype assignments of the non-lineage parents, 2) the genotypes, and 3) the prior probability  $\alpha$ . The child of a non-lineage parent inherits either one of the two equally-likely non-lineage haplotypes, if the non-lineage parent has a haplotype assignment, or one of the  $2^m$  possible founder haplotypes drawn from  $\alpha$ , if the non-lineage parent is an ungenotyped founder. The prior and transmission probabilities yield a tree-like graphical model from which to learn the MFP haplotype distribution. For the lineage  $L(p, q)$ , let  $\phi_{p,q}(i, j, k, l)$  be the marginal probability of the haplotype assignment  $(i, j)$  to  $p$  and  $(k, l)$  to  $q$  conditioned on the genotypes in the lineage and the haplotype assignments of the non-lineage parents. This is a marginal probability, because it is computed by summing over possible haplotype assignments for lineage descendants.

Some fairly standard bottom-up dynamic-programming equations yield the MFP marginal  $\phi_{p,q}(i, j, k, l)$  and incomplete marginals, or messages, for the lineage descendants (see the peeling algorithm in [15]). The descendant marginals are incomplete, because they are computed by summing only over possible haplotype assignments to their descendants (rather than summing also over possible assignments to their ancestors), and are conditioned only on the genotypes of their descendants (rather than all of the lineage genotypes). For an MFP child,  $r$ , with two lineage haplotypes  $(i, j)$ , define  $\phi_r(i, j)$  as the incomplete marginal probability of  $r$  having haplotypes  $(i, j)$  conditioned on the genotypes of  $r$ 's descendants. Similarly, for all other lineage descendants, with one lineage haplotype, define  $\phi_w(i)$  as the incomplete marginal probability of  $w$  having lineage haplotype  $i$  conditioned on the genotypes of  $w$ 's descendants (see the Supplement for equations).

When considering all possible haplotype assignments, computation of these conditional probabilities takes time  $O(N^4 \cdot L)$  where  $N = 2^m$  is the number of possible haplotypes and  $L$  is the number of individuals in the lineage. Recall that  $m$  is small and the computation is feasible, because all the SNPs are in a short region of the genome and are in linkage disequilibrium. In cases where

only perfect phylogeny haplotypes are considered, the running time is reduced to  $O((m + 1)^4 L)$ . This follows from the fact that a perfect phylogeny contains at most  $m + 1$  haplotypes.

In order to update the haplotype state of a lineage, we use a top-down random propagation algorithm that chooses a new pair of haplotypes for each individual in the lineage (similar to the random propagation algorithm described in [15]). Random propagation allows us to choose haplotype assignments for each person from the correct, or complete, marginal haplotype distribution for that individual. For the pair of founders,  $p, q$ , haplotypes  $(i, j, k, l)$  are chosen proportional to  $\alpha(i)\alpha(j)\alpha(k)\alpha(l)\phi_{p,q}(i, j, k, l)$ . Children,  $r$ , of the MFP are randomly assigned haplotypes  $(h_r, h'_r) \in \{(i, k), (i, l), (j, k), (j, l)\}$ , conditional on the MFP haplotype assignment  $(i, j, k, l)$ , with probability proportional to  $\phi_r(h_r, h'_r)$ . All other lineage descendants,  $w$ , are given a lineage haplotype  $h_w$  conditional on the haplotypes  $(h_{l(w)}, h'_{l(w)})$  of their lineage parent  $l(w)$ . So,  $Pr[h_w = h_{l(w)}]$  is proportional to  $\phi_w(h_{l(w)})$ . And the non-lineage haplotypes  $h'_w$  is chosen from the set  $\{h_{n(w)}, h'_{n(w)}\}$  of haplotypes for the non-lineage parent  $n(w)$  and probability proportional to  $\alpha(h'_w)C(w, h_w, h'_w)$ . The random propagation scheme is also accomplished in time  $O(N^4 L)$ , except when perfect phylogeny haplotypes are known, making the running time  $O((m + 1)^4 L)$ .

*Inference for Multiple Lineages.* We now extend our algorithm to consider several monogamous founding pairs simultaneously. We no longer assume that there is a fixed  $\alpha$  distribution or that the haplotype states never change. Instead, we use an iterative process that computes a new haplotype distribution  $\alpha^t$  at each iteration  $t$  and maintains a consistent, non-recombinant haplotype state for all the individuals in the pedigree. For each iteration,  $t$ , consider each MFP  $(p, q)$  and its lineage  $L(p, q)$ :

- Given the previous estimate of  $\alpha^{t-1}$ , perform the bottom-up dynamic programming calculation to compute  $\phi_{p,q}^t(i, j, k, l)$ ,  $\phi_r^t(i, j)$  and  $\phi_w^t(i)$  for the MFP  $(p, q)$ .
- Use  $\alpha^{t-1}$  together with the various  $\phi^t$  probabilities in the random propagation scheme to sample a new haplotype state for the individuals in the lineage.

After obtaining an updated  $\phi_{p,q}^t(i, j, k, l)$  for each MFP  $(p, q)$ , compute the updated prior distribution as the marginal average  $\alpha^t(h) \propto \sum_{(p,q)} m_{p,q}^t(i) + m_{p,q}^t(j) + m_{p,q}^t(k) + m_{p,q}^t(l)$  where the marginal  $m_{p,q}^t(i) = \sum_j \sum_k \sum_l \phi_{p,q}^t(i, j, k, l)$  and similar definitions apply for  $m_{p,q}^t(j)$ ,  $m_{p,q}^t(k)$ , and  $m_{p,q}^t(l)$ . The iterations continue until the  $l_1$  deviation between  $\alpha^t$  and  $\alpha^{t-1}$  falls below a pre-determined threshold. Clearly the running time of our method depends on the number of iterations until convergence. In practice, the  $l_1$  deviation of the  $\alpha^t$  estimates drop rapidly and most of the blocks in Fig. 3 converged in roughly 6-8 iterations (see Supplement).

*Correctness.* We have described a blocked Gibbs sampling scheme where in each iteration, the updated block is a non-inbred subgraph of a pedigree

lineage. Each update step uses a mixture of bottom-up recursion and top-down sampling to update the haplotype assignments in each block. A Markov Chain employing this update algorithm will converge to the correct posterior probability distribution when the haplotype states of the pedigree form an irreducible state space. In each update iteration, the haplotypes for the lineage individuals are updated conditional on the haplotypes assigned to the non-lineage parents, while the haplotypes of the non-lineage parents and all other pedigree individuals are unchanged. If the unchanged haplotypes are drawn from the stationary distribution, then after an update, all the haplotypes together represent a sample from the stationary distribution. This would be true for any blocking scheme, but we have chosen the lineage blocking scheme for ease of computation.

### 3 Results

*Pedigree Simulations.* In order to test the accuracy of our method, we simulated a set of pedigrees with their corresponding haplotypes. Given a pedigree, founder haplotypes were generated uniformly at random from the phased HapMap CEU haplotypes for Chromosome 1. We considered only common SNPs (with minor allele frequencies at least 0.05). We performed multiple trials, where each trial consisted of a distinct sample of SNPs chosen to have a specific density along the genome. This allowed us to vary the mean physical distance between neighboring SNPs. Each sample of SNPs was arbitrarily partitioned into non-overlapping blocks of a fixed length for haplotype inference.

The non-founders were generated in successive generations using Poisson-distributed recombinations (without interference), where the recombination rate was a function of the physical distance, such that there is an average of two recombinations on the length of Chromosome 1. Considering each non-founder in turn, we obtained one haplotype from each parent by uniformly choosing one of the parental haplotypes to provide the allele for the first SNP. Alleles for successive SNPs were chosen either to be non-recombinant or recombinant according to the recombination rate. We refer to the complete simulation output (of phased haplotypes) as the *gold-standard data*.

We chose pedigrees with fixed structure. For each pedigree we fixed the number and set of individuals to be genotyped in the data input to each of the phasing algorithms (and removed the phase information for all of the ungenotyped individuals).

- L1** 10 copies of a 20-individual family with 1 lineage and exactly 13 genotyped individuals (1000 blocks of 3 SNPs with 11 kbp between SNPs)
- S1** single family with 10 lineages and 59 individuals, exactly 24 of them being genotyped on 1000 blocks of 3 SNPs.
- M1** 5 copies of the family from S1, with exactly 24 individuals genotyped in each family (1000 blocks with 3 SNPs).

**M2** 10 copies of a 10-individual family with 2 lineages and exactly 5 genotyped individuals (10,000 blocks of 3 SNPs).

**H1** single 16-individual, 2-lineage pedigree with half-siblings and exactly 9 genotyped individuals on 300 blocks of 5 SNPs.

*Comparison.* We compared our approach to two others, Merlin [5] and Superlink [10]. Both Merlin and Superlink perform a maximum-likelihood calculation on a similar graphical model of inheritance in a pedigree, where recombination rates and founder allele frequencies are given as fixed parameters of the model. However, Merlin employs a different elimination order for the EM algorithm than does Superlink and has an option for non-recombinant haplotype inference (this option was not used here, because it seemed to make little difference to inference accuracy). PhyloPed uses a graphical model of inheritance that is similar to that used by Merlin and Superlink but does not require the founder allele frequencies or recombination rates.

The input data consisted of the pedigree relationships and the genotype data for only the typed pedigree members. Merlin and Superlink were additionally provided with the correct recombination rates and with either an uninformative prior for the founder alleles or the perfect prior (i.e., the correct allele frequencies). Every phasing program was run on consecutive, non-overlapping blocks of  $k$  SNPs, and all programs ran on the same blocks. The output of each of the phasing programs was compared to the gold-standard data, and again the comparison used the same  $k$ -sized blocks. In cases where phasing programs provided a list of possible phasings, the first phasing was tested for accuracy. Accuracy was measured as the percentage of haplotype assignments in the phase estimate that matched the haplotypes in the gold-standard haplotype data. Notice that in this definition of accuracy the parental origin of the haplotype is irrelevant. Notice also that if the assumptions of PhyloPed are not satisfied, meaning that a particular family required recombinant haplotypes, then PhyloPed produced no estimate, and we conservatively chose to penalize our method by scoring the lack of a prediction as zero accuracy.

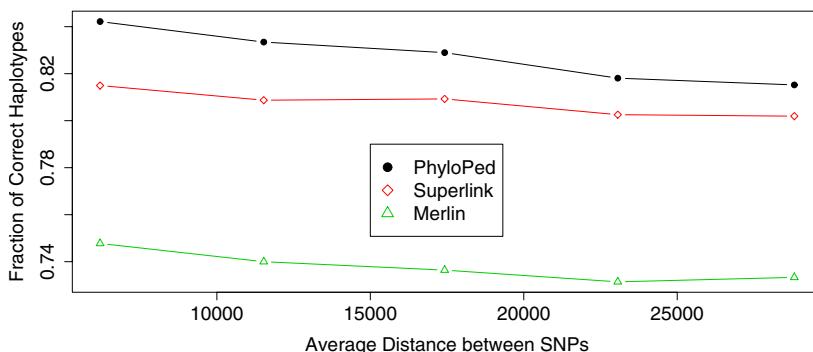
*Simple vs Complex Pedigrees.* For the single-lineage pedigree L1, we simulated blocks of size  $k = 3$  with the average physical distance between SNPs being 11 kbp. All methods estimated haplotypes with similar accuracy (Table 1, row L1). This suggests that the models have few practical differences on simple pedigrees.

For multi-lineage pedigrees S1, M1, M2, and H1, we see that PhyloPed outperforms the other methods (Table 1, rows S1, M1, and H1, and Fig. 3). Most of these results were generated for blocks with  $k = 3$  SNPs, because larger blocks were infeasible for Merlin. However, pedigree H1 was simulated with  $k = 5$  SNPs, and still PhyloPed outperforms the others.

*Violations of Assumptions.* We consider two violations of the assumptions for the three methods. First, we consider the performance of the three methods for different physical distances between SNPs in the block (resulting in a range of recombination rates). PhyloPed consistently outperforms Superlink and Merlin even as

**Table 1. Average accuracy and standard deviation.** In all cases, PhyloPed dramatically outperforms Merlin. PhyloPed is substantially better than Superlink, when given a non-informative perfect prior. When given the perfect prior, Superlink performs no better than PhyloPed. In cases where Superlink and PhyloPed have comparable performance, we see that the uninformative prior particularly hurts Superlink’s accuracy. Merlin was unable to execute S1 and M1 due to running time.

Pedigree	Method	Perfect Prior		Uninformative Prior	
		Avg	Std-Dev	Avg	Std-Dev
L1	PhyloPed	<b>0.867</b>	0.030	<b>0.867</b>	0.030
	Merlin	0.855	0.018	0.857	0.018
	Superlink	0.836	0.034	0.819	0.023
S1	PhyloPed	<b>0.809</b>	0.065	<b>0.809</b>	0.065
	Superlink	0.796	0.064	0.642	0.066
M1	PhyloPed	<b>0.808</b>	0.060	<b>0.808</b>	0.060
	Superlink	0.795	0.058	0.636	0.058
H1	PhyloPed	<b>0.816</b>	0.161	<b>0.816</b>	0.161
	Merlin	0.750	0.138	0.761	0.124
	Superlink	0.799	0.116	0.717	0.148



**Fig. 3. Accuracy Against Recombination Rate.** This plot shows results of 10,000 blocks for M2 the 2-lineage, 10-individual family. The accuracy of each method was computed for different physical distance between neighboring SNPs.

the recombination rate increases (Fig. 3). Second, it is possible for the founder allele frequencies to be unknown, even while the recombination rates may be known. We provide both Superlink and Merlin with uninformative founder allele frequencies (i.e. frequency 0.5 for all alleles). In this scenario, Merlin performs comparable to when it is given a perfect prior, but Superlink’s accuracy decreases dramatically.

## 4 Discussion

We have introduced PhyloPed which leverages the population genetics of the founders to produce superior haplotype estimates for multi-lineage pedigrees.

Specifically, PhyloPed assumes that the founder haplotypes are drawn from a perfect phylogeny and that haplotypes are inherited without recombination in the pedigree. As we have shown, this approach works very well for short regions with dense SNPs.

In addition to the perfect phylogeny model, there are several other reasons that PhyloPed outperforms other methods. Intuitively, Occam's razor suggests that our method would be preferable on blocks having little recombination. Assuming no recombination provides not only fewer phasing options to consider but also fewer parameters and less over-fitting. PhyloPed requires no prior information, either for the recombination rates or for the founder allele frequencies, which avoids the possibility that an inaccurate prior might mislead our algorithm.

Many factors influence the accuracy of haplotype estimation, including the complexity of the pedigree, the number and relationships of genotyped individuals, and the number of linked SNPs. The number of genotyped individuals in the pedigree and their relationships with the other pedigree members influences the number of constraints available for haplotype estimation. Typically, having genotypes for more individuals yields better haplotype estimates. Similarly, simultaneous phasing of larger numbers of linked SNPs can reveal more haplotype information, provided that the pedigree is not so large that the computational burden is infeasible. This paper has focused on inference in deep and complex pedigrees and partitioned the genome into blocks before phasing. In order to properly treat the whole genome, future research should consider partitioning schemes and methods for producing whole genome haplotype estimates from the estimates for each partition. One possible approach is using an HMM, similar to some of the tag SNP research for unrelated individuals [12].

Pedigrees should not be made unnecessarily complex. Multiple-lineage pedigrees are only useful in the case where each founding lineage provides information about either the phenotype or the relatedness of genotyped individuals. For example, estimation of haplotypes in a nuclear family whose members are genotyped and phenotyped would not benefit from the introduction of grandparents whose additional degrees of freedom provide no additional constraints on the haplotypes or phenotypes. However, if a pair of grandparents are the common ancestors of this nuclear family and another genotyped family, then the grandparents' presence in the pedigree (along with the additional family) would provide useful constraints.

Within 10 years, it is plausible that cost-effective sequencing methods will provide haplotypes for samples. However, the availability of haplotypes for some individuals in the pedigree does not obviate the need for phasing the unsampled individuals. Algorithms such as the one presented here provide consistent resolutions for the ancestry of each haplotype and yield haplotype assignments for pedigree members whose DNA is unavailable. Notice that phasing the unsampled individuals is (nearly) equivalent to the problem of finding the recombinations that produced the observed haplotypes.

There are a number of open problems, and further work is needed to take full advantage of the information provided by both genotype and sequence data. For

instance, other population genetic models could be applied to the founder haplotypes. These models have the added benefit of inferring which recombinations occurred in the pedigree versus in the ancestral haplotypes. Another important question for sequencing data is how best to take advantage of known information about identity by descent.

**PhyloPed Implementation and Supplementary Materials.** Available at:  
<http://phyloped.icsi.berkeley.edu/phyloped/>

**Acknowledgments.** We thank the reviewers for their insightful comments. B.K. was supported under a National Science Foundation Graduate Research Fellowship. E.H. and R.M.K. were supported by NSF grant IIS-0513599. Eran Halperin is a faculty fellow of the Edmond J. Safra Bioinformatics program at Tel Aviv University.

## References

1. Abney, M., Ober, C., McPeek, M.S.: Quantitative-trait homozygosity and association mapping and empirical genomewide significance in large, complex pedigrees: Fasting serum-insulin level in the hutterites. *The American Journal of Human Genetics* 70(4), 920–934 (2002)
2. Barrett, J.C., Hansoul, S., Nicolae, D.L., Cho, J.H., Duerr, R.H., Rioux, J.D., Brant, S.R., Silverberg, M.S., Taylor, K.D., Barmada, M.M., et al.: Genome-wide association defines more than 30 distinct susceptibility loci for crohn's disease. *Nature Genetics* 40, 955–962 (2008)
3. Burdick, J.T., Chen, W., Abecasis, G.R., Cheung, V.G.: In silico method for inferring genotypes in pedigrees. *Nature Genetics* 38, 1002–1004 (2006)
4. Cannings, C., Sheehan, N.A.: On a Misconception About Irreducibility of the Single-Site Gibbs Sampler in a Pedigree Application. *Genetics* 162(2), 993–996 (2002)
5. Chen, W.-M., Abecasis, G.R.: Family-based association tests for genomewide association scans. *American Journal of Human Genetics* 81, 913–926 (2007)
6. Daly, M.J., Rioux, J.D., Schaffner, S.F., Hudson, T.J., Lander, E.S.: High-resolution haplotype structure in the human genome. *Nature Genetics* 29(2), 229–232 (2001)
7. Ding, Z., Filkov, V., Gusfield, D.: A linear-time algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology* 2, 522–553 (2006)
8. Elston, R.C., Stewart, J.: A general model for the analysis of pedigree data. *Human Heredity* 21, 523–542 (1971)
9. Eskin, E., Halperin, E., Karp, R.: Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology* 1(1), 1–20 (2003)
10. Fishelson, M., Dovgolevsky, N., Geiger, D.: Maximum likelihood haplotyping for general pedigrees. *Human Heredity* 59, 41–60 (2005)
11. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In: Proceedings of the 6th Annual International Conference on (Research in) Computational (Molecular) Biology (2002)

12. Halperin, E., Kimmel, G., Shamir, R.: Tag SNP selection in genotype data for maximizing SNP prediction accuracy. *Bioinformatics* 21(suppl. 1), i195–i203 (2005)
13. Jensen, C.S., Kong, A.: Blocking gibbs sampling for linkage analysis in large pedigrees with many loops. *American Journal of Human Genetics* 65 (1999)
14. Lander, E.S., Green, P.: Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Science* 84(5), 2363–2367 (1987)
15. Lauritzen, S.L., Sheehan, N.A.: Graphical models for genetic analysis. *Statistical Science* 18(4), 489–514 (2003)
16. Piccolboni, A., Gusfield, D.: On the complexity of fundamental computational problems in pedigree analysis. *Journal of Computational Biology* 10(5), 763–773 (2003)
17. Sutter, N.B., Bustamante, C.D., Chase, K., Gray, M.M., Zhao, K., Zhu, L., Padukasahasram, B., Karlins, E., Davis, S., Jones, P.G., Quignon, P., Johnson, G.S., Parker, H.G., Fretwell, N., Mosher, D.S., Lawler, D.F., Satyaraj, E., Nordborg, M., Lark, K.G., Wayne, R.K., Ostrander, E.A.: A Single IGF1 Allele Is a Major Determinant of Small Size in Dogs. *Science* 316(5821), 112–115 (2007)
18. Thomas, A., Abkevich, V., Gutin, A., Bansal, A.: Multilocus linkage analysis by blocked gibbs sampling. *Statistics and Computing* 10(3), 259–269 (2000)

# Storage and Retrieval of Individual Genomes

Veli Mäkinen<sup>1,\*</sup>, Gonzalo Navarro<sup>2,\*\*</sup>, Jouni Sirén<sup>1,\*\*\*</sup>, and Niko Välimäki<sup>1,†</sup>

<sup>1</sup> Department of Computer Science, University of Helsinki, Finland  
`{vmakinen,jltsiren,nvalimak}@cs.helsinki.fi`

<sup>2</sup> Department of Computer Science, University of Chile, Chile  
`gnavarro@dcc.uchile.cl`

**Abstract.** A repetitive sequence collection is one where portions of a *base sequence* of length  $n$  are repeated many times with small variations, forming a collection of total length  $N$ . Examples of such collections are version control data and genome sequences of individuals, where the differences can be expressed by lists of basic edit operations. Flexible and efficient data analysis on a such typically huge collection is plausible using suffix trees. However, suffix tree occupies  $O(N \log N)$  bits, which very soon inhibits in-memory analyses. Recent advances in full-text *self-indexing* reduce the space of suffix tree to  $O(N \log \sigma)$  bits, where  $\sigma$  is the alphabet size. In practice, the space reduction is more than 10-fold, for example on suffix tree of Human Genome. However, this reduction factor remains constant when more sequences are added to the collection.

We develop a new family of self-indexes suited for the repetitive sequence collection setting. Their expected space requirement depends only on the length  $n$  of the base sequence and the number  $s$  of variations in its repeated copies. That is, the space reduction factor is no longer constant, but depends on  $N/n$ .

We believe the structures developed in this work will provide a fundamental basis for storage and retrieval of individual genomes as they become available due to rapid progress in the sequencing technologies.

**Keywords:** Comparative genomics, full-text indexing, suffix tree, compressed data structures.

## 1 Introduction

### 1.1 Motivation

*Self-indexing* [15] is a new proposal for storing and retrieving sequence data. It aims to represent the sequence (a.k.a. text or string) compressed in a way that not only random access to the sequence is possible, but also pattern searches are supported [4,7,20].

\* Funded by the Academy of Finland under grant 119815.

\*\* Partially funded by Millennium Institute for Cell Dynamics and Biotechnology (ICDB), Grant ICM P05-001-F, Mideplan, Chile.

\*\*\* Funded by the Research Foundation of the University of Helsinki.

† Funded by the Helsinki Graduate School in Computer Science and Engineering.

A special case of a text collection is one which contains several *versions* of one or more *base sequences*. Such collections are soon becoming reality in the field of molecular biology. As the DNA sequencing technologies become faster and more cost-effective, the sequencing of individual genomes will become a feasible task [3][10][17]. This is likely to happen in the near future, see for example the 1000 Genomes project<sup>1</sup>. With such data in hand, many fundamental issues become of top concern, like how to store, say, one million Human Genomes, not to speak about analyzing them. For the analysis of such collections, one would clearly need to use some variant of a *generalized suffix tree* [9], which provides a variety of algorithmic tools to do analyses in linear or near-linear time. The memory requirement of such a solution, however, is unimaginable with current random access memories, and also challenging in permanent storage.

Self-indexes should, in principle, cope well with genome sequences, as genomes contain high amounts of repetitive structure. In particular, as the main building blocks of *compressed suffix trees* [2][19][18][6], self-indexes enable compressing sequence collections close to their *high-order entropy* and enabling flexible analysis tasks to be carried out<sup>2</sup>. Those indexes have been successful in bringing down the space requirement of the suffix tree of *one* Human Genome to fit the capabilities of a desktop computer. However, they suffer from a fundamental limit: The high-order entropies they achieve are defined by the frequencies of symbols in their fixed-length contexts, and these contexts do not change *at all* when more *identical* sequences are added to the collection. Hence, these self-indexes are not at all able to exploit the fact that the texts in the collection are highly similar.

## 1.2 Content

In this paper we propose a new family of self-indexes that are suitable for storing highly repetitive collections of sequences, and a new compressed suffix tree based on it. Our scheme can also be thought of as a self-index for a given multiple alignment of a sequence collection, where one can retrieve any part of any sequence as well as make queries on the content of all the aligned sequences. The main technical contribution is a new strategy to store *suffix array* samples that uses and improves a classical solution for *persistent selection*.

We show analytically that the expected space requirement of our new self-indexes improves upon the existing ones on highly repetitive collections. We also provide experiments on a collection of resequenced yeast genomes, showing that our indexes behave in practice as predicted by our analysis.

## 1.3 Definitions and Background

A string  $S = S_{1,n} = s_1s_2 \cdots s_n$  is a *sequence of symbols* (a.k.a. characters or letters). Each symbol is an element of an *alphabet*  $\Sigma = \{1, 2, \dots, \sigma\}$ . A *substring*

<sup>1</sup> <http://www.1000genomes.com>

<sup>2</sup> For a concrete example, the *SUDS Genome Browser* at <http://www.cs.helsinki.fi/group/suds/cst>, runs a compressed suffix tree of the Human Genome using 8.8 GB of main memory.

of  $S$  is written  $S_{i,j} = s_i s_{i+1} \dots s_j$ . A *prefix* of  $S$  is a substring of the form  $S_{1,j}$ , and a *suffix* is a substring of the form  $S_{i,n}$ . If  $i > j$  then  $S_{i,j} = \varepsilon$ , the empty string of length  $|\varepsilon| = 0$ . A *text* string  $T = T_{1,n}$  is a string terminated by the special symbol  $t_n = \$ \notin \Sigma$ , smaller than any other symbol in  $\Sigma$ . The *lexicographical order* “ $<$ ” among strings is defined in the obvious way.

We use the standard notion of *empirical k-th order entropy*  $H_k(T)$ . For formal definition, see e.g. [14]. For our purposes, it is enough to know the basic property  $0 \leq H_k(T) \leq H_{k-1}(T) \leq \dots \leq H_0(T) \leq \log \sigma$ .

The compressors to be discussed are derivatives of the *Burrows-Wheeler transform (BWT)* [2]. The transform produces a permutation of  $T$ , denoted by  $T^{bwt}$ , as follows: (i) Build the *suffix array* [13]  $\text{SA}[1, n]$  of  $T$ , that is an array of pointers to all the suffixes of  $T$  in the lexicographic order; (ii) The transformed text is  $T^{bwt} = L$ , where  $L[i] = T[\text{SA}[i] - 1]$ , taking  $T[0] = T[n]$ . The BWT is reversible, that is, given  $T^{bwt} = L$  we can obtain  $T$  as follows: (a) Compute the array  $C[1, \sigma]$  storing in  $C[c]$  the number of occurrences of characters  $\{\$, 1, \dots, c-1\}$  in the text  $T$ ; (b) Define the *LF mapping* as follows:  $LF(i) = C[L[i]] + rank_{L[i]}(L, i)$ , where  $rank_c(L, i)$  is the number of occurrences of character  $c$  in the prefix  $L[1, i]$ ; (c) Reconstruct  $T$  backwards as follows: set  $s = 1$ , for each  $n-1, \dots, 1$  do  $t_i \leftarrow L[s]$  and  $s \leftarrow LF[s]$ . Finally put the end marker  $t_n \leftarrow \$$ .

Let a *point mutation* (or just *mutation*) denote the event of a symbol changing into another symbol inside a string. We study the following problem (other types of mutations are considered later).

**Definition 1.** Given a collection  $\mathcal{C}$  of  $r$  sequences  $T^k \in \mathcal{C}$  such that  $|T^k| = n$  for  $1 \leq k \leq r$  and  $\sum_{k=1}^r |T^k| = N$ , where  $T^2, T^3, \dots, T^r$  are mutated copies of the base sequence  $T^1$  containing overall  $s$  point mutations, the repetitive collection indexing problem is to store  $\mathcal{C}$  in as small space as possible such that the following operations are supported as efficiently as possible: **count**( $P$ ) (how many times  $P$  appears as a substring of the texts in  $\mathcal{C}$ ?); **locate**( $P$ ) (list the occurrence positions of  $P$  in  $\mathcal{C}$ ); and **display**( $k, i, j$ ) (return  $T_{i,j}^k$ ).

The above is an extension of the well-known *basic indexing problem*, where the collection only has one sequence  $T$ . We call a solution to the basic indexing problem a *self-index* if it does not need  $T$  to solve the three queries above. Thus a self-index replaces  $T$ .

A classical solution to the basic indexing problem uses  $T$  and the suffix array  $\text{SA}[1, n]$ . Two binary searches find the interval  $\text{SA}[sp, ep]$  pointing to all the suffixes of  $T$  starting with  $P$ , that is, to all the occurrences of  $P$  in  $T$  (this solves **count** and **locate**) [13], and  $T$  is at hand for **display**. The solution is not space-efficient, since array  $\text{SA}$  requires  $n \log n$  bits (compared to  $n \log \sigma$  bits used by  $T$ ), and it is not a self-index, since  $T$  is needed.

The *FM-index* [4] is a self-index based on the BWT. It solves **count** by finding the interval  $\text{SA}[sp, ep]$  that contains the occurrences of  $P$ . The FM-index uses the array  $C$  and function  $rank_c(L, i)$  in the so-called *backward search* algorithm, calling function  $rank_c(L, i)$   $O(|P|)$  times. The two other basic queries are solved using sampling of  $\text{SA}$  and its inverse  $\text{SA}^{-1}$ , and the *LF-mapping* to derive the unsampled values from the sampled ones. Many variants of the FM-index have

been derived that differ mainly in the way the  $rank_c(L, i)$ -queries are solved [15]. For example, on small alphabets, it is possible to achieve  $nH_k + o(n \log \sigma)$  bits of space, for moderate  $k$ , with constant time support for  $rank_c(L, i)$  [5].

Now, the repetitive collection indexing problem can be solved using the normal self-index for the concatenation  $T^1\$T^2\$ \dots T^r\$$ . However, the space requirement achieved, even with a high-entropy compressed index, is not attractive for the case of repetitive collections. For example, an FM-index [5] requires  $NH_k(C) + o(N \log \sigma)$  bits. Notice that with the collection of Def. 1 and even with  $s = 0$ , it holds  $H_k(C) \approx H_k(T^1)$ , and hence the space is about  $r$  times that for the base sequence, not taking any advantage of repetitiveness.

In the sequel, we derive solutions whose space requirements depend on  $n$  and  $s$  instead of  $N$ . Let us first consider a natural lower bound that takes into account these specific problem parameters. Consider a two-part compression scheme that compresses  $T^1$  with a high-order compressor, and the rest of the sequences by encoding the mutations needed to convert each other sequence into  $T^1$ . The lower bound for any such compressor is

$$nH_k(T^1) + \log \binom{N-n}{s} + s \log \sigma \approx nH_k(T^1) + s \log \frac{N}{s} + s \log \sigma \quad (1)$$

where the first part is the lower bound of encoding  $T^1$  with any high-order compressor, the second part is the lower bound for telling the positions of the mutations among the  $N - n$  possibilities, and the third part is the lower bound for listing the  $s$  mutated values.

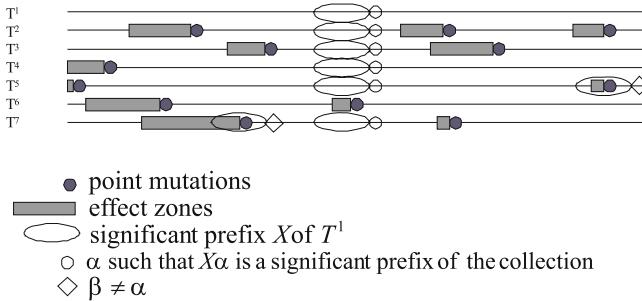
Notice that it is not difficult to achieve *just plain compression* approaching the bound of Eq. (1) (omitting alphabet-dependent factors), but we aim higher: Our goal is to solve the repetitive collection indexing problem within the same space. We do not yet achieve that goal, but the space of our indexes can be expressed in similar terms; we encourage the reader to compare our final result with Eq. (1) to see the connection.

The abstract problem with point mutations studied here is much simpler than the real variations occurring in genome sequences. However, all the techniques introduced can be extended to the full set of mutation events, as is done in our implementation. This will be discussed in Sect. 3.

## 2 Methods

### 2.1 Analysis of Runs

*Self-repetitions* are the fundamental source of redundancy in suffix arrays, enabling their compression. A self-repetition is a maximal interval  $\text{SA}[i, i+l]$  of suffix array  $\text{SA}$  having a *target interval*  $\text{SA}[j, j+l]$  such that  $\text{SA}[j+r] = \text{SA}[i+r] + 1$  for all  $0 \leq r \leq l$ . Let  $\Psi(i) = \text{SA}^{-1}[\text{SA}[i] + 1]$ . The intervals of  $\Psi$  corresponding to self-repetitions in the suffix array are called *runs*. The name stems from the fact that  $\Psi(i+1) = \Psi(i) + 1$  when both  $\Psi(i)$  and  $\Psi(i+1)$  are contained in the same run (see [12, 15] for more details).



**Fig. 1.** An example of the significant prefix concept. Let (a repeated collection with) base text  $S^1 = T^1$  contain a significant prefix  $X$ . Substring  $X$  becomes repeated in the mutated copies  $T^2, T^3, T^4, T^5$ , and  $T^7$ , of  $T^1$ . Text  $T^6$  has a mutation inside  $X$ . Due to other mutations, texts  $T^5$  and  $T^7$  now contain  $X$  in some other positions, and hence  $X$  is no longer a significant prefix of the mutated collection. However, extending  $X$  with string  $\alpha$  makes  $X\alpha$  unique to the original position of  $X$ , while the other two occurrences of  $X$  are succeeded by string  $\beta \neq \alpha$ . Hence,  $X\alpha$  is a significant prefix, being  $\alpha$  the shortest extension having the required property. The significant prefixes starting at the *effect zones* shown are affected by the mutations.

Let  $R_\Psi(T)$  be the number of runs in  $\Psi$  of text  $T = T_{1,n}$  and  $R_{bwt}(T)$  the number of equal-letter runs in  $T^{bwt}$ , the BWT of  $T$ . If the text is evident from the context, we will usually drop  $T$  and write just  $R_\Psi$  and  $R_{bwt}$ . It is known that  $R_\Psi \leq R_{bwt} \leq R_\Psi + \sigma$ , making the two types of runs almost equal [12]. Hence, we may simplify the notation further by denoting just  $R = R_{bwt}(T)$ . In addition to the trivial bound  $R \leq n$ , we also have  $R \leq nH_k + \sigma^k$  for all  $k$  [12].

We will now prove some further bounds for texts obtained by repeating and mutating substrings of a base sequence. To simplify the analysis, we add a new character  $\#$  such that  $\# < \$ < c$  for all  $c \in \Sigma$ . We use  $\#$  as a separator between texts in the collection, and assume that the ordering between two occurrences of character  $\#$  is decided by their positions in the sequence, making each occurrence of  $\#$  a different character in practice.

**Definition 2.** The  $r$  times repeated collection of base text  $S = S_{1,n}$  is  $S^r = S^1S^2 \dots S^r$ , where  $S^r = S = S_{1,n-1}\$$  and  $S^i = S_{1,n-1}\#$  for all  $i < r$ .

**Definition 3.** Let  $T^r$  be a collection of  $r$  texts, each derived by mutations from a base sequence  $T = T^1$ . The significant prefix  $SP_{i,j}$  of the suffix starting at position  $j$  of sequence  $T^i$  is the shortest prefix not occurring anywhere else in  $T^r$  as a substring except possibly as a prefix of some  $T_{j,n}^k$ ,  $k \neq i$ .

Notice that significant prefix concept is well-defined also a repeated collection, since it is a collection with no mutations. In that case, significant prefixes are identical to those of a collection consisting only of the base text. Figure 1 illustrates the definitions.

We now show some basic results concerning the number of runs in repeated and mutated texts. Proofs are sketched here for conciseness. Full proofs

will appear in the full version. Expected case proofs extend those in [23, pp.263–265].

**Lemma 1.** *For all texts  $S$  and all  $r \geq 1$ ,  $R_\Psi(S) = R_\Psi(S^r)$ .*

*Proof.* (Sketch) All suffixes corresponding to the same suffix of  $S$  are grouped together in the suffix array of  $S^r$ . Each group is further ordered from the suffix of  $S^1$  to the suffix of  $S^r$ . Hence there is one-to-one correspondence between the self-repetitions of suffix arrays of  $S$  and  $S^r$ .

**Lemma 2.** *Let  $S^r = S^1S^2\cdots S^r$  be a repeated collection and  $\mathcal{T}^r$  the collection created by transforming  $s_j^i$ , for some  $1 < i \leq r$  and  $1 \leq j < n$ , into another character. Then  $R_\Psi(\mathcal{T}^r) \leq R_\Psi(S^r) + 2c + 2 = R_\Psi(S) + 2c + 2$ , where  $c$  is the number of significant prefixes covering  $t_j^i$ .*

*Proof.* (Sketch) The relative position of a suffix in the suffix array can change only if its significant prefix has mutated. Each such suffix can interfere with a constant number of runs. The ordering of suffixes sharing a significant prefix can change, but this does not create additional runs.

**Lemma 3.** *Let  $S = S_{1,n}$  be a random text. The expected length of the longest repeated substring is  $O(\log_\sigma n)$ .*

*Proof.* (Sketch) The expected number of non-overlapping repeats of length  $l$  is  $O(n^2/\sigma^l)$ . Markov's inequality bounds the probability of having a repeat of length  $c \cdot \log_\sigma n$  exponentially in  $c^{-1}$ . Overlapping repeats are handled in a similar manner.

**Lemma 4.** *Let  $S^r$  be the repeated collection of random text  $S = S_{1,n}$  with total length  $N = nr$ . Let  $\mathcal{T}^r$  be  $S^r$  after  $s$  point mutations at random positions in  $S^2S^3\cdots S^r$ . The expected value of  $R_\Psi(\mathcal{T}^r)$  is at most  $R_\Psi(S) + O(s \log_\sigma N)$ .*

The above analysis can be extended to other types of mutations. When we insert a new copy of an existing substring, the significant prefixes completely within the new copy remain unchanged. Only the significant prefixes covering either end of the inserted copy change. Hence the insertion is essentially equivalent to two point mutations. Similarly the deletion of a substring is equivalent to one point mutation.

## 2.2 Backward Search for Repetitive Collections

Our prior work [22] introduced three solutions to the repetitive collection indexing problem, restricted to  $\text{count}(P)$  query. The three indexes are *Run-Length Compressed Suffix Array (RLCSA)*, *Run-Length Encoded Wavelet Tree (RLWT)*, and *Improved Run-Length FM-Index (RLFM+)*. They achieve different space vs. time trade-offs. For example, RLFM+ requires space  $(R \log \sigma + 2R \log \frac{N}{R})(1 + o(1)) + O(R \log \log \frac{N}{R})$  bits. Query  $\text{rank}_c(T^{bwt}, i)$ , and retrieving symbol  $t_i^{bwt}$ , are solved in  $O(t_{LF})$  time, where  $t_{LF} = O(\log R)$ <sup>3</sup>. Hence, RLFM+ supports  $\text{count}(P)$  in time  $O(|P|t_{LF})$ . Lemma 4 can be used to bound  $R$  with

---

<sup>3</sup> One can achieve  $o((\log \log N)^2)$  time by adding  $O(R \frac{\log N}{\log R})$  further bits of space [8].

$n + O(s \log_\sigma N)$  in the expected case (even for an incompressible  $T^1$ ), which gives a space bound close to the terms of Eq. (II).

### 2.3 Suffix Array Samples

Supporting the other two functions of the repetitive collection indexing problem, namely, `display()` and `locate()`, is the main contribution of this paper. We address this now.

We need to be able to map the suffixes of the text into suffix array indexes and vice versa. The standard solution [15] in self-indexes is to sample every  $d$ -th suffix of each text in the collection in an array  $D[1, N/d + 1]$ , such that  $D[i] = \text{SA}^{-1}(i \cdot d)$ , mark the locations  $D[i]$  in a bit-vector  $B[1, N]$ , such that  $B[D[i]] = 1$  for all  $1 \leq i \leq N/d + 1$ , and store the samples in the suffix array order in a table  $S[1, N/d + 1]$ , such that  $S[\text{rank}_1(B, D[i])] = i \cdot d$ .

Then `display( $k, i, j$ )` works as follows. Let  $St[k]$  be the starting position of  $T^k$  in the concatenated sequence  $T = T^1T^2\cdots T^r$ . Value  $D[(St[k]+j)/d+2] = e$  tells us that the nearest sampled suffix after  $T_{SP[k]+j, N}$  is pointed from  $\text{SA}[e]$ . Following *LF*-mapping from position  $e$  reveals us backwards a substring that covers  $T_{i,j}^k$  in time  $O(t_{LF}(d+j-i))$ .

Function `locate( $P$ )` works as follows. First, backward search finds the range  $\text{SA}[sp, ep]$  containing the occurrences of  $P$ , and then  $\text{SA}[i]$  is computed for each  $sp \leq i \leq ep$  as follows. If suffix  $\text{SA}[i]$  is not sampled ( $B[i] = 0$ ), then *LF*-mapping is applied until an index  $j$  is found where  $\text{SA}[j]$  is sampled ( $B[j] = 1$ ). Then  $\text{SA}[i] = S[\text{rank}_1(B, j)] + c$ , where  $c < d$  is the number of times *LF*-mapping was applied. This takes time  $t_{SA} = O(t_{LF} \cdot d)$ .

The space required by the standard solution is  $O((N/d) \log N + N)$  bits, which can be reduced to  $O((N/d) \log N)$  by using the *binary searchable dictionary (BSD)* representation [8]; this changes the time for `locate()` into  $t_{SA} = O((t_{LF} + t_{SA})d)$ , where  $t_{SA} = O(\log d)$ .

Our objective is to have all time requirements in  $O(\text{polylog}(N))$ , which holds only with the above approaches if we assume  $r = O(\text{polylog}(N))$ ; then  $d$  can be chosen as  $r \log N$  to make  $O((N/d) \log N) = O(n)$ , i.e., independent of  $N$  as we wish.

**Improving Space for `display()`.** We will store samples only for  $T^1$ , that is, table  $D[1, n/d + 1]$  has the suffix array entry of every  $d$ -th suffix  $T_{i,d,n}^1$  stored at  $D[i] = \text{SA}^{-1}(i \cdot d)$ .

To be able to use the same samples for other texts in the collection, we mark the locations of mutations into bit-vectors. Let  $M^k[1, |T^k|]$  be a bit-vector where the locations of the mutations inside  $T^k$  are marked. The mutated symbols are stored in another array  $MS^k[1, \text{rank}_1(M^k, |T^k|)]$  in their order of occurrence in  $T^k$ .

Consider now a query `display( $k, i, j$ )`. The substring  $T_{i,j}^1$  is extracted using the samples just like in the standard approach. It is easy to see that while extracting  $T_{i,j}^1$ , the mutations stored for  $T^k$  can also be extracted using *rank*-function on  $M^k$ . Table  $MS^k$  occupies overall  $s \log \sigma$  bits. Bit-vectors  $M^k$  can be

represented using BSD [8] in overall  $s \log \frac{N-n}{s} (1 + o(1)) + O(s \log \log \frac{N-n}{s})$  bits. What we gain is that  $O((N/d) \log N)$  becomes  $O((n/d) \log n)$ .

**Improving Space for `locate()`.** We use the same strategy as for `display()`, sampling only  $T^1$ , but this time we need to sample also parts of the other texts, as discussed next.

Let us first consider the case of  $r$  identical texts. We know that the suffixes  $T_{p,n}^1, T_{p,n}^2, \dots, T_{p,n}^r$  will all be consecutive and in the same order in  $\text{SA}$ . Assume every  $d$ -th suffix of  $T^1$  is sampled and those sampled  $\text{SA}$  positions are marked in a bit vector  $B$ . Then we can reveal any  $\text{SA}[i]$  by applying  $LF$ -mapping at most  $d$  times until finding an entry  $j$  such that  $\text{SA}[j']$  is sampled for some  $j' < j$  and  $j - j' \leq r$ . The candidate  $j' < j$  to check is  $j' = \text{select}_1(B, \text{rank}_1(B, j))$ , where  $\text{select}_1(B, x)$  gives the position of the  $x$ -th 1 in  $B$ . Then  $\text{SA}[j]$  corresponds to suffix  $T_{S[\text{rank}_1(B, j')] + c, n}^k$ , where  $S$  is the table storing the sampled suffixes in the order they appear in  $\text{SA}$ ,  $c < d$  is the number of times  $LF$ -mapping was applied, and  $k = j - j' + 1$ .

Generalizing the scheme to work under mutations is non-trivial. We introduce a strategy that splits the suffixes into two classes A and B such that class A suffixes are computed via  $T^1$  samples and for class B we add new samples from all the texts. Recall Lemma 2: Class B contains the  $c$  suffixes whose significant prefixes overlap one or more mutations. Class A contains all other suffixes.

Let us first consider the case when  $\text{SA}[i]$  is a class B suffix. Class B suffixes form at most  $s$  disjoint regions in texts  $T^k$ ,  $2 \leq k \leq r$ . We sample every  $d$ -th suffix inside each of these regions. The suffix array indexes containing these sampled suffixes are marked in a bit-vector  $E[1, N]$ , and a table  $S^B[1, \text{rank}_1(E, N)]$  stores these sampled suffixes in the order they appear in  $\text{SA}$ . Retrieving  $\text{SA}[i]$  is completely analogous to the standard sampling scheme by using  $S^B$  in place of  $S$  and  $E$  in place of  $B$ . The space is bounded by  $O((c/d) \log N)$ , which is  $O(((s \log_\sigma N)/d) \log N)$  in the average case.

Computing  $\text{SA}[i]$  for class A suffixes is more challenging than in the case of  $r$  identical texts, when all suffixes were class A. The problem can be divided into the following subproblems: (i) Not all sampled suffixes of  $T^1$  will have counterparts in all the other texts. Hence, we need to store explicitly a list  $Q[\text{rank}_1(B, \text{SA}^{-1}[i \cdot d])] = k_1 k_2 \dots k_p$  denoting texts  $T^{k_1}, T^{k_2}, \dots, T^{k_p}$ ,  $p \leq r$ , that correspond to a sampled suffix  $T_{i \cdot d, n}^1$ . However, this takes too much space. (ii) Class B suffixes break the order of the suffixes aligned to the same sampled  $T^1$  suffix, making it difficult to know, once at  $\text{SA}[j]$ , whether there is a sampled suffix of  $T^1$  at some close enough position  $\text{SA}[j']$ .

Let us consider subproblem (ii) first. A solution is to explicitly mark all class B suffixes in  $\text{SA}$  into a bit-vector  $F[1, N]$ , and to store for each sampled suffix  $T_{i \cdot d, n}^1$  its lexicographic rank  $e$  among the suffixes in the list  $Q[\text{rank}_1(B, \text{SA}^{-1}[i \cdot d])] = k_1 k_2 \dots k_p$ , that is,  $e$  such that  $k_e = 1$ . Now, consider again the situation where  $\text{SA}[i]$  belongs to class A and  $LF$  mapping has brought us to entry  $\text{SA}[j]$ . Let us compute  $\text{prev} = \text{select}_1(B, \text{rank}_1(B, j))$ ,  $\text{succ} = \text{select}_1(B, \text{rank}_1(B, j) + 1)$ ,  $d\text{prev} = (j - \text{prev}) - (\text{rank}_1(F, j) - \text{rank}_1(F, \text{prev}))$ , and  $d\text{succ} = (\text{succ} - j) - (\text{rank}_1(F, \text{succ}) - \text{rank}_1(F, j))$ . Let  $Q[\text{rank}_1(B, \text{prev})] = k_1 k_2 \dots k_p$  and  $e$  be

such that  $k_e = 1$ . If  $d_{prev} \leq p - e$  then  $k_{e+d_{prev}}$  is the number of the text where suffix  $SA[j]$  belongs to. This follows from the fact that the effect of class B suffixes is eliminated using *rank*, so it remains to calculate how many class A suffixes there are between the sampled suffix and current position. If this number is smaller than (or equal to the) the number of suffixes with rank higher than that of  $SA[prev]$  in the list  $Q[rank_1(B, prev)]$ , then (and only then)  $SA[j]$  belongs to the same list. Analogously, one can check whether  $SA[j]$  belongs to the list  $Q[rank_1(B, succ)] = k_1 k_2 \dots k_p$  of  $SA[succ]$ . After at most  $d$  steps of *LF*-mapping the correct  $Q$ -list is found. The additional space needed is  $O(c \log \frac{N-n}{c})$  bits for the *BSD* of bit vector  $F$ .

Finally, we are left with subproblem (i): the lists  $Q[1], Q[2], \dots, Q[n/d]$  occupy in total  $O((n/d)r \log r)$  bits. We will next improve the space to  $O(s \log s)$  bits modifying a classical solution by Overmars [16] to  *$k^{th}$  element/rank searching in the past*. The original structure is reviewed in Theorem 1 and then Theorem 2 improves the space and makes the structure *confluent/persistent* (see [11] for background).

**Definition 4.** Let  $E(t) = e_1^t e_2^t \dots e_{p_t}^t \in \mathcal{R}^* = \{1, 2, \dots, r\}^*$  be a sequence of elements at time point  $t \in H$ , where  $H \subseteq \mathcal{H} = \{1, 2, \dots, h\}$ , such that  $E(t)$  can be constructed from  $E(tprev)$ ,  $tprev = \max\{t' \in H \mid t' < t\}$ , by deleting some  $e_k^{tprev}$  or inserting a new element  $e \in \mathcal{R}$  between some  $e_{k-1}^{tprev}$  and  $e_k^{tprev}$  (or before  $e_1^{tprev}$  or after  $e_{p_t}^{tprev}$ ). The persistent selection problem is to construct a static data structure  $\mathcal{D}$  on  $\{E(t) \mid t \in H\}$  that supports operation  $\text{select}(t, k) = e_k^t$ . The online persistent selection problem is to maintain  $\mathcal{D}$  such that it supports  $\text{insert}(t, e, k)$  and  $\text{delete}(t, k)$ , where value  $t$  must be at least  $\max(H)$ . The confluent/persistent selection problem allows value  $t$  to be any  $t \in \mathcal{H}$  also for insertions and deletions.

**Theorem 1** ([16]). There is a data structure  $\mathcal{D}$  for the online persistent selection problem occupying  $O(x(\log x \log h + \log r))$  bits of space and supporting  $\text{select}(t, k)$  in  $O(\log x)$  time, and  $\text{insert}(t, e, k)$  and  $\text{delete}(t, k)$  in amortized  $O(\log x)$  time, where  $x$  is the number of insertion and deletion operations executed during the lifetime of  $\mathcal{D}$ .

*Proof.* (Sketch) The structure  $\mathcal{D}$  is a variant of balanced binary tree that stores subtree sizes in its internal nodes, enhanced with *path copying* and *fractional cascading* to support persistence: Consider a tree  $\mathcal{T}(t)$  for storing elements of  $E(t)$  in its leaves and having subtree sizes stored in its internal nodes. Selecting the  $k$ -th leaf equals accessing  $e_k^t$ . It is easy to find that leaf by following the path from the root and comparing  $k$  with the sum of subtree sizes of nodes that remain hanging left side of the path; if at node  $v$  the current sum plus subtree size of the left child of  $v$  is smaller than  $k$ , go right, otherwise go left. Now, consider an insertion to produce  $E(t)$  from  $E(tprev)$ . To produce  $\mathcal{T}(t)$  one can add a new leaf to  $\mathcal{T}(tprev)$  and increment the subtree sizes by one on the path to the new leaf. To make this change persistent, the idea in [16] is to copy the old subtree size information into a new field on each node on the path and increment that. The field is labeled with the time  $t$  and also pointers are associated to the

corresponding fields on the left and right child of the node, respectively. Here corresponding means a field whose time-stamp is largest  $t'$  such that  $t' \leq t$ . Analogous procedure is executed for deletions, except that the corresponding leaf is not deleted, but only the subtree sizes are updated accordingly. This procedure is repeated over all time points and the tree is rebalanced when necessary. The rotations to rebalance the tree require merging the lists of fields storing the time-stamped information. The cost of rebalancing can be amortized over insertions and deletions [16]. The root of the tree stores the time-stamped list as a binary search tree to provide  $O(\log x)$  time access to the entries. The required space for the tree itself is  $O(x \log x \log h)$  bits as each of the  $x$  updates creates a new field occupying  $O(\log h)$  bits for each of the  $O(\log x)$  nodes on the path from root to the leaf. In addition, each leaf contains a value of size  $\log r$  bits.

**Theorem 2.** *There is a data structure  $\mathcal{D}$  for the persistent selection problem occupying  $O(x(\log x + \log h + \log r))$  bits of space and supporting  $\text{select}(t, k)$  in  $O(\log x)$  time, where  $x$  is the number of insertions and deletions to construct  $\mathcal{D}$ . There is also an online/confluent version of  $\mathcal{D}$  that occupies the same space, but  $\text{select}(t, k)$  takes  $O(\log^2 x)$  time, and  $\text{insert}(t, e, k)$  and  $\text{delete}(t, k)$  take amortized  $O(\log^2 x)$  time.*

*Proof.* We modify the structure of Theorem 1 by replacing the time-stamped lists of fields in each node of the tree with two partial sums that can be represented succinctly. Let  $S^v = s_0^v s_1^v s_2^v \cdots s_{k^v}^v$  be the list of subtree sizes stored in some node  $v$ , where  $s_0^v = 0$ . Let  $\hat{S}^v = (s_1^v - s_0^v)(s_2^v - s_1^v) \cdots (s_{k^v}^v - s_{k^v-1}^v)$ . We represent  $\hat{S}^v$  via succinct data structure for (dynamic) partial sums to support operations  $\text{select}(\hat{S}^v, i) = \sum_{j=1}^i \hat{s}_j^v = s_i^v$ . In addition, we construct a bit-vector  $B^v[1, k^v]$  where  $B^v[i] = 1$  if and only if the change  $s_i^v$  came from the right child of  $v$ . Notice that we do not need the explicit fractional cascading links anymore, as we have the connection  $\text{select}(\hat{S}^v, i) = \text{select}(\hat{S}^l, i - i') + \text{select}(\hat{S}^r, i')$ , where  $i' = \text{rank}_1(B^v, i)$ , and  $l$  and  $r$  are the left and right children of  $v$ . That is,  $\text{select}(\hat{S}^l, i - i')$  and  $\text{select}(\hat{S}^r, i')$  are the subtree sizes of nodes  $l$  and  $r$ , respectively, at the same time point as  $s_i^v$ . In the root of the tree we keep the original binary search tree to map the parameter  $t$  to its rank  $i$  and after that the formulas above can be used to compare subtree sizes to value of parameter  $k$ . Notice also that confluent  $\text{insert}$  and  $\text{delete}$  are immediately provided if we can support dynamic  $\text{select}$  on  $\hat{S}^v$  and dynamic  $\text{rank}$  on  $B^v$ .

Let us consider how to provide  $\text{select}(\hat{S}^v, i) = s_i^v$ . First we observe that  $\sum_{v \in \mathcal{T}} \sum_{j=1}^{k^v} \hat{s}_j^v = O(x \log x)$  because each insertion or deletion changes the subtree size by one on  $O(\log x)$  nodes. Hence, we can afford to use unary coding for these values. We represent each  $\hat{S}^v$  by a bit-vector  $F^v = f(\hat{s}_1^v)f(\hat{s}_2^v) \cdots f(\hat{s}_{k^v}^v)$ , where  $f(x) = 1^x$  if  $x > 0$  otherwise  $f(x) = 0^{-x}$ , and by a bit-vector  $G = 10^{|f(\hat{s}_1^v)|-1}10^{|f(\hat{s}_2^v)|-1} \cdots 10^{|f(\hat{s}_{k^v}^v)|-1}$ . Then  $\text{select}(\hat{S}^v, i)$  equals  $2 \cdot \text{rank}_1(F^v, j-1) - (j-1)$ , where  $j = \text{select}_1(G^v, i+1)$ . That is,  $\sum_{v \in \mathcal{T}} (|F^v| + |G^v|)(1 + o(1)) = O(x \log x)$  bits is enough to support constant time  $\text{select}$  on all subtree sizes, when the tree is static. In the dynamic case,  $\text{select}$  takes  $O(\log x)$  time [1]. Same analysis holds for bit-vectors  $B^v$ .

In summary, the tree in the root takes  $O(x \log h)$  bits, and support rank for  $t$  in  $O(\log x)$  time. The bit-vectors in the main tree occupy  $O(x \log x)$  bits and make a slowdown of  $O(1)$  or  $O(\log x)$  per node depending on the case. The associated values in the leaves occupy  $O(x \log r)$  bits.

Combining Lemma 4 and RLFM+ structure of Sect. 2.2 with Theorem 2 applied to sampling gives us the main result of the paper:

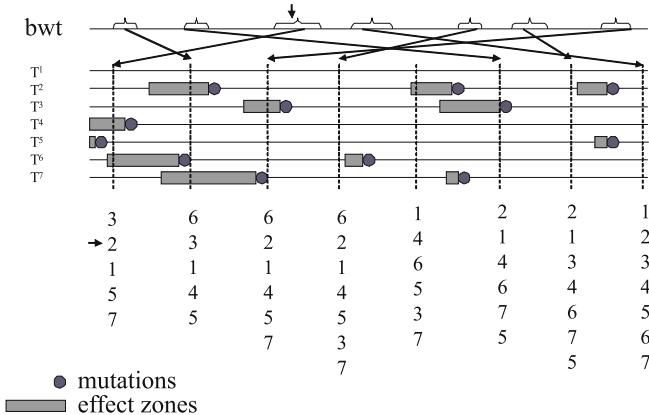
**Theorem 3.** *Given a collection  $\mathcal{C}$  and a concatenated sequence  $T$  of all the  $r$  sequences  $T^i \in \mathcal{C}$ , there is a data structure for the repetitive collection indexing problem taking*

$$\begin{aligned} & (R \log \sigma + 2R \log \frac{N}{R})(1 + o(1)) + O\left(R \log \log \frac{N}{R}\right) \\ & + O\left(s \log_{\sigma} N \log \frac{N}{s \log_{\sigma} N}\right) + O(s \log s) + O(r \log N) \\ & + O(((s \log_{\sigma} N)/d) \log N) + O((n/d) \log n) \end{aligned}$$

bits of space in the average case. The structure supports  $\text{count}(P)$  in time  $O(|P|t_{\text{LF}})$ ,  $\text{locate}(P)$  in time of  $\text{count}(P)$  plus  $O(d(t_{\text{LF}} + t_{\text{SA}}) + \log s)$  per occurrence,  $\text{display}(k, i, j)$  in time  $O((d + j - i)(t_{\text{LF}} + t_{\text{SA}}))$ , computing  $\text{SA}[i]$  and  $\text{SA}^{-1}[(k, j)]$  in time  $t_{\text{SA}} = O(d(t_{\text{LF}} + t_{\text{SA}}) + \log s)$ , and  $T(\text{SA}[i])$  in time  $O(\log \sigma)$ , where  $t_{\text{LF}} = O(\log R)$  and  $t_{\text{SA}} = O(\log d)$ .

*Proof.* (Sketch) The discussion preceding persistent selection developed data structures occupying  $O(s \log_{\sigma} N \log \frac{N}{s \log_{\sigma} N}) + O(((s \log_{\sigma} N)/d) \log N)$  bits to support parts of the remaining  $\text{locate}()$  operation. These are larger than the ones for  $\text{display}()$ . Theorem 2 provides a solution to subproblem (i): we can replace the lists  $Q[1], Q[2], \dots, Q[n/d]$  by persistent select, where the  $s$  mutations cause insertions and deletions to the structure (as they change the rank of a text between two samples, see Fig. 2 for an example). There will be  $s$  such updates, and on any given position  $i$  of the text  $T^1$  (including those that are sampled) one can select the  $k$ -th text aligned to that suffix in  $O(\log s)$  time. The space usage of this persistent structure is  $O(s \log s)$  bits. Computation of  $\text{SA}[i]$  is identical to  $\text{locate}()$ , but computation of  $\text{SA}^{-1}[(k, j)]$  needs some interplay with the structures of Theorem 2 considered next.

In the case  $t_j^k$  belongs to an area where a sampled position is at distance  $d$ , computation of  $\text{SA}^{-1}[(k, j)]$  resembles the display operation. Otherwise one must follow the closest sampled position after  $t_j^1$  to suffix array, and use at most  $d$  times the  $LF$ -mapping to find out  $\text{SA}^{-1}[(1, j)]$ . Now, to find the rank of text  $T^k$  with respect to that of text  $T^1$  in the persistent tree of Theorem 2 storing the lexicographic order of suffixes aligned to position  $j$ , one can do the following. Whenever a new leaf is added to the persistent tree, associate to that text position a pointer to this leaf. These pointers can be stored in  $O(s \log s)$  bits and their locations can be marked using  $s \log \frac{N}{s}(1 + o(1))$  space, so that one can find the closest location to  $(k, j)$  having a pointer, using  $\text{rank}$  in  $t_{\text{SA}}$  time.



**Fig. 2.** Persistent selection and changes in the lexicographic order of sampled suffixes. Text  $T^1$  has been sampled regularly and the pointers to the sampled suffixes are stored with respect to the BWT sequence. Each such pointer is associated a range containing the occurrences of the same significant prefix in the mutated copies of  $T^1$ . The relative lexicographic order of these aligned suffixes (shown below the sampled positions) change only when there is a mutation effect zone between the sampled positions; when an effect zone starts, the corresponding text is removed from the list, and when it ends (with the mutation), the text is inserted to the list with a new relative lexicographic order.

Following this pointer to the persistent tree leaf, and continuing to the root of the tree (and back), one can compute the rank of the leaf (text  $T^k$ ) in  $O(\log s)$  time. Computing the rank of  $(1, j)$  is analogous. By comparing these two ranks, one can find the correct index in the vicinity of  $\text{SA}^{-1}[(1, j)]$  making a *select()* operation on the bit-vector  $F$  used for *locate()* operation. The overall time is the same as for computing  $\text{SA}[i]$ . Finally, with a gap-encoded bit vectors storing tables  $C$  and  $C_B$ , the operation  $T[\text{SA}[i]]$  works in  $AT(N, \sigma)$  time.

The confluent version of Theorem 2 can be used to handle dynamic samples.

The result can be used to derive new compressed suffix trees: The entropy-bounded compressed suffix tree of Fischer et al. [6] uses an encoding of *LCP*-values (lengths of longest common prefixes of  $\text{SA}[i]$  and  $\text{SA}[i + 1]$ ) that consists of two bit-vectors of length  $N$ , each containing  $R$  bits set. In addition, only  $o(N)$  bit structures and normal suffix array operations are used for supporting an extended set of suffix tree operations. We can now use Theorem 3 to support suffix array functionality and BSD representation [8] to store *LCP*-values in  $2R \log \frac{N}{R} + O(R \log \log \frac{N}{R})$  bits. Thus, adding  $2R \log \frac{N}{R} + O(R \log \log \frac{N}{R}) + o(N)$  bits to the structure of Theorem 3, one can support all the suffix tree operations listed in [6] in  $O(\text{polylog}(N))$  time.<sup>4</sup>

<sup>4</sup> We remark that the solution is not quite satisfactory, as in  $o(N)$  space one can afford to use the standard suffix array sampling as well. Converting  $o(N)$  to  $o(n)$  is an open problem, and it seems to be common to all different compressed suffix tree approaches.

**Table 1.** Base structure sizes and times for `count()` and `display()` for various self-indexes on a collection of genomes of multiple strains of *Saccharomyces paradoxus* (36 sequences, 409 MB). The genomes were obtained from the Durbin Research Group at the Sanger Institute (<http://www.sanger.ac.uk/Teams/Team71/durbin/sgrp/>).  $\Psi$  sampling rate was set to 128 in CSA and to 32 bytes in RLCSA. Reported times are in microseconds / character.

Index	Size (MB)	<code>count()</code>	<code>display()</code>
CSA	95.51	2.86	0.41
SSA	121.70	0.48	0.40
RLFM	146.40	1.21	1.38
RLCSA	42.78	1.93	0.77
RLWT	34.67	17.30	10.24
RLFM+	54.77	3.03	2.10

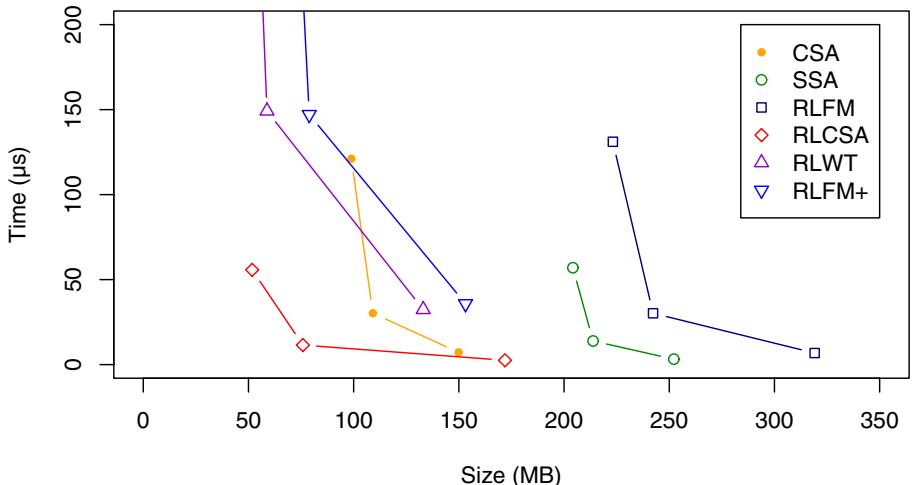
### 3 Implementation and Experiments

So far we have considered only point mutations on DNA, although there are many other types of mutations, like insertions, deletions, translocations, and reversals. The runs in the Burrows-Wheeler transform change only for those suffixes whose lexicographic order is affected by a mutation. In all mutation types (except in reversals) the effect to the lexicographic order of suffixes is similar to point mutations, so the expected case bounds limiting the length of significant prefixes extend easily to the real variation occurring in genomes. Reverse complementation is easy to take into account as well, by adding the reverse complement of the base sequence to the collection.

The base structures (e.g. RLFM+ index) for counting queries are universal in the sense that they do not need to know what and where the mutations are. Standard sampling techniques can be used to add reasonably efficient support for locating and displaying, as shown in Table 1 and Fig. 3. Compressed Suffix Array (CSA) [20], Succinct Suffix Array (SSA) [125], and Run-Length FM-index (RLFM) [12] are existing indexes similar to our RLCSA, RLWT, and RLFM+, respectively.

The experiments were performed on a 2.66 GHz Intel Core 2 Duo system with 3 GB of RAM running Fedora Core 8 based Linux. Counting and locating times are averages over 1000 patterns of length 10. Displaying a substring of length  $l$  requires the extraction of  $d/2 + l$  characters on the average.

RLWT and RLFM+ currently store the suffix array samples in a more space efficient manner than RLCSA. By using the same method for all three indexes, the size differences between them would be determined only by the sizes of base structures for counting. On the other hand, `locate()` in RLCSA and RLWT has been optimized for retrieving multiple occurrences in parallel. As the base structures are run-length encoded, we can perform the base step (*LF*-mapping in RLWT) for an entire run for roughly the same cost as for a single occurrence. If similar optimizations were implemented in RLFM+, its locating speed would be close to that of RLCSA.



**Fig. 3.** Sizes and times for `locate()` for self-indexes on the *S. paradoxus* collection. Each index was tested with sampling rates  $d = 32, 128$ , and  $512$ . Reported times are in microseconds / occurrence.

The new structures for `display()` and `locate()` require the alignment of each sequence with the base sequence to be given; for succinctness we considered the easy case of identical length sequences and point mutations (where the alignment is trivial to compute). The structures are easy to extend to more general alignments. Our current implementation supports alignments with gaps (i.e. runs of Ns) as well as insertions and deletions in addition to substitutions.

The main component required is the static structure supporting persistent selection. For its construction, we implemented also the dynamic structure supporting online persistent selection (with minor modifications it would support confluent persistent selection as well). Once it is constructed for the given alignment, it is converted into a static structure. The static structure is in fact more space-efficient than the one described in Theorem 2, as we discard completely the tree structure and instead concatenate levelwise the two bitvectors stored at the nodes of the tree; a third bitvector is added marking the leaves, which enables us to navigate in the tree whose nodes are now represented as ranges. The time-to-rank mapping in the root of the persistent tree can be stored space-efficiently using the BSD representation. The space requirement is  $6x \log x(1 + o(1)) + x \log \frac{h}{x}(1 + o(1)) + O(x \log \log \frac{h}{x}) + x \log r$  bits, where  $6x \log x(1 + o(1))$  comes from the 3 bitvectors of length  $x$  supporting `rank` and `select` on each of the at most  $2 \log x$  levels of the red-black balanced tree,  $x \log \frac{h}{x}(1 + o(1)) + O(x \log \log \frac{h}{x})$  comes from the BSD representation, and  $x \log r$  from the values stored at leaves.

The interesting question is at which mutation rates the persistent selection approach will become competitive with the standard sampling approach. With the mutation rates occurring in yeast collection of Fig. 3, the persistent selection approach does not seem to be a good choice; it occupied 8.49 MB on the 36 strains

**Table 2.** Standard sampling versus persistent selection. The rows give the size of the base structure (RLWT), size of suffix array samples, size of display structures, size of persistent selection structure including bookkeeping of zones, and the total size.

Approach/Size	Mutation rate 0.001		Mutation rate 0.0001	
	Standard	Persistent	Standard	Persistent
Base (MB)	4.06	4.06	2.19	2.19
Samples (MB)	1.24	0.28	1.02	0.25
Display (MB)		0.32		0.07
Persistent (MB)		3.22		0.31
Total size (MB)	5.30	7.89	3.21	2.82

of *S paradoxus* chromosome 2 (28.67 MB), while RLWT with standard sampling approach occupied 3.97 MB.<sup>5</sup> The sampling parameters were chosen so that both approaches obtained similar time efficiency (403 versus 320 microseconds for one locate, respectively). To empirically explore the turning point where the persistent selection approach becomes competitive, we generated a DNA sequence collection with 100 copies of a 1 MB reference sequence and applied different amount of random mutations on it. Table 2 illustrates the turning point by giving the space requirements for RLWT+sampling versus RLWT+persistent selection on two different mutation rates, where their order changes. We used sampling rate  $d = 512$  for standard sampling, and  $d = 64$  ( $d = 32$ ) for persistent selection approach on mutation rate 0.001 (on mutation rate 0.0001). This made the running times reasonably close; for example, one locate took 172 versus 184 microseconds on 0.0001 mutation rate, respectively.

## 4 Conclusions

We have studied the problem of representing highly repetitive sequences in such a way that their repetitiveness is exploited to achieve little space, yet at the same time any part of the sequences can be extracted and searched without decompressing it. This problem is becoming crucial in Computational Biology, due to the cheaper and cheaper availability of sequence data and the interest in analyzing it.

We have shown that the current compressed text indexing technology is not well suited to cope with this problem, and have devised variants that have shown to be much more successful.

In the full paper we will show how to allow for dynamism, that is, permitting to handle a compressed collection where insertion and deletion of sequences can be efficiently intermixed with searches. We achieve the same space bounds, whereas the time requirements are multiplied roughly by a logarithmic factor.

An important challenge for future work is to look for schemes achieving further compression. For example, LZ77 algorithm is an excellent candidate to compress

<sup>5</sup> We observed that the given multiple alignments were not the best possible; the size would be reduced significantly by the choice of better consensus sequences.

repetitive collections, achieving space proportional to the number of mutations. For example, the 409 MB collection of *Saccharomyces paradoxus* strains studied here can be compressed into 4.93 MB using an efficient LZ77 implementation<sup>6</sup>. This is over 7 times less space than what the new self-indexes studied in this paper achieve. Yet, LZ77 has defied for years its adaptation to a self-index form. Thus there is a wide margin of opportunity for such a development.

## Acknowledgments

We wish to thank Teemu Kivioja from Institute of Biomedicine, University of Helsinki, for turning our attention to the challenges of individual genomes. We wish also to thank Kimmo Palin from Sanger Institute, Hinxton, for pointing us the yeast genome collection.

## References

1. Blanford, D., Blelloch, G.: Compact representations of ordered sets. In: Proc. 15th SODA, pp. 11–19 (2004)
2. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Technical Report Technical Report 124, Digital Equipment Corporation (1994)
3. Church, G.M.: Genomes for all. *Scientific American* 294(1), 47–54 (2006)
4. Ferragina, P., Manzini, G.: Indexing compressed texts. *Journal of the ACM* 52(4), 552–581 (2005)
5. Ferragina, P., Manzini, G., Mäkinen, V., Navarro, G.: Compressed representations of sequences and full-text indexes. *ACM Transactions on Algorithms (TALG)* 3(2) article 20 (2007)
6. Fischer, J., Mäkinen, V., Navarro, G.: An(other) entropy-bounded compressed suffix tree. In: Ferragina, P., Landau, G.M. (eds.) CPM 2008. LNCS, vol. 5029, pp. 152–165. Springer, Heidelberg (2008)
7. Grossi, R., Vitter, J.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing* 35(2), 378–407 (2006)
8. Gupta, A., Hon, W.-K., Shah, R., Vitter, J.S.: Compressed data structures: Dictionaries and data-aware measures. In: DCC 2006: Proceedings of the Data Compression Conference (DCC 2006), pp. 213–222 (2006)
9. Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
10. Hall, N.: Advanced sequencing technologies and their wider impact in microbiology. *The Journal of Experimental Biology* 209, 1518–1525 (2007)
11. Kaplan, H.: Persistent Data Structures. In: Mehta, D.P., Sahni, S. (eds.) *Handbook of Data Structures and Applications*, vol. 31. Chapman & Hall, Boca Raton (2005)
12. Mäkinen, V., Navarro, G.: Succinct suffix arrays based on run-length encoding. *Nordic Journal of Computing* 12(1), 40–66 (2005)
13. Manber, U., Myers, G.: Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.* 22(5), 935–948 (1993)

---

<sup>6</sup> <http://p7zip.sourceforge.net/>

14. Manzini, G.: An analysis of the Burrows-Wheeler transform. *Journal of the ACM* 48(3), 407–430 (2001)
15. Navarro, G., Mäkinen, V.: Compressed full-text indexes. *ACM Computing Surveys* 39(1) article 2 (2007)
16. Overmars, M.H.: Searching in the past, i. Technical Report Technical Report RUU-CS-81-7, Department of Computer Science, University of Utrecht, Utrecht, Netherlands (1981)
17. Pennisi, E.: Breakthrough of the year: Human genetic variation. *Science* 21, 1842–1843 (2007)
18. Russo, L., Navarro, G., Oliveira, A.: Dynamic fully-compressed suffix trees. In: Ferragina, P., Landau, G.M. (eds.) CPM 2008. LNCS, vol. 5029, pp. 191–203. Springer, Heidelberg (2008)
19. Russo, L., Navarro, G., Oliveira, A.: Fully-compressed suffix trees. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 362–373. Springer, Heidelberg (2008)
20. Sadakane, K.: New text indexing functionalities of the compressed suffix arrays. *Journal of Algorithms* 48(2), 294–313 (2003)
21. Sadakane, K.: Compressed suffix trees with full functionality. *Theory of Computing Systems* 41(4), 589–607 (2007)
22. Sirén, J., Välimäki, N., Mäkinen, V., Navarro, G.: Run-length compressed indexes are superior for highly repetitive sequence collections. In: Amir, A., Turpin, A., Moffat, A. (eds.) SPIRE 2008. LNCS, vol. 5280, pp. 164–175. Springer, Heidelberg (2008)
23. Waterman, M.S.: Introduction to Computational Biology. Chapman & Hall, University Press (1995)

# An Online Approach for Mining Collective Behaviors from Molecular Dynamics Simulations

Arvind Ramanathan<sup>1</sup>, Pratul K. Agarwal<sup>2</sup>, Maria Kurnikova<sup>3</sup>,  
and Christopher J. Langmead<sup>1,4,\*</sup>

<sup>1</sup> Lane Center for Computational Biology, Carnegie Mellon University

<sup>2</sup> Computational Biology Institute, and Computer Science and Mathematics Division,  
Oak Ridge National Laboratory

<sup>3</sup> Chemistry Department, Carnegie Mellon University

<sup>4</sup> Computer Science Department, School of Computer Science, Carnegie Mellon University  
`cjl@cs.cmu.edu`

**Abstract.** Collective behavior involving distally separate regions in a protein is known to widely affect its function. In this paper, we present an online approach to study and characterize collective behavior in proteins as molecular dynamics simulations progress. Our representation of MD simulations as a stream of continuously evolving data allows us to succinctly capture spatial and temporal dependencies that may exist and analyze them efficiently using data mining techniques. By using multi-way analysis we identify (a) parts of the protein that are dynamically coupled, (b) constrained residues/ hinge sites that may potentially affect protein function and (c) time-points during the simulation where significant deviation in collective behavior occurred. We demonstrate the applicability of this method on two different protein simulations for barnase and cyclophilin A. For both these proteins we were able to identify constrained/ flexible regions, showing good agreement with experimental results and prior computational work. Similarly, for the two simulations, we were able to identify time windows where there were significant structural deviations. Of these time-windows, for both proteins, over 70% show collective displacements in two or more functionally relevant regions. Taken together, our results indicate that multi-way analysis techniques can be used to analyze protein dynamics and may be an attractive means to automatically track and monitor molecular dynamics simulations.

## 1 Introduction

With the proliferation of structural information for over 50,000 proteins, a systematic effort to understand the relationship between a protein's three-dimensional structure, dynamics and function is underway. Molecular dynamics (MD) / Monte-Carlo (MC) simulations have become standard tools to gain insight into fundamental behavior of protein structures [30]. With increasing computational power, and the development of specialized hardware and software for MD simulations such as Desmond [15] simulations now easily scale to tens or even hundreds of nanoseconds regularly. The data from these simulations can easily reach several terabytes. Therefore, efficient methods to store, process

---

\* Corresponding Author.

and analyze this data are needed. There is also a growing interest for development of tools that monitor and track MD simulations, such that rare events within a protein simulation (e.g. a protein undergoing a conformational change) can be automatically detected [38].

Collective behavior in a protein refers to a group of amino-acid residues that may be spatially separate yet exhibit similar dynamics [13]. The similarity in dynamics refers to whether a group of residues are *constrained*, i.e. exhibiting small variance in distances with respect to other residues in the protein, or *flexible*, i.e., showing large variance in distances. Often residues at the interface of constrained/ flexible regions, known as *hinge-sites*, affect protein dynamics and function [22]. Collective behavior in a protein has been assessed primarily using techniques such as principal component analysis (PCA) [31][29][10]. Most techniques use a static structure (single snapshot) to analyze intrinsic dynamics and reason about collective behavior [11]. Other techniques use a collection of snapshots from a MD trajectory and perform PCA post-process [26]. The scientific community does not yet possess an efficient technique that provides information on collective behavior in a protein as the simulation is progressing. There are also no automated ways to track and monitor MD simulations on the basis of collective behavior observed in a protein.

This paper describes an *online* approach to characterize the collective behavior within proteins as MD simulations are progressing. In our approach, protein structures (snapshots) from a MD trajectory are modeled as a multi-dimensional array or tensor. This representation allows us to capture both spatial and temporal dependencies simultaneously as the simulation is evolving. Using recent advances in tensor analysis and data-mining we show that one can succinctly capture the dynamical behavior of a protein over the simulation. This dynamical behavior captured can be used to (a) conveniently visualize clusters within a protein that exhibit coupled motions or collective behavior, (b) identify residues that may play a significant role in protein's dynamics and (c) identify time-points during a simulation which exhibit a significant deviation from normal behavior of the protein.

Our contributions in this paper introduce a novel representation of protein simulations as streaming data. An approach to mine streaming data allows us to reason about parts of a protein that are more flexible versus parts of the protein that are less flexible. The characterization of flexible/ constrained regions in the protein match well with experimental and prior computational work. We also identify time-points during a simulation where there have been significant changes in the protein's dynamical behavior. The identification of such time-points can be potentially used to fork-off other simulations that may lead to better sampling of the protein's conformational space. Taken together, our approach shows that it is possible to reason about collective behavior as simulations are progressing and this may be of immense use to scientists wanting to understand complex phenomena in protein structures.

## 2 Tensor Representation of Protein Structures and MD Simulations

Tensors are an extension of matrices beyond two dimensions and provide a convenient way to capture multiple dependencies that may exist in the underlying data.

Formally, a tensor  $\mathcal{X}$  of  $M$  dimensions can be defined as a multi-dimensional array of real values,

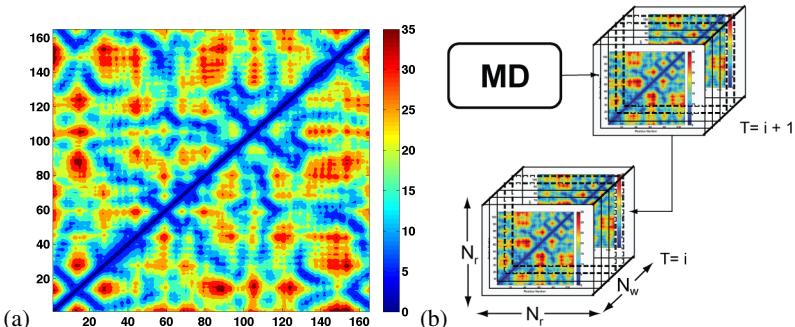
$$\mathcal{X} \in \Re^{N_1 \times N_2 \times \dots \times N_M} \quad (1)$$

where  $N_i$  represents the  $i^{th}$  dimension for  $(1 \leq i \leq M)$ . A discussion of possible operations on a tensor is beyond the scope of this report, however, a review of tensors and related operations are provided in Kolda, et al [32]. In what follows, tensors are represented with calligraphic letters (e.g.  $\mathcal{X}$ ), matrices by bold capital letters(e.g.  $\mathbf{X}$ ), vectors as bold small letters ( $\mathbf{x}$ ) and specific instances as normal text ( $x$ ).

In a protein, apart from spatial dependencies, an explicit temporal dimension is needed to study the evolution of collective behavior over time. A protein's spatial description can be captured as a *distance map*. A distance map is a matrix where each entry  $(i, j)$  is the distance between atom  $i$  and  $j$ , defined as follows:

$$\mathbf{C}_{ij} = \sqrt{(\mathbf{r}_i - \mathbf{r}_j)^2} \quad (2)$$

where,  $\mathbf{C}_{ij}$  is a second order tensor (matrix),  $\mathbf{r}_i$  represents the position vector of atom  $i$ . For simplicity, we chose to represent the distances between  $C^\alpha$  atoms, bringing the total number of entries in the  $\mathbf{C}$  matrix to be  $N_r \times N_r$ , where  $N_r$  is the number of residues in the protein. Note that by representing the distances between all  $C^\alpha$  atoms, we are able to account for local (immediate neighborhood) as well as global (over the entire protein) dependencies that may exist. An example distance map for the protein cyclophilin A (pdb id: 1AWQ) is shown in Fig. 1(a). There are distinct advantages of such a simple representation: one, distance maps are independent of rotations which may happen during the course of the simulation and two, distance maps also capture information about the entire conformation of a protein.



**Fig. 1. Distance matrix and Tensor Representation of MD simulations.** (a) Distance map representation of the enzyme cyclophilin A (pdb: 1AWQ). Distant residues are identified via darker shades of red. Distances are in Å (b) shows the streaming representation of MD simulations used in this paper. As new tensors keep arriving at every time interval  $T = i + 1$ , they are appended to the end of the current stream  $T = i$ .  $N_r$  is the number of residues,  $N_w$  represents the size of the window. This can be set by the end user depending on how often the user wants the analysis to run.

To capture temporal dependencies, we note that an MD simulation updates the coordinate positions at every time step  $t$ . An entire MD simulation can be thought of as a discrete collection of the distance maps  $\mathbf{C}(t)$  defined above. We may also define a discrete window in time, such that a time-slice from the MD simulation constitutes a third order tensor, with dimensions  $N_r \times N_r \times N_w$ , where  $N_r$  is the number of residues in the protein and  $N_w$  is the size of the window. A time-slice representation provides us with a mechanism to track collective behavior at short time scales and also at the time scale of the entire simulation.

If we define a discrete sized time window of size  $w$ , then the entire simulation can be considered a collection of  $T = n/w$  tensors, where  $n$  is the total number of snapshots in the simulation. An ordered collection of such tensors is commonly referred to as a *tensor stream*. Formally a tensor stream is a discrete collection of  $M^{th}$  order tensors  $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_T$  where  $T$  is an integer that increases with time. The tensor stream representation used for MD simulation is illustrated in Fig. 1(b). Each tensor represents a discrete time window within the simulation. As the MD simulation progresses, more and more tensors become available, and the new tensors are appended to the end of the tensor stream.

### 3 Tensor Analysis of Protein Dynamics

A simple way to detect patterns is to analyze the overall variance in the underlying data. In two dimensions, PCA identifies patterns by minimizing the least-squared error with respect to the overall variance observed in the data. For tensors, it is possible to use an extension of PCA in multiple dimensions commonly referred to as *tensor analysis*. The objective function in tensor analysis is very similar to that of PCA - we minimize the error with respect to the observed variance in every one of the  $M$  dimensions. Formally, given a collection of tensors  $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_T$ , each of dimension  $N_1 \times N_2 \times \dots \times N_M$ , tensor analysis will determine orthogonal matrices  $\mathbf{U}_i$  for each dimension  $N_i$  such that the reconstruction error  $e$  is minimized as follows:

$$\boxed{e} = \sum_{t=0}^T \|\mathcal{X}_t - \mathcal{X}_t \prod_{i=1}^M \times_i (\mathbf{U}_i \mathbf{U}_i^T)\|_F^2 \quad (3)$$

The operation  $\mathcal{X}_t \prod_{i=1}^M \times_i (\mathbf{U}_i \mathbf{U}_i^T)$  yields the approximation of  $\mathcal{X}_t$  spanned by the orthogonal matrices  $\mathbf{U}_i$ . The orthogonal matrices so determined will also reveal any underlying patterns that may be present within the data.

If all of the simulation data is already available, we may use multi-dimensional PCA to obtain insights into what parts of the protein are flexible or what parts of the protein are held relatively rigid. However, our goal is to characterize collective behavior in a dynamic environment such as MD. A recent algorithm called Dynamic Tensor Analysis (DTA) [42] has been successfully used to extract patterns from time-evolving data. This

<sup>1</sup>  $\|\mathcal{X}\|_F^2$  is the square Frobenius Norm and is defined as  $\|\mathcal{X}\|_F^2 = \sum_{i_1=1}^{N_1} \dots \sum_{i_M=1}^{N_M} \mathcal{X}(i_1, \dots, i_M)^2$ .

<sup>2</sup>  $\mathcal{X} \prod_{i=1}^M \times_i \mathbf{U}_i$  represents tensor multiplied by a set of matrices, defined as  $\mathcal{X} \times_1 \mathbf{U}_1 \dots \times_M \mathbf{U}_M$ .

algorithm exploits two main features of the underlying data: (a) it is possible to quickly compute PCA in two dimensions if the variance matrices are available and (b) one can update the variance matrices *incrementally* without having to store any previous history. These two observations are particularly relevant for MD simulations: we compute only variance for pairwise distances, which is easy to compute. MD simulations are ergodic and therefore no historical information needs to be assessed. The algorithm we use to perform online analysis of MD simulations is outlined in Algorithm 1.

---

**Algorithm 1.** Online Analysis of Molecular Dynamic Simulations

---

```

1: for every incoming MD data at time-window  $T$  do
2:   Convert MD data into tensor  $\mathcal{C}^{(T)}$  of dimension  $d = N_r \times N_r \times N_w$ 
   {/* Perform Tensor Analysis using DTA */}
3:   for  $i = 1$  to  $d$  do
4:     Matricize  $\mathcal{C}^{(T)}$  as  $\mathbf{C}_{(i)}^{(T)} \in \Re^{(\prod_{j \neq d} N_i) \times N_i}$ 
5:     Reconstruct variance  $\mathbf{V}_{(i)}^{(T-1)} \leftarrow \mathbf{U}_i^{(T-1)} \mathbf{S}_i^{(T-1)} (\mathbf{U}_i^{(T-1)})^T$ 
6:     Update variance matrix  $\mathbf{V}_{(i)}^{(T)} \leftarrow \mathbf{V}_{(i)}^{(T-1)} + (\mathbf{C}_{(i)}^{(T)})^T \mathbf{C}_{(i)}^{(T)}$ 
7:     Diagonalize  $\mathbf{V}_{(i)}^{(T)} \leftarrow \mathbf{U}_i^{(T)} \mathbf{S}_i^{(T)} (\mathbf{U}_i^{(T)})^T$ 
8:   end for
9:   Calculate core tensor:  $\mathcal{Y} = \mathcal{C}^{(T)} \prod_{i=1}^d \times \mathbf{U}_i^{(T)}$ 
   {/* Identify restrained residues */}
10:  Compute  $\mathbf{f}_j = \max(\mathbf{U}_i^{(T)}[j, :])$  to find maximum distance variance
11:  if  $\mathbf{f}_j \leq \text{mean}(\mathbf{f}_j) - \text{std}(\mathbf{f}_j)$  then
12:    Identify residue  $j$  as restrained
13:  end if
   {/* Identify rigid and flexible regions */}
14:  Use  $k$ -means algorithm to cluster  $\mathbf{U}_r^{(T)}$ .
15:  Map output of  $k$ -means onto the protein structure to identify dynamically coupled regions
   {/* Identify time points of interest */}
16:  if  $e_T \geq \text{mean}(e_i|_{i=1}^T) + \alpha \cdot \text{std}(e_i|_{i=1}^T)$  then
17:    Identify  $T$  as an “event of interest”
18:  end if
19: end for

```

---

A detailed description of DTA is provided in [42]; a brief description of DTA is given here (lines 3 to 9 in Algorithm 1). The algorithm proceeds by minimizing the variance in every dimension  $i$ , ( $1 \leq i \leq M$ ). The tensor  $\mathcal{C}$  is matricized<sup>3</sup> in the selected dimension, say  $d$  to obtain matrix  $\mathbf{C}_{(d)}$ . Then, using the previous orthogonal matrices at time  $T - 1$ ,  $\mathbf{U}_d$ , the variance matrix is reconstructed to obtain  $\mathbf{V}_d \leftarrow \mathbf{U}_d \mathbf{S}_d \mathbf{U}_d^T$ . Using the matrix  $\mathbf{C}_{(d)}$ , we now update the variance matrix at time  $T$  in the  $d^{th}$  dimension as:  $\mathbf{V}_d \leftarrow \mathbf{V}_d + \mathbf{C}_{(d)} \mathbf{C}_{(d)}^T$ . We now diagonalize the updated variance matrix  $\mathbf{V}_d$  to obtain the new projection matrices  $\mathbf{U}_d$  and  $\mathbf{S}_d$ . In order to capture a succinct representation of the data seen so far, we also construct a core tensor using  $\mathcal{Y} = \mathcal{C} \prod_{i=1}^M \times \mathbf{U}_d$ .

### 3.1 Outputs of DTA

The orthogonal matrices  $\mathbf{U}_i$  describe the underlying correlations within the data. The first two orthogonal matrices are identical (since the data is symmetric). Both these ma-

<sup>3</sup> Matricizing in dimension  $d$  is the process of unfolding a  $M^{th}$  dimension tensor  $\mathcal{C}$  obtained by keeping  $d$  fixed and varying all other dimensions. Thus we obtain a vector in  $\Re^{N_d}$ , represented as  $\mathbf{C}_{(d)} \in \Re^{(\prod_{i \neq d} N_i) \times N_d}$ .

trices ( $\mathbf{U}_r$ ) describe the inter-residue distance variance in time. The criterion for identifying rigid vs. flexible residues is shown in Algorithm 1 (lines 10 through 13). Lower values along the rows of the orthogonal matrices imply the residue is rigid, whereas, higher values in the orthogonal matrices imply that the residue is flexible. The third orthogonal matrix represents variances in time. This is used to identify time-points along the simulation where there is significant deviation in the protein's collective behavior.

It is also possible to cluster the orthogonal matrices to obtain insights into how different parts of the protein may be dynamically coupled (Algorithm 1; lines 14 and 15). Distance variance is a measure for how far two residues moved during the simulation with respect to each other. The correlations from the underlying variance must provide quantitative view of how much each residue moved with respect to its immediate neighbors as well as the entire protein. To visualize this, we used  $k$ -means clustering [25] to identify clusters of rigid/ flexible residues. Residues within the same cluster must exhibit similar fluctuations in distances. The individual cluster centers identify the average distance fluctuations within each cluster; larger distance fluctuations imply flexible regions, whereas smaller distance fluctuations imply rigid regions in the protein.

In order to identify time points where there are significant deviations from the normal dynamical behavior, we use the reconstruction error metric defined in Eqn. 3 (Algorithm 1, lines 16 through 18). This metric shows how different the incoming tensor was compared to the previous ones in the list. If the reconstruction error at time  $T$  is above  $\alpha$  standard deviations from the mean reconstruction error observed thus far, we define  $T$  to be an event of interest. A formal definition of this threshold is given below:

$$e_T \geq \text{mean}(e_i|_{i=1}^T) + \alpha.\text{std}(e_i|_{i=1}^T) \quad (4)$$

### 3.2 Related Work

A number of techniques have been developed to analyze and reason about collective behavior from MD simulations. PCA based techniques (reviewed in [13]) are very popular in the MD community and are used to visualize collective behavior in proteins. Other techniques based on spectral analysis [16] as well as mutual information [34][33] are also widely used in studying collective motions. However, there are two limitations to these approaches.

The first limitation relates to the fact that PCA can be done only in *two dimensions*; hence it is possible to obtain only insights into collective behavior from the perspective of spatial variance, no insight can be drawn from the temporal aspects of the simulation. This limitation can be overcome by using multi-dimensional PCA or multi-way analysis [32]. Multi-way analysis has been applied to assign nuclear magnetic resonance (NMR) spectra [35][40]. These techniques have not been used to study protein dynamics to our knowledge, however they are popular within the cheminformatics community [24]. Tensor analysis has also been applied in a variety of problem domains including visual analysis of images from electroencephalogram [1] and systems biology [46]. The identification of rigid/ flexible sub-structures within a protein is a well studied problem, especially from the viewpoint of rigidity theory [44][28]. Application of these techniques to multiple snapshots of proteins proves to be unreliable [36] and similarly, it is not possible to obtain information about time-points where deviation in collective behavior is observed.

The second limitation arises from the fact that analysis techniques developed for MD data can be applied *post-process*. The online techniques that are available provide only minimal feedback - either in terms of displacement vectors or individual snapshots from the simulation [12] and the end user is responsible to check whether these snapshots are of further interest; or improve performance of simulations [23]. By treating MD simulations as streaming applications, we have been able to apply techniques from data-mining to provide automated feedback to the end-user about how the behavior of the protein has changed with respect to time.

## 4 Implementation and Results

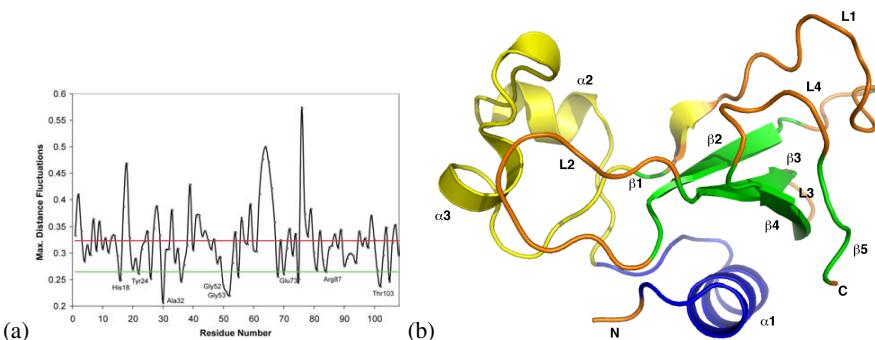
We were provided access to a set of MD simulations from two independent groups. This was done in order to ensure good variability in the available data and also to test the applicability of the method on different types of simulations. The first set of simulations constituted an equilibrium simulation for a 108 residue protein - barnase [36], with a total length of 10 nanoseconds (2,500 snapshots). The second set of simulations consisted of a free-energy profile for an enzymatic reaction catalyzed by the enzyme cyclophilin A (172 residues) [5], with a total of 18,000 snapshots. Both proteins have been studied extensively using both experimental and theoretical/computational techniques and are ideal to test the utility of the technique outlined above. We first processed the MD trajectories to obtain the distance maps for every individual snapshot.

Third order tensors were then constructed with every 10 snapshots ( $N_w = 10$ ) aggregated into one tensor, thus providing a total of 250 tensors for barnase. For cyclophilin A, we chose evenly spaced snapshots to yield 1800 snapshots to yield 180 tensors. The window size can be provided by the user; in several situations it is desirable to set larger window sizes. It is also dependent on the end-user how often the user would like to run the analysis. The tensors were processed using the tensor toolkit libraries [78] and DTA algorithm [42] in MATLAB.

### 4.1 Analysis of Collective Behavior in Proteins

A plot of the rigid versus flexible residues for the protein barnase is shown in Fig. 2(a). An examination of the plot reveals that certain residues show lower distance variance within the protein. A low distance variance implies that the residue has a lesser degree of freedom in terms of moving around compared to the other residues and hence, is rigid. We considered those residues below one standard deviation interval from the plot. A total of 15 out of 108 residues were found to be constrained. The maximally constrained residues namely Tyr24, Ala32, Gly52, Gly53, Arg87 and Thr103 are known to be important for barnase's folding with Ala30 being a nucleation site [19]. Similarly, Gly52 and Gly53 form a hinge-site of the protein, implicated in the stabilization of the protein's motions [37]. It is also interesting to note that the catalytically important residue Glu71 is also constrained [20][21], however Lys25 is not constrained to the extent of the residues that are implicated in protein folding or the functional hinge-site.

We now look at clusters of residues that are dynamically coupled. Using  $k$ -means clustering on the orthogonal matrices  $\mathbf{U}_r$ , we identified a total of four clusters. The

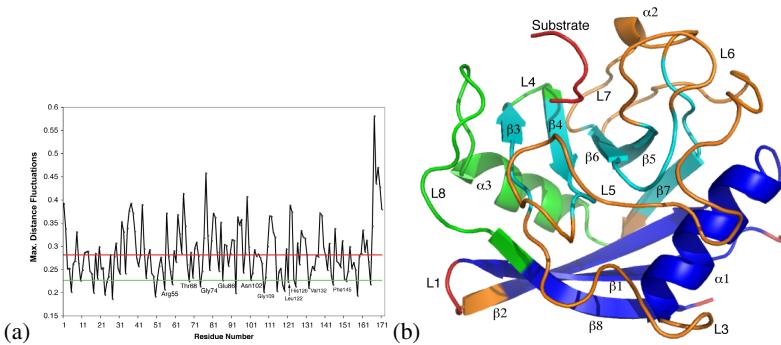


**Fig. 2. Barnase constrained residues and flexible regions.** (a) shows constrained residues in barnase. The red line indicates the mean distance variance and the green line indicates the first standard deviation interval below the mean value. Residues that are constrained are marked on the plot. Only a few residues are marked for clarity; also note that the first two residues were not present in the simulation. (b) indicates flexible clusters in barnase. Four clusters are identified;  $\alpha_1$  and  $\beta_1-\beta_5$  form two clusters shown in blue and green form the hydrophobic core of the protein, showing low distance variance.  $\alpha_2-\alpha_3$  (residues 21-48) shown in yellow forms a separate functional domain and loops L1-L4 form the most flexible parts of the protein.

selection of the best value of  $k$  for clustering was based on the mean value of cluster separation; for any value of  $k$ , if the mean value of the cluster separation was higher than that of  $k - 1$ , another round of  $k$ -means clustering was applied, otherwise,  $k$  was chosen to be the optimal number of clusters. The assignment of clusters obtained from  $k$ -means was then mapped onto the three-dimensional structure of the protein and visualized using PyMOL [17]. As shown in Fig. 2(b),  $\alpha_1$  and  $\beta$ -sheet ( $\beta_1 - \beta_5$ ) form separate clusters. Residues 21-48 formed of  $\alpha_2$  and  $\alpha_3$  show higher flexibility than the hydrophobic core of the protein and is known to be a functional domain in the protein [45]. The residues 49-52 are clustered into three different groups. These residues form a functional hinge-site of the protein [45][21]. The loop regions (L1-L4; N- and C-termini) are identified to be very flexible.

For the enzyme Cyclophilin A (Fig. 3(a)), we were able to identify two distinct groups of residues; the first set of residues consisted of the enzyme itself. A second set of residues consisting of higher distance variance than the enzyme turned out to be the substrate bound to the enzyme. Of the enzyme residues, we identified a total of 24 constrained residues. Of these Arg55, Thr68, Gly74, Asn102 and Gly109, Leu122, His126, Val132 and Phe145 are part of a connected network of interactions identified in previous experimental [18] and computational work [5][2] and are known to be of functional importance.

Based on our approach, we were able to identify six dynamically coupled regions within Cyclophilin A (Fig. 3(b)). Of these, the substrate and the N/C- termini of the protein formed a separate cluster. The  $\beta$ -sheet in the protein clustered into two different regions; the first cluster (shown in blue;  $\beta_{1-2}, \beta_8$ ) represents the hydrophobic core of the protein. The second cluster (shown in cyan;  $\beta_{3-7}$ ) represents the surface region of the protein in close contact with the substrate. The  $\alpha$ -helices form two clusters; one constrained via hydrophobic interactions ( $\alpha_1$ ) and held rigid whereas the second helix ( $\alpha_3$ ) exhibits much more flexibility and coupled with the loop at the active site of the



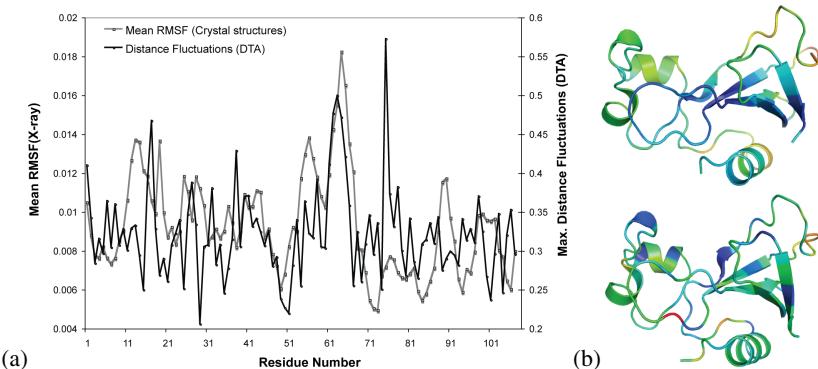
**Fig. 3. Cyclophilin A constrained residues and flexible regions.** (a) Constrained residues below the first standard deviation interval (green line) form a conserved network of interactions in the enzyme known to affect catalytic function [5]. (b) shows regions of the protein that are dynamically coupled; the hydrophobic part formed by  $\beta 1$ -2, 8 and  $\alpha 1$  experience low distance variance,  $\beta 3$ -7 undergo slightly larger distance variance, L4 and L8 are grouped into one cluster, L3-7 are indeed very flexible whereas loop L1 and the substrate form the most flexible part of the protein.

protein (L4, L7; shown in green). Note that the loops that belong to the same cluster show high correlations as noted in previous studies [5].

## 4.2 Comparing with Experimental/ Theoretical Work

A large amount of structural information for barnase is available within PDB [14]. Of the 55 structures available, we used a total of 6 structures (without mutations) for comparing the root mean square fluctuations (RMSF) to that of the distance fluctuations obtained from DTA. The RMSF values were normalized to compare the different RMSF values, as shown in Fig. 4(a). Note that there is a close correspondence in the location of the hinge site as well as the flexible regions of the protein. Further more, plotting the RMSF (Fig. 4(b) top panel) as well as DTA determined distance fluctuations (Fig. 4(b) bottom panel), we observe a close correspondence in the way distance fluctuations reflect the dynamic behavior of barnase. It is also important to note that many of the structures had the protein's substrate bound to it, hence small differences are visible in terms of the location of flexible regions; note that residues 71-81 are more flexible from our simulations.

We were able to compare our results with previously published work based on Gaussian Network Model (GNM) [9], MD simulations and NMR experiments [37][48]. As illustrated in Fig. 5(a), an overlap of the root mean squared fluctuations (RMSF) determined from GNM and distance fluctuations computed via DTA from the MD simulation, show considerable agreement in the location of hinge site (Gly52/53). The identification of flexible loop regions also show good agreement. The residues 21-48 exhibited different dynamical properties compared to the rest of the protein and hence this region was clustered into a separate region via DTA. Previous experimental work [37][48] also indicates that these residues are indeed dynamically distinct unit, forming a separate domain.

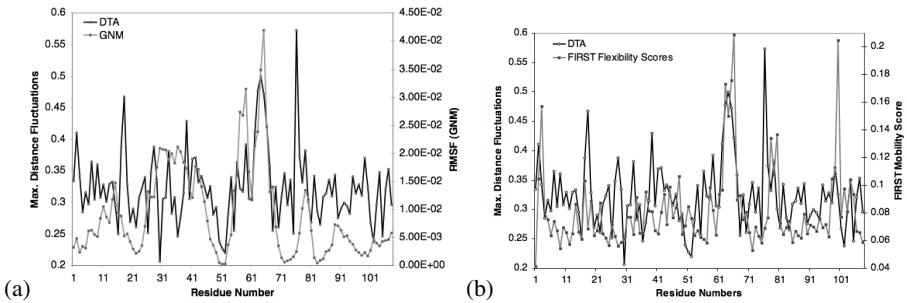


**Fig. 4. Baranase Experimental versus DTA comparisons.** (a) DTA distance fluctuations are plotted (black solid line) against the average root mean square fluctuations (RMSF) determined from 6 different crystal structures of barnase. (b) shows regions of barnase that are flexible determined from X-ray crystallography (average B-factors of 6 structures). Note the close correspondence in the location of constrained regions. All the constrained residues identified from Fig. 2(a) are also constrained from the X-ray determined RMSF. Particularly, at the hinge site (48-51) as well as the flexible loop regions, there is good agreement. There is some disagreement in terms of the flexibility observed from the crystal structures in regions 71-81; this is mainly because the crystal structures had the substrate bound.

GNM uses the graph laplacian to predict positional fluctuations of residues based entirely on their topology. However, DTA's approach allows it to track distance variations observed during MD simulations and provides insights into constrained/ flexible regions. Note that while we have used only  $C^\alpha$  distances as input to DTA, the method is quite general and can be used to track even hydrogen bond donor-acceptor distances/ hydrophobic interactions.

We were also able to compare for rigid/ mobile regions within barnase using the program Floppy Incursions and Rigid Sub-structure Topography (FIRST) [28][27][36]. An overlap of the flexibility scores determined via the Tool for Identifying Mobility in Macromolecular Ensembles (TIMME) as part of the FIRST suite of programs is shown in Fig. 5(b). Note that we used the same MD trajectory that we used for DTA as input to the program. A casual inspection of the plot indicates that there is good agreement between flexible regions of the protein. We also compared the rigid clusters formed by FIRST for barnase from previous work [28][27][36] and found that the largest rigid cluster identified by FIRST consisting of  $\alpha 1$  helix and  $\beta 1-\beta 5$  indeed exhibited overall smaller distance fluctuations in DTA. We also note that comparing our results with that of FIRST for cyclophilin A [47], we found good agreement in terms of the location of flexible versus constrained regions in the enzyme.

Note that FIRST can only show what parts of a protein tend to be constrained and cannot distinguish domains on the basis of dynamics exhibited by a protein. FIRST uses a local constraint counting approach to estimate the flexibility/ rigidity of a particular region within a protein and thus, cannot make predictions about global constraints that may influence distally separated regions in the protein. On the other hand, DTA includes the notion of distance dependency between every  $C^\alpha$  atom and hence global



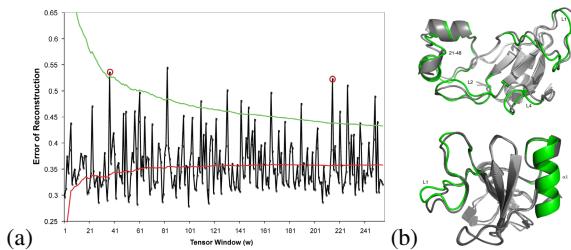
**Fig. 5. Comparison of DTA to GNM and FIRST.** (a) The predicted root mean squared fluctuations (RMSF) determined via GNM is plotted along with the distance fluctuations determined from DTA. Note the correspondence in flexible regions versus constrained regions. Closely agreeing regions observed from DTA and GNM are between 41-71, 76-84. Other regions show low positional fluctuations mostly depending on how they are connected to their neighbors. This is true especially with the secondary structure regions of the protein. (b) Constrained residues identified via FIRST and DTA. The flexible regions show close correspondence in both the methods, however, FIRST based flexibility scores do not correspond well with the hinge site of the protein. FIRST based rigid clusters identify the region from 48-51 to be highly flexible.

behavior from the simulation can be analyzed. Since the determination of constrained/flexible regions is determined by an ensemble of structures, the approach is not sensitive to changes in placement of hydrogen or other atoms which is often a problem with approaches such as FIRST [36].

It is also relevant to point out the timescales of performing DTA and MD simulations. While our DTA was performed on a dual processor Intel machine running at 2.4 GHz, the MD simulations were performed on two different supercomputers. With code optimization, one could run about 1 ns of simulation on a dual core processor, it would take approximately 5 days to obtain 5 ns simulation for barnase. DTA takes less than 1 s to process a window from the simulation, and thus, we believe that there would be no significant overhead of using DTA to analyze the data.

#### 4.3 Identifying Events of Interest in MD Simulations

In order to identify time-points where there was significant deviation from the dynamical behavior observed so far, we plotted the reconstruction error(Eqn. 3) against the tensor windows. For barnase, as shown in Fig. 6(a), most tensors (236 or 94%) fall within the mean and second standard deviation interval. Beyond the second standard deviation interval, we find about 14 tensor windows. Of these small number of tensors, we found that about 70% of structures within the tensor windows show an average RMS deviation of about 1 Å compared to the immediate predecessor window; the rest of the structures show RMS deviation of  $\geq 0.6$  Å. We selected some of these structures for detailed analysis. In the tensor window 83 (highlighted by a red circle), the greatest distance deviation observed was the displacement of the flexible loops in residues 21-48 with respect to the loops L1 and L2 towards each other as seen in Fig. 6(b) (top panel). In another window (215), loop L1 had moved with respect to  $\alpha$ 1 helix compared to the



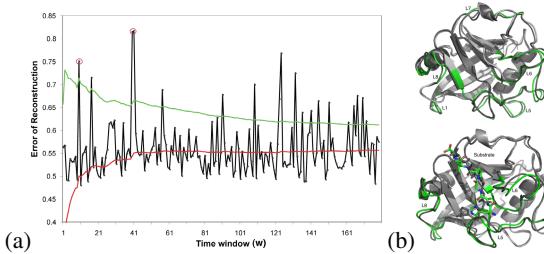
**Fig. 6. Reconstruction Error for barnase.** (a) shows the reconstruction error plotted as a function of tensor window. The red line indicates the mean reconstruction error as per Eqn. 3 and the second standard deviation above the mean is shown in green. Red circles are used to highlight those tensor snapshots shown in the adjacent panel. (b) Structural changes associated with  $w = 40$  (top panel) showing movements in L1 and L2 along with the functional domain 21-48, overall RMSD 1.03 Å;  $w = 215$  (bottom) showing movements associated with  $\alpha 1$  and L1 with an overall RMSD of 0.612 Å. In both cases, the regions shown in a darker shade of gray represent predecessor windows whereas lighter shade shows the current window.

predecessor window (214), illustrated in Fig. 6(b) (bottom). The collective displacement of these loops with respect to residues 21-48 is of importance since they represent inter-domain movements [37].

A similar analysis was done for cyclophilin A (Fig. 7(a)). We identified a total of 17 out of 180 tensors showing deviations beyond the second standard deviation interval. The average structural deviation in each of these windows was about 0.8 Å, compared to its immediate predecessor window. For example, a comparison of the structures from  $w = 10$  with that of  $w = 9$  (Fig. 7(b); top panel), indicated motions of the flexible loop L1 with respect to that of  $\alpha 3$ , L4 and L8, followed by motions across the L5-L6 regions of the protein. When we compared the two windows  $w = 40$  with  $w = 39$  (Fig. 7(b); bottom panel), we found that the substrate molecule showed large deviations with respect to both its interacting partners, namely  $\beta 3$  and L6-L8. The other windows reflected similar motions in these regions of the enzyme. The motions associated with these regions of the enzyme are known to be correlated with the catalytic process [52] and therefore seem to indicate functional importance.

RMS deviations are a good measure of how a protein's structure evolves during the course of a simulation. However, it is important to note that RMS values do not provide any information on the collective behavior that influences spatially separate regions of a protein. To show that our approach is indeed sensitive to such changes, we provide two specific examples from each of our simulations. In barnase, the time point between  $w = 30$  and  $w = 40$  (Fig. 6(b) top panel), show an RMS deviation of 1.033 Å, whereas the window  $w = 214$  and  $w = 215$  (Fig. 6(b) bottom panel) shows an RMS deviation of just 0.612 Å. Although this RMSD is quite small, there is a large movement with respect to the loop structures of the protein.

In cyclophilin A, the time point between  $w = 9$  and  $w = 10$  shows an RMS deviation of 0.617 Å (Fig. 7(a) top panel) and the window  $w = 30$  and window  $w = 40$  shows an RMS deviation of 0.650 Å (Fig. 7(b) bottom panel). It is important to note that even though the overall RMSD for cyclophilin A in the two windows is relatively small,



**Fig. 7. Reconstruction Error for Cyclophilin A.** (a) shows the reconstruction error plotted as a function of tensor window. Red line: mean reconstruction error, green line: second standard deviation interval. Red circles are used to highlight those tensor windows shown in the adjacent panel. (b) Structural changes associated with cyclophilin A for two windows, namely  $w = 10$  (top; overall RMSD 0.617 Å) and  $w = 40$  (bottom; overall RMSD 0.650 Å). The structure from the predecessor window is shown in dark gray and the current window is shown in light gray; regions involved in collective movements highlighted in green. Note the large movement in the substrate molecule, shown as sticks in the bottom panel. This cannot be picked up using traditional metrics such as RMSD since it is only an average measure of structural deviations. However, tracking distance variations, we note a significant difference in the placement of the substrate.

the substrate peptide highlighted in Fig. 7(b) (bottom panel), shows a distinct conformational rearrangement which could not have been detected by using RMS deviations alone. Since DTA incorporates the deviations in the distances to analyze simulations, we can keep track of changes that may relate two distally located residues.

## 5 Conclusions

We have shown that it is possible to obtain biologically relevant information about flexible and rigid substructures within a protein as the simulations progress. This was illustrated on two entirely different sets of simulations and in both cases, we were able to identify dynamically coupled regions as well as constrained residues within the protein. We were also able to correlate the residues identified to be constrained to have some functional role in protein dynamics and function as verified via experimental data and prior computational work. We were also able to demonstrate that using the reconstruction error as a metric, the structures identified from different windows of the simulation showed significant deviations in two or more regions of the protein, indicating a change in collective behavior.

There are several problem domains where one may want to apply our approach. First, we note that although we used Cartesian distances as a means to capture spatial dependencies, the method is very flexible and can use any feature that can be represented as a tensor. For example, it may be beneficial to track forces or velocities over different atoms to detect patterns of energy flow in protein structures. We are also extending the method to track multiple features, such as distances between hydrogen bonds/ hydrophobic interactions in the same simulation. By tracking cross correlations that may exist between different streams of data, we may be able to provide detailed information to end-users about correlated changes in non-covalent interactions and how they

account for large-scale motions within a protein. This is of fundamental interest to chemists and biologists alike in domains such as protein and drug design where networks of covalent and non-covalent interactions are known to widely affect protein function [344148].

The core tensor  $\mathcal{X} = \mathcal{C} \prod_{i=1}^d \times \mathbf{U}_i$  can be used as an efficient means to approximate the dynamics thus far observed in a simulation to build linear kernels such that one may learn and reason about protein dynamics on a long time scale. This allows us to develop new techniques to perform unsupervised learning from such large-scale data. On the other hand, tensor analysis also provides an ideal handle for data reduction if one were to use a suitable rank approximation on the number of eigenvectors to approximate the space spanned by the current data. This is valuable in case of storing and processing simulation data, which can easily reach terabyte scales, even for small simulations.

The approach outlined here is an unsupervised learning for an MD simulation. The core tensor can be used to summarize the behavior of a protein during the course of a simulation. However, an extension to this method would allow one to check for consistency between two or more simulations based on the same protein. Also, recent work indicates that tensor based methods are being increasingly used in supervised learning [43]. It would be interesting to see if one could use such approaches to learn from MD simulations. This approach will be useful to predict the behavior of a protein over the course of a simulation.

Unlike GNM [9]/ ANM [6], our approach is not limited to studying only biophysical motions involved in binding or molecular recognition. The flexibility of our approach allows us to investigate motions involved in biochemical processes such as catalysis (as was done in this paper) and even protein folding simulations. This will allow one to generalize approaches such as GNM or ANM to time-series data, which may be of potential benefit to analyze complex biological processes.

From the perspective of analyzing simulations online, a direct extension to our method will be to fork simulations from the time-points where significant deviation in dynamics is observed in order to sample a larger conformational space. This in turn is particularly useful to drive simulations involving chemical reactions or folding pathways which are known to be hard to simulate. Further, the identification of *events of interest* within a simulation is particularly useful in folding applications where tracking/ monitoring structural changes is of prime importance. We propose to investigate folding trajectories in the near future to make useful predictions of how one may sample the conformational space and identify structural intermediates that may be important for the protein to fold correctly.

## Acknowledgements

We thank Christos Faloutsos and Jimeng Sun from the Computer Science Department at Carnegie Mellon for introducing us to DTA and providing us with the implementation of DTA. This work is supported in part by US Department of Energy (DOE) Career Award and a grant from Microsoft Research to CJL. Pratul K. Agarwal would like to acknowledge the financial support from the Laboratory Directed Research and Development (LDRD) Program of Oak Ridge National Laboratory (ORNL), managed by

UT-Battelle, LLC for the U. S. DOE. We thank Tatyana Mamonova from the Kurnikova group for providing us access to MD simulations of barnase. We also thank Hetunandan Kamisetty from the Langmead lab for constructive discussions. We thank the anonymous reviewers for their valuable comments.

## References

1. Acar, E., Aykut-Bingol, C., Bingol, H., Bro, R., Yener, B.: Multiway analysis of epilepsy tensors. *Bioinformatics* 23(13), i10–i18 (2007)
2. Agarwal, P.K.: Cis/trans isomerization in hiv-1 capsid protein catalyzed by cyclophilin a: Insights from computational and theoretical studies. *Proteins: Struct., Funct., Bioinformatics* 56, 449–463 (2004)
3. Agarwal, P.K.: Enzymes: An integrated view of structure, dynamics and function. *Microbial Cell Factories* 5 (2006)
4. Agarwal, P.K., Billeter, S.R., Rajagopalan, P.T.R., Hammes-Schiffer, S., Benkovic, S.J.: Network of coupled promoting motions in enzyme catalysis. *Proc. Natl. Acad. Sci. USA* 99, 2794–2799 (2002)
5. Agarwal, P.K., Geist, A., Gorin, A.: Protein dynamics and enzymatic catalysis: Investigating the peptidyl-prolyl cis-trans isomerization activity of cyclophilin a. *Biochemistry* 43(33), 10605–10618 (2004)
6. Atilgan, A.R., Durell, S.R., Jernigan, R.L., Demirel, M.C., Keskin, O., Bahar, I.: Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophys. J.* 80, 505–515 (2001)
7. Bader, B.W., Kolda, T.G.: Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software* 32(4), 635–653 (2006)
8. Bader, B.W., Kolda, T.G.: Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing* 30(1), 205–231 (2007)
9. Bahar, I., Atilgan, A.R., Demirel, M.C., Erman, B.: Vibrational dynamics of folded proteins. significance of slow and fast modes in relation to function and stability. *Phys. Rev. Lett.* 80, 2733–2736 (1998)
10. Bahar, I., Cui, Q.: Normal Mode Analysis: Theory and Applications to Biological and Chemical Systems. Mathematical and Computational Biology Series. Chapman and Hall/ CRC, New York (2003)
11. Bahar, I., Rader, A.J.: Coarse grained normal mode analysis in structural biology. *Cur. Op. Struct. Biol.* 15, 1–7 (2005)
12. Beazley, D.M., Lomdahl, P.S.: Lightweight computational steering of very large scale molecular dynamics simulations. In: Supercomputing 1996 proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM), Washington, DC, USA, p. 50. IEEE Computer Society Press, Los Alamitos (1996)
13. Berendsen, H.J.C., Hayward, S.: Collective protein dynamics in relation to function. *Current Opinion in Structural Biology* 10(2), 165–169 (2000)
14. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. *Nucleic Acids Research* 28, 235–242 (2002)
15. Bowers, K.J., Chow, E., Xu, H., Dror, R.O., Eastwood, M.P., Gregersen, B.A., Klepeis, J.L., Kolossvary, I., Moraes, M.A., Sacerdoti, F.D., Salmon, J.K., Shan, Y., Shaw, D.E.: Scalable algorithms for molecular dynamics simulations on commodity clusters. In: SC Conference, p. 43 (2006)
16. Chodera, J.D., Singhal, N., Pander, V.S., Dill, K.A., Swope, W.C.: Automatic discovery of metastable states for the construction of markov models of macromolecular conformational dynamics. *J. Chem. Phys.* 126, 155101 (2007)

17. DeLano, W.L.: The pymol molecular graphics system (2003)
18. Eisenmesser, E.Z., Bosco, D.A., Akke, M., Kern, D.: Enzyme dynamics during catalysis. *Science* 295(5559), 1520–1523 (2002)
19. Fersht, A.R., Daggett, V.: Protein folding and unfolding at atomic resolution. *Cell* 108(4), 573–582 (2002)
20. Fersht, A.R., Matouschek, A., Sancho, J., Serrano, L., Vuilleumier, S.: Pathway of protein folding. *Faraday Discuss* 93, 183–193 (1992)
21. Fersht, A.R.: Protein folding and stability: the pathway of folding of barnase. *FEBS Letters* 325(1-2), 5–16 (1993)
22. Gerstein, M., Krebs, W.: A database of macromolecular motions. *Nucl. Acids Res.* 26(18), 4280–4290 (1998)
23. Gu, W., Eisenhauer, G., Kraemer, E., Schwan, K., Stasko, J., Vetter, J., Mallavarupu, N.: Falcon: on-line monitoring and steering of large-scale parallel programs. In: *Symposium on the Frontiers of Massively Parallel Processing*, p. 422 (1995)
24. Gussio, R., Pattabiraman, N., Kellogg, G.E., Zaharevitz, D.W.: Use of 3d qsar methodology for data mining the national cancer institute repository of small molecules: Application to hiv-1 reverse transcriptase inhibition. *Methods* 14, 255–263 (1998)
25. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. *App. Stat.* 28(1), 100–108 (1979)
26. Hayward, S., Go, N.: Collective variable description of native protein dynamics. *Annual Review of Physical Chemistry* 46(1), 223–250 (1995)
27. Hespenheide, B.M., Rader, A.J., Thorpe, M.F., Kuhn, L.A.: Identifying protein folding cores: observing the evolution of rigid and flexible regions during unfolding. *J. Mol. Graph. and Model.* 21, 195–207 (2002)
28. Jacobs, D.J., Rader, A.J., Kuhn, L.A., Thorpe, M.F.: Protein flexibility predictions using graph theory. *Proteins: Struct., Funct., Genet.* 44(2), 150–165 (2001)
29. Jolliffe, I.T.: *Principal Component Analysis*. Springer, Heidelberg (2002)
30. Karplus, M., McCammon, J.A.: Molecular dynamics simulations of biomolecules. *Nat. Struct. Biol.* 9, 646–652 (2002)
31. Karplus, M., Kushick, J.N.: Method for estimating the configurational entropy of macromolecules. *Macromolecules* 14(2), 325–332 (1981)
32. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. Technical report, Sandia National Laboratories (2007)
33. Lange, O.F., Grubmuller, H.: Full correlation analysis of conformational protein dynamics. *Proteins: Struct., Funct. and Bioinformatics* 70, 1294–1312 (2008)
34. Lenaerts, T., Ferkinghoff-Borg, J., Stricher, F., Serrano, L., Schymkowitz, J.W.H., Rousseau, F.: Quantifying information transfer by protein domains: Analysis of the fyn sh2 domain structure. *BMC Struct. Biol.* 8, 43 (2008)
35. Malmodin, D., Billeter, M.: Multiway decomposition of nmr spectra with coupled evolution periods. *J. Am. Chem. Soc.* 127(39), 13486–13487 (2005)
36. Mamanova, T., Hespenheide, B., Straub, R., Thorpe, M.F., Kurnikova, M.: Protein flexibility using constraints from molecular dynamics simulations. *Phys. Biol.* 2(4), S137–S147 (2005)
37. Nolde, S.B., Arseniev, A.S., Yu, V., Billeter, M.: Essential domain motions in barnase revealed by md simulations. *Proteins: Struct., Funct. and Bioinformatics* 46(3), 250–258 (2003)
38. Shao, J., Tanner, S.W., Thompson, N., Cheatham, T.E.: Clustering molecular dynamics trajectories: 1. characterizing the performance of different clustering algorithms. *Journal of Chemical Theory and Computation* 3(6), 2312–2334 (2007)
39. Smilde, A., Bro, R., Geladi, P.: *Multi-way Analysis: Applications in the Chemical Sciences*. J. Wiley and Sons, Ltd., Chichester (2004)

40. Staykova, D., Fredriksson, J., Bermel, W., Billeter, M.A.: Assignment of protein nmr spectra based on projections, multi-way decomposition and a fast correlation approach. *Journal of Biomolecular NMR* (2008)
41. Suel, G.M., Lockless, S.W., Wall, M.A., Ranganathan, R.: Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nat. Struct. Biol.* 10, 59–69 (2003)
42. Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: Dynamic tensor analysis (2006)
43. Tao, D., Li, X., Wu, X., Hu, W., Stephen, J.M.: Supervised tensor learning. *Knowledge and Information Systems* 13, 42 (2007)
44. Whiteley, W.: Rigidity of Molecular structures: generic and geometric analysis. In: *Rigidity Theory and Applications*. Kluwer Academic/ Plenum, New York (1999)
45. Yanagawa, H., Yoshida, K., Torigoe, C., Park, J.S., Sato, K., Shirai, T., Go, M.: Protein anatomy: functional roles of barnase module. *J. Biol. Chem.* 268(8), 5861–5865 (1993)
46. Yener, B., Acar, E., Aguis, P., Bennett, K., Vandenberg, S., Plopper, G.: Multiway modeling and analysis in stem cell systems biology. *BMC Systems Biology* 2(1), 63 (2008)
47. Zavodszky, M.I., Lei, M., Thorpe, M.F., Day, A.R., Kuhn, L.A.: Modeling correlated main-chain motions in proteins for flexible molecular recognition. *Proteins: Struct. Funct. and Bioinformatics* 57(2), 243–261 (2004)
48. Zhuravleva, A., Korzhnev, D.M., Nolde, S.B., Kay, L.E., Arseniev, A.S., Billeter, M., Orekhov, V.Y.: Propagation of dynamic changes in barnase upon binding of barstar: An nmr and computational study. *Journal of Molecular Biology* 367(4), 1079–1092 (2007)

# Parameter Synthesis in Nonlinear Dynamical Systems: Application to Systems Biology

Alexandre Donzé<sup>1</sup>, Gilles Clermont<sup>2</sup>,  
Axel Legay<sup>1</sup>, and Christopher J. Langmead<sup>1,3,\*</sup>

<sup>1</sup> Computer Science Department, Carnegie Mellon University, Pittsburgh, PA

<sup>2</sup> Department of Critical Care Medicine, University of Pittsburgh, Pittsburgh, PA

<sup>3</sup> Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA  
`cjl@cs.cmu.edu`

**Abstract.** The dynamics of biological processes are often modeled as systems of nonlinear ordinary differential equations (ODE). An important feature of nonlinear ODEs is that seemingly minor changes in initial conditions or parameters can lead to radically different behaviors. This is problematic because in general it is never possible to know/measure the precise state of any biological system due to measurement errors. The parameter synthesis problem is to identify sets of parameters (including initial conditions) for which a given system of nonlinear ODEs does not reach a given set of undesirable states. We present an efficient algorithm for solving this problem that combines sensitivity analysis with an efficient search over initial conditions. It scales to high-dimensional models and is exact if the given model is affine. We demonstrate our method on a model of the acute inflammatory response to bacterial infection, and identify initial conditions consistent with 3 biologically relevant outcomes.

**Keywords:** Verification, Nonlinear Dynamical Systems, Uncertainty, Systems Biology, Acute Illness.

## 1 Introduction

The fields of Systems Biology, Synthetic Biology, and Medicine produce and use a variety of formalisms for modeling the dynamics of biological systems. Regardless of its mathematical form, a model is an invaluable tool for thoroughly examining how the behavior of a system changes when the initial conditions are altered. Such studies can be used to generate verifiable predictions, and/or to address the uncertainty associated with experimental measurements obtained from real systems.

In this paper, we consider the *parameter synthesis problem* which is to identify sets of parameters for which the system does (or does not) reach a given set of states. Here, the term “parameter” refers to both the initial conditions of the model (e.g., bacterial load at time  $t = 0$ ) and dynamical parameters (e.g., the bacterium’s doubling rate). For example, in the context of medicine, we might be interested in partitioning the parameter space into two regions — those that, without medical intervention, deterministically

---

\* Corresponding author.

lead to the patient’s recovery, and those that lead to the patient’s death. The parameter synthesis problem is relatively easy to solve when the system has linear dynamics, and there are a variety of methods for doing so (e.g., [6,7,8]). Our algorithm, in contrast, solves the parameter synthesis problem for *nonlinear dynamical systems*. That is, for systems of nonlinear ordinary differential equations (ODEs). Moreover, our approach can also be extended to nonlinear hybrid systems (i.e., those containing mixtures of discrete and continuous variables, see [11] for details). Nonlinear ODE and hybrid models are very common in the Systems Biology, Synthetic Biology, and in Medical literature but there are very few techniques for solving the parameter synthesis problem in such systems. This paper’s primary contribution is a practical algorithm that can handle systems of this complexity.

Our algorithm combines sensitivity analysis with an efficient search over parameters. The method is exact if the model has affine dynamics. For nonlinear dynamical systems, we can guarantee an arbitrarily high degree of accuracy with respect to identifying the boundary delineating reachable and non-reachable sets. Moreover, our method runs in minutes, even on high-dimensional models. We demonstrate the method by examining two models of the inflammatory response to bacterial infection [20,26]. In each case, we identify sets of initial conditions that lead to each of 3 biologically relevant outcomes.

The contributions of this paper are as follows:

- An algorithm for computing parameter synthesis in nonlinear dynamical systems. This work builds on and extends formal verification techniques that were first introduced in the context of continuous and hybrid nonlinear dynamical systems [13].
- The results of two studies on two different models of the inflammatory response to bacterial infection. The first model is a 4-equation model, the second is a 17-equation model.

This paper is organized as follows: We outline previous work in reachability for biological systems in Sec. 2. Next, we present our algorithm in Sec. 3. We demonstrate our method on two models of acute inflammation in Sec. 4. We finish by discussing our results and ideas for future work in Sec. 5.

## 2 Background

Our work falls under the category of *formal verification*, a large area of research which focus on techniques for computing provable guarantees that a system satisfies a given property. Formal verification methods can be characterized by the kind of system they consider (e.g., discrete-time vs continuous-time, finite-state vs continuous-state, linear vs non-linear dynamics, etc), and by the kind of properties they can verify (e.g, reachability – the system can be in a given state, liveness – the system will be in a given set of state infinitely often, etc). The algorithm presented in this paper is intended for verifying reachability properties under parameter uncertainty in nonlinear hybrid systems. The most closely related work in this area uses *symbolic* methods for restricted class of models (e.g., timed automata [4], linear hybrid systems [1,19,17]). Symbolic methods for hybrid systems have the advantage that they are exhaustive, but in general only scale

to systems of small size ( $< 10$  continuous state variables). Another class of techniques invokes abstractions of the model [2]. Such methods have been applied to biological systems whose dynamics can be described by multi-affine functions. Here, examples include applications to genetic regulatory networks (e.g., [6,7,8]). Batt and co-workers proposed an approach to verify reachability and liveness properties written in the linear temporal logic (LTL) [24] (LTL can be used to check assumptions about the future such as equilibrium points) of genetic regulatory networks under parameter uncertainty. In that work, the authors show that one can reduce the verification of qualitative properties of genetic regulatory networks to the application of Model Checking techniques [10] on a conservative discrete abstraction. Our method is more general in the sense that we can handle arbitrary nonlinear systems but a limitation is that we cannot handle liveness properties. However, we believe that our algorithm can be extended to handle liveness properties by combining it with a recent technique proposed by Fainekos [15]. We note that there is also work in the area of analyzing piecewise (stochastic) hybrid systems (e.g., [14,16,18,9]). Our method does not handle stochastic models at the present time.

Several techniques relying on numerical computations of the reachable set apply to systems with general nonlinear dynamics ([5,28,22]). In [5], the authors presents an hybridization technique, which consists in approximating the system with a piecewise-affine approximation to take advantage of the wider family of methods existing for this class of systems. In [21], the authors reduce the reachability problem to a partial differential equation which they solve numerically. As far as we know, none of these techniques have been applied successfully to nonlinear systems of more than a few variables. By contrast, our method builds on techniques proposed in [12,13] which can be applied to significantly larger models.

A more “traditional” tool used for the analysis of nonlinear ODEs is bifurcation analysis, which was applied to the biological models used in our experiments ([26,20]). Our approach deviates from bifurcation analysis in several ways. First, it is simpler to apply since it only relies on the capacity to compute numerical simulations for the system, avoiding the need of computing equilibrium points or limit cycles. Second, it provides the capacity of analyzing transient behaviors. Finally, when it encounters an ambiguous behavior (e.g., bi-stability) for a given parameter set, it reports that the parameter has uncertain dynamics and can refine the result to make such uncertain sets as small as desired.

### 3 Algorithm

In this section, we give a mathematical description of the main algorithm used in this work.

#### 3.1 Preliminaries

The set  $\mathbb{R}^n$  and the set of  $n \times n$  matrices are equipped with the infinite norm, noted  $\|\cdot\|$ . We define the diameter of a compact set  $\mathcal{R}$  to be  $\|\mathcal{R}\| = \sup_{(x,x') \in \mathcal{R}^2} \|x - x'\|$ . The distance from  $x$  to  $\mathcal{R}$  is  $d(x, \mathcal{R}) = \inf_{y \in \mathcal{R}} \|x - y\|$ . The Haussdorff distance between two sets  $\mathcal{R}_1$  and  $\mathcal{R}_2$  is:

$$d_H(\mathcal{R}_1, \mathcal{R}_2) = \max\left(\sup_{x_1 \in \mathcal{R}_1} d(x_1, \mathcal{R}_2), \sup_{x_2 \in \mathcal{R}_2} d(x_2, \mathcal{R}_1)\right).$$

Given a matrix  $S$  and a set  $\mathcal{P}$ ,  $S\mathcal{P}$  represents the set  $\{Sp, p \in \mathcal{P}\}$ . Given two sets  $\mathcal{R}_1$  and  $\mathcal{R}_2$ ,  $\mathcal{R}_1 \oplus \mathcal{R}_2$  is the Minkowski sum of  $\mathcal{R}_1$  and  $\mathcal{R}_2$ , i.e.,  $\mathcal{R}_1 \oplus \mathcal{R}_2 = \{x_1 + x_2, x_1 \in \mathcal{R}_1, x_2 \in \mathcal{R}_2\}$ .

### 3.2 Simulation and Sensitivity Analysis

We consider a dynamical system  $\mathcal{S}ys = (f, \mathcal{P})$  of the form:

$$\dot{x} = f(t, x, p), \quad p \in \mathcal{P}, \quad (1)$$

where  $x \in \mathbb{R}^n$ ,  $p$  is a *parameter vector* and  $\mathcal{P}$  is a compact subset of  $\mathbb{R}^{n_p}$ . We assume that  $f$  is continuously differentiable. Let  $\mathcal{T} \subset \mathbb{R}^+$  be a time set. For a given  $p$ , a *trajectory*  $\xi_p$  is a function of  $\mathcal{T}$  which satisfies the ODE (Eq. 1), i.e., for all  $t$  in  $\mathcal{T}$ ,  $\dot{\xi}_p(t) = f(t, \xi_p(t), p)$ . For convenience, we include the initial state in the parameter vector by assuming that if  $p = (p_1, p_2, \dots, p_{n_p})$  then  $\xi_p(0) = (p_1(0), p_2(0), \dots, p_n(0))$ . Under these conditions, we know by the Cauchy-Lipshitz theorem that the trajectory  $\xi_p$  is uniquely defined.

The purpose of sensitivity analysis techniques is to predict the influence on a trajectory of a perturbation of its parameter vector. A first order approximation of this influence can be obtained by a Taylor expansion of  $\xi_p(t)$  around  $p$ . Let  $\delta p \in \mathbb{R}^{n_p}$ . We have:

$$\xi_{p+\delta p}(t) = \xi_p(t) + \frac{\partial \xi_p}{\partial p}(t) \delta p + \mathcal{O}(\|\delta p\|^2). \quad (2)$$

The second term in the right hand side of Eq. (2) is the derivative of the trajectory with respect to  $p$ . Since  $p$  is a vector, this derivative is a matrix, which is called the *sensitivity matrix*. We denote it as:  $S_p(t) = \frac{\partial \xi_p}{\partial p}(t)$

The sensitivity matrix can be computed as the solution of a system of ODEs. Let  $\mathbf{s}_i = \frac{\partial \xi_p}{\partial p_i}(t)$  be the  $i^{\text{th}}$  column of  $S_p$ . If we apply the chain rule to its time derivative, we get:

$$\begin{cases} \dot{\mathbf{s}}_i(t) = \frac{\partial f}{\partial x}(t, x(t), p) \mathbf{s}_i(t) + \frac{\partial f}{\partial p_i}(t, x(t), p), \\ \mathbf{s}_i(0) = \frac{\partial x(0)}{\partial p_i}. \end{cases} \quad (3)$$

Here  $\frac{\partial f}{\partial x}(t, x(t), p)$  is the Jacobian matrix of  $f$  at time  $t$ . The equation above is thus an affine, time-varying ODE. In the core of our implementation, we compute  $\xi_p$  and the sensitivity matrix  $S_p$  using the CVODES numerical solver [27], which is designed to solve efficiently and accurately ODEs (like Eq. 1) and sensitivity equations (like Eq. 3).

### 3.3 Reachable Set Estimation Using Sensitivity

The reachability problem is the problem of computing the set of all the states visited by the trajectories starting from all the possible initial parameters in  $\mathcal{P}$  at a given time  $t$ .

**Definition 1 (Reachable Set).** *The reachable set induced by the set of parameters  $\mathcal{P}$  at time  $t$  is:*

$$\mathcal{R}_t(\mathcal{P}) = \bigcup_{p \in \mathcal{P}} \xi_p(t).$$

The set  $\mathcal{R}_t(\mathcal{P})$  can be approximated by using sensitivity analysis. Assume that for a given parameter  $p$  in  $\mathcal{P}$  we computed a trajectory  $\xi_p$  and the sensitivity matrix  $S_p$  associated with it. Given another parameter vector  $p'$  in  $\mathcal{P}$ , we can use this matrix to get an estimate  $\hat{\xi}_{p'}^p(t)$  of  $\xi_{p'}(t)$ . This is done by dropping higher order terms in the Taylor expansion given in Equation 2. We have:

$$\hat{\xi}_{p'}^p(t) = \xi_p(t) + S_p(t)(p' - p). \quad (4)$$

If we extend this estimation to all parameters  $p'$  in  $\mathcal{P}$ , we get the following estimate of the reachable set  $\mathcal{R}_t(\mathcal{P})$ :

$$\hat{\mathcal{R}}_t^p(\mathcal{P}) = \bigcup_{p' \in \mathcal{P}} \hat{\xi}_{p'}^p(t) = \{\xi_p(t) - S_p(t)p\} \oplus S_p(t)\mathcal{P}. \quad (5)$$

Thus  $\hat{\mathcal{R}}_t^p$  is an affine mapping of the initial set  $\mathcal{P}$  into  $\mathbb{R}^n$  (see Figure 1).

It can be shown that if the dynamics are affine, i.e., if  $f(t, x, p) = A(t, p)x + b(t, p)$ , then the estimation is exact. However, in the general case,  $\hat{\mathcal{R}}_t^p(\mathcal{P})$  is different from  $\mathcal{R}_t(\mathcal{P})$ . Since the estimation is based on a first order approximation around parameter  $p$ , it is local in the parameter space and its quality depends on how “big”  $\mathcal{P}$  is. In order to improve the estimation, we can partition  $\mathcal{P}$  into smaller subsets  $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_l$  and compute trajectories using new initial parameters  $p_1, p_2, \dots, p_l$  to get more precise local estimates. As a practical matter, we need to be able to estimate the benefit of such a refinement. To do so, we compare  $\hat{\mathcal{R}}_t^p(\mathcal{P}_j)$  — the estimate we get when using the “global” center,  $p$ ; to  $\hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)$  — the estimate we get when using the “local” center,  $p_j$ , and  $p_i \in \mathcal{P}_j$ . We do this for each  $\mathcal{P}_j$ . Figure 1 illustrates the essential features of the algorithm.

**Proposition 1.** *We have*

$$d_H(\hat{\mathcal{R}}_t^p(\mathcal{P}_j), \hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)) \leq Err(\mathcal{P}, \mathcal{P}_j), \quad (6)$$

where

$$Err(\mathcal{P}, \mathcal{P}_j) = \|\xi_{p_j}(t) - \hat{\xi}_{p_j}^p(t)\| + \|S_{p_j}(t) - S_p(t)\| \|\mathcal{P}_j\|. \quad (7)$$

In other words, the difference between the global and the local estimate can be decomposed into the error introduced in the estimation  $\hat{\xi}_{p_j}^p(t)$  of the state reached at time  $t$  using  $p_j$  (first term on RHS of Eq. 7), and another term involving the difference between the local and the global sensitivity matrices and the distance from local center (second term on RHS of Eq. 7).

*Proof.* let  $y$  be in  $\hat{\mathcal{R}}_t^p(\mathcal{P}_j)$ . There exists  $p_y$  in  $\mathcal{P}_j$  such that  $y = \hat{\xi}_{p_y}^p(t)$ . We need to compare

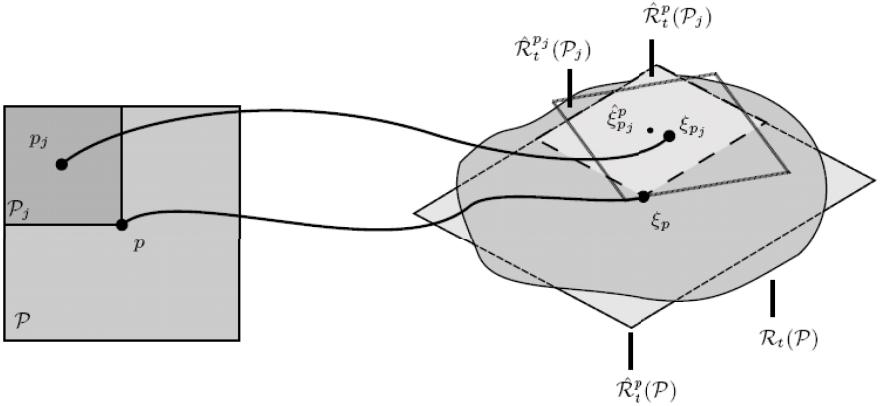
$$\hat{\xi}_{p_y}^p(t) = \xi_p(t) + S_p(t)(p_y - p) \quad (8)$$

with

$$\hat{\xi}_{p_j}^{p_j}(t) = \xi_{p_j}(t) + S_{p_j}(t)(p_y - p_j). \quad (9)$$

By introducing

$$\hat{\xi}_{p_j}^p(t) = \xi_p(t) + S_p(t)(p_j - p) \quad (10)$$



**Fig. 1.** Comparison between a “global” and a “local” estimate of the reachable set. The large square on the left hand side represent a region of parameter space,  $\mathcal{P}$ . The oval-shaped region on the right hand side corresponds to the true reachable set,  $\mathcal{R}_t(\mathcal{P})$ , induced by parameters  $\mathcal{P}$  at time  $t$ . The large parallelogram on the right hand side corresponds to the *estimated* reachable set,  $\hat{\mathcal{R}}_t^p(\mathcal{P})$ , using a sensitivity analysis based on trajectory labeled  $\xi_p$  which starts at point  $p \in \mathcal{P}$ . The point labeled  $\hat{\xi}_{p_j}^p$ , for example, is an estimate of where a trajectory starting at point  $p_j$  would reach at time  $t$ . If we partition  $\mathcal{P}$  and consider some particular partition,  $\mathcal{P}_j$ , we can then compare the estimated reachable sets  $\hat{\mathcal{R}}_t^p(\mathcal{P}_j)$  and  $\hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)$ , which correspond to the small light-gray and small dark gray parallelograms, respectively. We continue to refine until the distance between  $\hat{\mathcal{R}}_t^p(\mathcal{P}_j)$  and  $\hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)$  (Eq. 7) falls below some user-specified tolerance.

and after some algebraic manipulations of (8), (9), and (10), we get

$$\begin{aligned} \hat{\xi}_{p_y}^p(t) - \hat{\xi}_{p_y}^{p_j}(t) &= \xi_{p_j}(t) - \hat{\xi}_{p_j}^p(t) + (S_{p_j}(t) - S_p(t))(p_y - p_j) \\ &\leq \|\xi_{p_j}(t) - \hat{\xi}_{p_j}^p(t)\| + \|S_{p_j}(t) - S_p(t)\| \|\mathcal{P}_j\| = Err(\mathcal{P}, \mathcal{P}_j). \end{aligned} \quad (11)$$

Let  $x = \hat{\xi}_{p_y}^{p_j}(t)$  which is in  $\hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)$ , then it can be shown that  $\|y - x\| \leq Err(\mathcal{P}, \mathcal{P}_j)$  and so  $d(y, \hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)) \leq Err(\mathcal{P}, \mathcal{P}_j)$ . This is true for any  $y \in \mathcal{P}_j$ , thus

$$\sup_{y \in \hat{\mathcal{R}}_t^p(\mathcal{P}_j)} d(y, \hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)) \leq Err(\mathcal{P}, \mathcal{P}_j).$$

Similarly, we can show that

$$\sup_{x \in \hat{\mathcal{R}}_t^{p_j}(\mathcal{P}_j)} d(x, \hat{\mathcal{R}}_t^p(\mathcal{P}_j)) \leq Err(\mathcal{P}, \mathcal{P}_j)$$

which proves the result.  $\square$

The quantity  $Err(\mathcal{P}, \mathcal{P}_j)$  can be easily computed from trajectories  $\xi_p$  and  $\xi_{p_j}$ , their corresponding sensitivity matrices, and  $\|\mathcal{P}_j\|$ . It has the following properties:

- If the dynamics is affine, then  $Err(\mathcal{P}, \mathcal{P}_j) = 0$ . Indeed, in this case, we have  $\hat{\xi}_{p_j}^p = \xi_{p_j}$  so the first term vanishes and  $S_p = S_{p_j}$  so the second term vanishes as well;

- If limit  $\|\mathcal{P}\|$  is 0, then limit  $Err(\mathcal{P}, \mathcal{P}_j)$  is also 0. Indeed, as  $\|\mathcal{P}\|$  decreases, so does  $\|p - p_j\|$ , and thus  $\|\xi_{p_j}(t) - \hat{\xi}_{p_j}^p(t)\|$  and  $\|\mathcal{P}_j\|$ , since  $\mathcal{P}_j$  is a subset of  $\mathcal{P}$ . We can show that the convergence is quadratic.

The computation of reachable sets at a given time  $t$  can be extended to time intervals. Assume that  $\mathcal{T}$  is a time interval of the form  $\mathcal{T} = [t_0, t_f]$ . The set reachable from  $\mathcal{P}$  during  $\mathcal{T}$  is  $\mathcal{R}_{\mathcal{T}}(\mathcal{P}) = \cup_{t \in \mathcal{T}} \mathcal{R}_t(\mathcal{P})$ . It can be approximated by simple interpolation between  $\mathcal{R}_{t_0}(\mathcal{P})$  and  $\mathcal{R}_{t_f}(\mathcal{P})$ . Of course, it may be necessary to subdivide  $\mathcal{T}$  into smaller intervals to improve the precision of the interpolation. A reasonable choice for this subdivision is to use the time steps taken by the numerical solver to compute the solution of the ODE and the sensitivity matrices.

### 3.4 Parameter Synthesis Algorithm

In this section, we state a parameter synthesis problem and propose an algorithm that provides an approximate solution. Let  $\mathcal{F}$  be a set of "bad" states. Our goal is to partition the set  $\mathcal{P}$  into safe and bad parameters. That is, we want to partition the parameters into those that induce trajectories that intersect  $\mathcal{F}$  during some time interval  $\mathcal{T}$ , and those that do not.

**Definition 2.** A solution of the parameter synthesis problem ( $Sys = (f, \mathcal{P}), \mathcal{F}, \mathcal{T}$ ) where  $\mathcal{F}$  is a set states and  $\mathcal{T}$  a subset of  $\mathbb{R}_{\geq 0}$ , is a partition  $\mathcal{P}_{bad} \cup \mathcal{P}_{saf}$  of  $\mathcal{P}$  such that for all  $p \in \mathcal{P}_{bad}$  (resp.  $p \in \mathcal{P}_{saf}$ ),  $\xi_p(t) \cap \mathcal{F} \neq \emptyset$  (resp.  $\xi_p(t) \cap \mathcal{F} = \emptyset$ ) for all  $t \in \mathcal{T}$ . An approximate solution is a partition  $\mathcal{P} = \mathcal{P}_{saf} \cup \mathcal{P}_{unc} \cup \mathcal{P}_{bad}$  where  $\mathcal{P}_{saf}$  and  $\mathcal{P}_{bad}$  are defined as before and  $\mathcal{P}_{unc}$  (i.e., uncertain) may contain both safe and bad parameters.

Exact solutions cannot be obtained in general, but we can try to compute an approximate solution with the uncertain subset being as small as possible. The idea is to iteratively refine  $\mathcal{P}$  and to classify the subsets into the three categories. A subset  $\mathcal{P}_j$  qualifies as safe (resp. bad) if:

1.  $\hat{\mathcal{R}}_{\mathcal{T}}^{p_j}(\mathcal{P}_j)$  is a reliable estimation of  $\mathcal{R}_{\mathcal{T}}(\mathcal{P}_j)$  based on the  $Err$  function;
2.  $\hat{\mathcal{R}}_{\mathcal{T}}^{p_j}(\mathcal{P}_j)$  does not (resp. does) intersect with  $\mathcal{F}$ .

To guarantee that the process ends, we need to ensure that each refinement introduces only subsets that are strictly smaller than the refined set.

**Definition 3 (Refining Partition).** A refining partition of a set  $\mathcal{P}$  is a finite set of sets  $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_l\}$  such that

- $\mathcal{P} = \bigcup_{j=1}^l \mathcal{P}_j$ ;
- There exists  $\gamma < 1$  such that  $\max_{j \in \{1, \dots, l\}} \|\mathcal{P}_j\| \leq \gamma \|\mathcal{P}\|$ .

Let  $\rho$  be a function that maps a set to one of its refining partitions. Our algorithm stops whenever the uncertain partition is empty, or it contains only subsets with a diameter smaller than some user-specified value,  $\delta_p$ . The complete algorithm is given by Algorithm 1 below.

**Algorithm 1.** Parameter Synthesis Algorithm

---

```

procedure SAFE( $\mathcal{P}, \mathcal{F}, t, \delta p, Tol$ )
   $\mathcal{P}_{\text{saf}} = \mathcal{P}_{\text{bad}} = \emptyset, \mathcal{P}_{\text{unc}} = \{\mathcal{P}\}$ 
  repeat
    Pick and remove  $\mathcal{Q}$  from  $\mathcal{P}_{\text{unc}}$  and let  $q \in \mathcal{Q}$ 
    for each  $(q_j, \mathcal{Q}_j) \in \rho(\mathcal{Q})$  do
      if  $Err(\mathcal{Q}, \mathcal{Q}_j) \leq Tol$  then ▷ Reach set estimation is reliable
        if  $\hat{\mathcal{R}}_T^q(\mathcal{Q}_j) \cap \mathcal{F} = \emptyset$  then ▷ Reach set away from  $\mathcal{F}$ 
           $\mathcal{P}_{\text{saf}} = \mathcal{P}_{\text{saf}} \cup \mathcal{Q}_j$ 
        else if  $\hat{\mathcal{R}}_T^q(\mathcal{Q}_j) \subset \mathcal{F}$  then ▷ Reach set inside  $\mathcal{F}$ 
           $\mathcal{P}_{\text{bad}} = \mathcal{P}_{\text{bad}} \cup \mathcal{Q}_j$ 
        else
           $\mathcal{P}_{\text{unc}} = \mathcal{P}_{\text{unc}} \cup \{(q_j, \mathcal{Q}_j)\}$  ▷ Some intersection with the bad set
        end if
      else
         $\mathcal{P}_{\text{unc}} = \mathcal{P}_{\text{unc}} \cup \{(q_j, \mathcal{Q}_j)\}$  ▷ Reach set estimation not enough precise
      end if
    end for
  until  $\mathcal{P}_{\text{unc}} = \emptyset$  or  $\max_{P_j \in \mathcal{P}_{\text{unc}}} \|P_j\| \leq \delta p$ 
  return  $\mathcal{P}_{\text{saf}}, \mathcal{P}_{\text{unc}}, \mathcal{P}_{\text{bad}}$ 
end procedure

```

---

The algorithm has been implemented within the Matlab toolbox Breach [12], which combines Matlab routines to manipulate partitions with the CVODES numerical solver, which can efficiently compute ODEs solutions with sensitivity matrices.

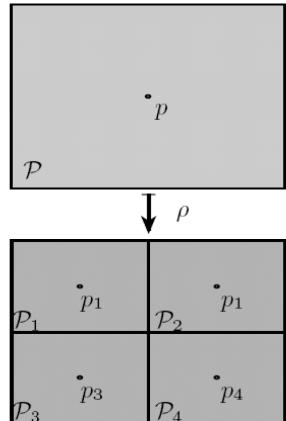
It uses rectangular partitions of the form

$$\mathcal{P}(p, \epsilon) = \{p' : p - \epsilon \leq p' \leq p + \epsilon\}$$

The refinement operator  $\rho$  is such that

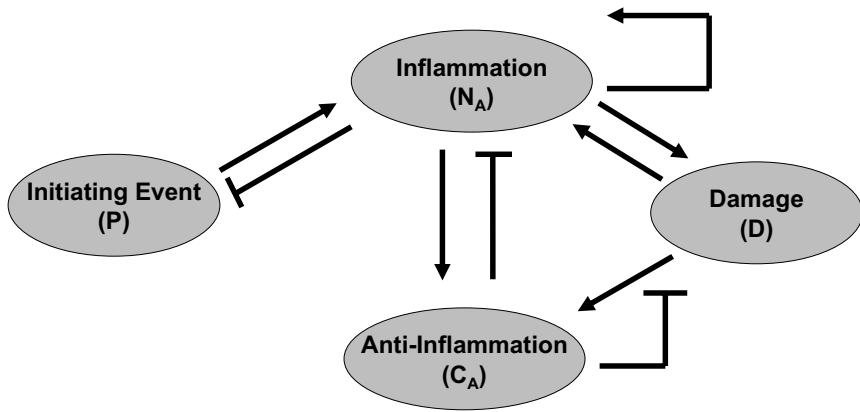
$$\rho(\mathcal{P}(p, \epsilon)) = \{\mathcal{P}(p^1, \epsilon^1), \mathcal{P}(p^2, \epsilon^2), \dots, \mathcal{P}(p^l, \epsilon^l)\},$$

with  $\epsilon^k = \epsilon/2$  and  $p^k = p + (\pm \frac{\epsilon_1}{2}, \pm \frac{\epsilon_2}{2}, \dots, \pm \frac{\epsilon_n}{2})$ . This operation is illustrated in the Figure on the right.



## 4 Application to Models of Acute Inflammation

We applied our method to two models of the acute inflammatory response to infection. The first is the 4-equation, 22-parameter model presented in [26], and the second is the 17-equation, 79-parameter model presented in [20]. The primary difference



**Fig. 2.** Cartoon representation of the 4-equation model of the acute immune response. Arrows represent up-regulation, bars represent down-regulation. Figure is adapted from Figure 1 in [26].

between these models is one of detail, and the first model can be thought of as a reduced dimensional version of the second.

The acute inflammatory response to infection has evolved to promote healing by ridding the organism of the pathogen. The actual response is a complex and carefully regulated combination of molecular and cellular cascades that exhibit both pro and anti-inflammatory behaviors. The pro-inflammatory elements are primarily responsible for eliminating the pathogen, but bacterial killing can cause collateral tissue damage. Tissue damage, in turn, triggers an escalation in the pro-inflammatory response creating a positive feedback cycle (Figure 2). The anti-inflammatory elements counteract this cycle, thereby minimizing tissue damage and promoting healing. However, in cases of extreme infection, the delicate balance between pro and anti-inflammatory elements is destroyed, resulting in a potentially lethal amount of tissue damage.

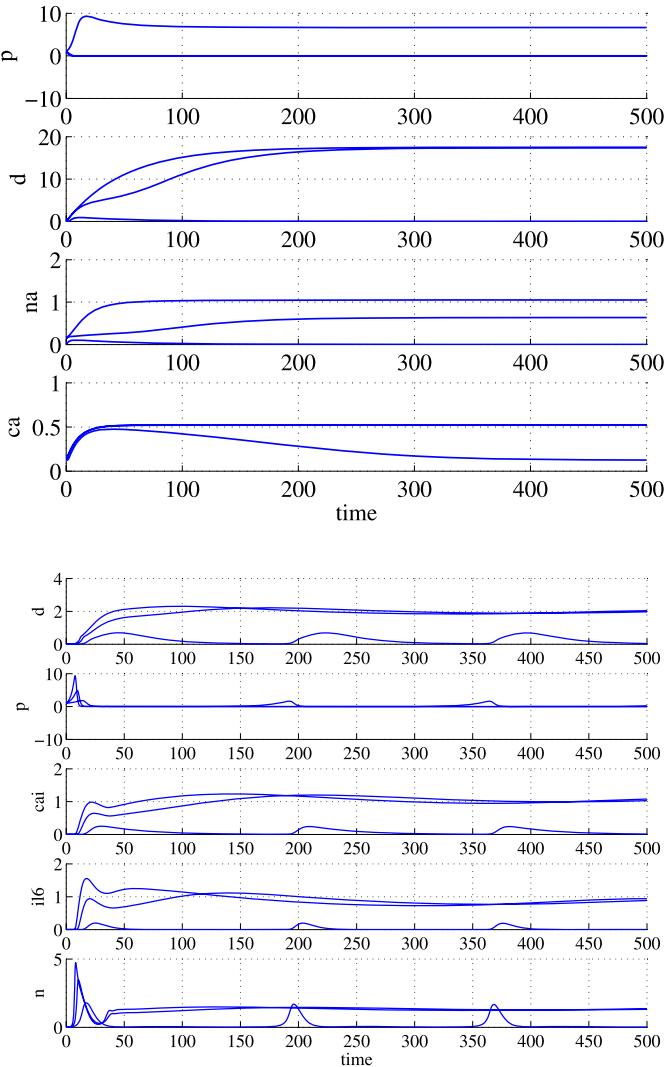
The 4-equation model is as follows:

$$\begin{aligned}
 \frac{dP}{dt} &= k_{pg}P\left(1 - \frac{P}{p_\infty}\right) - \frac{k_{pm}s_mP}{\mu_m + k_{mp}P} - k_{pm}f(N_A)P, \\
 \frac{dN_A}{dt} &= \frac{s_{nr}R}{\mu_{nr} + R} - \mu_nN_A, \\
 \frac{dD}{dt} &= k_{dn}f_s(f(N_A)) - \mu_dD, \\
 \frac{dC_A}{dt} &= s_c + \frac{k_{cn}f(N_A + k_{cmd}D)}{1 + f(N_A + k_{cmd}D)} - \mu_cC_A,
 \end{aligned}$$

where

$$R = f(k_{nn}N_A + k_{np}P + k_{nd}D), \quad f(V) = \frac{V}{(1 + (C_A/c_\infty)^2)} \quad \text{and} \quad f_s(V) = \frac{V^6}{x_{dn}^6 + V^6}.$$

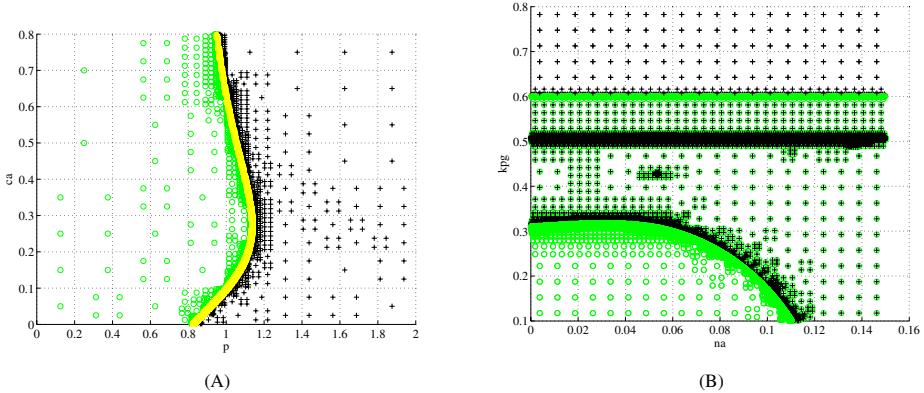
Here,  $k_*$ ,  $\mu_*$ ,  $s_*$ ,  $p_*$  are parameters, as defined in [26]. The state variables  $P$ ,  $N_A$ ,  $D$ , and  $C_A$ , correspond to the amounts of pathogen, pro-inflammatory mediators



**Fig. 3.** (Top) Examples trace from the 4-equation model. There are three different traces corresponding to septic death, aseptic death and health. (Bottom) Example traces from the 17-equation model; 5 of the 17 variables are shown. There are also three traces, illustrating the richer dynamics of the model. Two traces corresponds to aseptic death and the third to health with a periodic small resurgence of the pathogen. Time is measured in hours.

(e.g., activated neutrophils), tissue damage, and anti-inflammatory mediators (e.g., cortisol and interleukin-10), respectively. The 17-equation model, naturally, is far more detailed in terms of which mediators are modeled.

In each model, there are 3 clinically relevant outcomes: (i) a return to health, (ii) aseptic death, and (iii) septic death. Death is defined as a sustained amount of tissue damage ( $D$ ) above a specified threshold value and constitutes the undesirable or “bad”



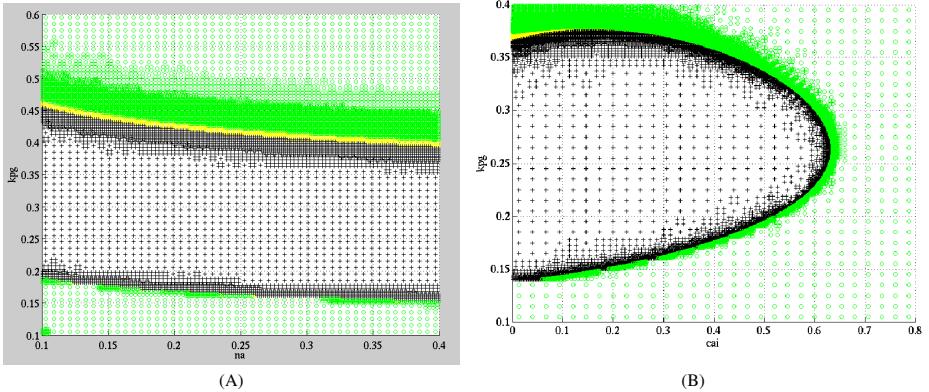
**Fig. 4.** Results obtained for the 4-equation model. Figure (A) reproduces results presented in Figure 8 of [26] with  $k_{pg} = 0.3$ , which was obtained using classical bifurcation analysis. Circles are parameter values leading to Health while crosses represent values leading to Death. Figure (B) illustrates how a pair of parameters ( $N_A$  and  $k_{pg}$ ) can be partitioned into the three possible outcomes. Circles alone lead to Health, crosses and circles lead to Aseptic Death and crosses alone lead to Septic Death. The separation between regions is induced from small uncertain regions computed by the algorithm.

outcome we wish to avoid. Aseptic and septic death are distinguished by whether the pathogen ( $P$ ) is cleared below a specified threshold value. Let  $\mathcal{F}_{alive}$  (resp.  $\mathcal{F}_{dead}$ ) refer to the set of states such that  $D$  is below (resp. above) some threshold  $D_{death}$ , and let  $\mathcal{F}_{septic}$  (resp.  $\mathcal{F}_{aseptic}$ ) refer to the set of states such that  $P$  is above (resp. below) some threshold  $P_{septic}$ . We can now define three sets of states corresponding to the three clinically relevant outcomes as follows: (i) *Health* =  $\mathcal{F}_{alive} \cap \mathcal{F}_{aseptic}$ ; (ii) *Aseptic death* =  $\mathcal{F}_{dead} \cap \mathcal{F}_{aseptic}$ ; and (iii) *Septic death* =  $\mathcal{F}_{dead} \cap \mathcal{F}_{septic}$ . In Figure 3 we present sample traces for both the 4 and 17 equation models.

#### 4.1 Experiments

We performed several experiments. In the first experiment, we validated our method by reproducing results previously obtained in [26] using bifurcation analysis. Figure 4-A contrasts the initial amount of pathogen,  $P_0$ , and the initial amount of anti-inflammatory mediators,  $C_{A0}$ . The growth rate of pathogen,  $k_{pg}$ , was set to 0.3 and other parameters to their nominal values. The region  $\mathcal{F}_{death}$  given by  $D \geq 5$  was used in our algorithm and we checked the intersection with reachable set at time 300 hours. Crosses correspond to initial values leading to death while circles lead to an healthy outcome. We can see that the resulting partition is quantitatively consistent with Figure 8 in [26]. In our second experiment, we varied growth rate of pathogen,  $k_{pg}$ , and  $N_A$ . Figure 4-(B) shows that there are three distinct regions in the  $k_{pg}$ - $N_A$  plane, corresponding to the three clinical outcomes.

We then performed several experimentations with the 17-equation model. Figures 5 (A) and (B) depict the  $k_{pg}$ - $N_A$  and  $k_{pg}$ - $C_{AI}$  planes, respectively.  $C_{AI}$  is a generic anti-inflammatory mediator. We partitioned the region using  $\mathcal{F}_{death} = D > 1.5$  and checked



**Fig. 5.** Results for the 17-equation model. Figure (A) illustrates the  $k_{pg}$ - $N_A$  plane, partitioned into regions leading to death (here aseptic death, represented by crosses) and regions leading to health (represented by circles). Figure (B) illustrates the  $k_{pg}$ - $C_{AI}$  plane. Interestingly, the separation is not monotone with the growth of pathogen  $k_{pg}$ .

after time 300 hours. The 17-equation model exhibits an interesting behavior in the  $k_{pg}$ - $C_{AI}$  plane. Namely, that the separation between health and death is not monotone in the growth rate of the pathogen.

As previously mentioned, our algorithm is implemented in Matlab and uses the CVODES numerical solver. Figures 4 (A) and (B) were generated in a few seconds, and Figures 5 (A) and (B) were generated in about an hour on a standard laptop (Intel Dual Core 1.8GHz with 2 Gb of memory). We note that the algorithm could easily be parallelized.

## 5 Discussion and Conclusions

Complex models are increasingly being used to make predictions about complex phenomena in biology and medicine (e.g., [3,25]). Such models can be potentially very useful in guiding early decisions regarding intervention, but it is often impossible to obtain accurate estimates for every parameter. Thus, it is important to have tools for explicitly examining a range of possible parameters to determine whether the behavior of the model is sensitive to those parameters that are poorly estimated. Performing this task for nonlinear models is especially challenging. We have presented an algorithm for solving the parameter synthesis problem for nonlinear dynamical models and applied it on two models of acute inflammation from the literature. The larger of the two has 17 equations and 79 parameters, demonstrating the scalability of our approach.

Our approach has several limitations. First, our refinement process implies that the number of partitions increases exponentially with the number of varying parameters. Thus, in practice, some variables must be held fixed while analyzing the behavior of the model. On the other hand, the number of state variables is not a limiting factor, as illustrated in our experiments on the 17-equation model. Second, our method relies on numerical simulations. Numerical methods are fundamentally limited in the context of verification since numerical image computation is not semi-decidable for nonlinear

differential equations [23]. Moreover, there are no known methods capable of providing provable bounds on numerical errors for general nonlinear differential equations. Thus, we cannot claim to provide *formal* guarantees on the correctness of the results computed by our method. However *asymptotic* guarantees exist, meaning that results can always be improved by decreasing tolerance factors in the numerical computations. A nice feature of our approach is that it enables one to obtain qualitative results using a few simulations (e.g. a coarse partition between regions leading to qualitatively different behaviors). These qualitative results can then be made as precise as desired by focusing on smaller partitions.

There are several areas for future research. Our first order error control mechanism can be improved to make the refinements more efficient and more adaptive when non-linear (i.e. higher order) behaviors dominate any linear dependance on parameter variations. We are also interested in developing techniques for verifying properties that are more complex than the reachability predicates we considered in this paper. For instance, temporal properties could easily be introduced in our framework. The extended system could then be used, for example, verify the possible outcomes associated with a particular medical intervention. Finally, we believe that the method could easily be used in the context of personalized medicine. In particular, given individual or longitudinal measurements from a specific patient, we could define a reachable set,  $\mathcal{F}_{obs}$ , that includes these observations (possibly convolved with a model of the measurement errors). We could then use our method to identify the set of parameters that are consistent with the observations. The refined parameters can then be used to make patient-specific predictions.

## Acknowledgments

This work is supported in part by US Department of Energy Career Award and a grant from Microsoft Research to CJL.

## References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. *Theoretical Computer Science* 138(1), 3–34 (1995)
2. Alur, R., Henzinger, T., Lafferriere, G., Pappas, G.: Discrete abstractions of hybrid systems, vol. 88(7), pp. 971–984. IEEE, Los Alamitos (2000)
3. An, G.: Agent based computer simulation and sirs: building a gap between basic science and clinical trials. *Shock* 16, 266–273 (2001)
4. Annichini, A., Asarin, E., Bouajjani, A.: Symbolic techniques for parametric reasoning about counter and clock systems. In: Emerson, E.A., Sistla, A.P. (eds.) *CAV 2000. LNCS*, vol. 1855, pp. 419–434. Springer, Heidelberg (2000)
5. Asarin, E., Dang, T., Girard, A.: Hybridization methods for verification of non-linear systems. In: *ECC-CDC 2005 joint conference: Conference on Decision and Control CDC and European Control Conference ECC* (2005)

6. Batt, G., Belta, C., Weiss, R.: Model checking genetic regulatory networks with parameter uncertainty. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 61–75. Springer, Heidelberg (2007)
7. Batt, G., Belta, C., Weiss, R.: Model checking liveness properties of genetic regulatory networks. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 323–338. Springer, Heidelberg (2007)
8. Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D.: Analysis and verification of qualitative models of genetic regulatory networks: A model-checking approach. In: IJCAI, pp. 370–375 (2005)
9. Cinquemani, E., Milias-Argeitis, A., Lygeros, J.: Identification of genetic regulatory networks: A stochastic hybrid approach. In: IFAC World Congress (2008)
10. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
11. Donzé, A., Krogh, B., Rajhans, A.: Parameter synthesis for hybrid systems with an application to simulink models. In: Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control (HSCC 2009). LNCS. Springer, Heidelberg (2009)
12. Donzé, A.: Trajectory-Based Verification and Controller Synthesis for Continuous and Hybrid Systems. PhD thesis, University Joseph Fourier (June 2007)
13. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 174–189. Springer, Heidelberg (2007)
14. Drulhe, S., Ferrari-Trecate, G., de Jong, H., Viari, A.: Reconstruction of switching thresholds in piecewise-affine models of genetic regulatory networks. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 184–199. Springer, Heidelberg (2006)
15. Fainekos, G.E., Pappas, G.J.: Robust sampling for MITL specifications. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 147–162. Springer, Heidelberg (2007)
16. Fagot, E., Gouze, J.-L.: How to control a biological switch: a mathematical framework for the control of piecewise affine models of gene networks. Technical report, INRIA Sophia Antipolis (2006)
17. Frehse, G., Jha, S.K., Krogh, B.H.: A counterexample-guided approach to parameter synthesis for linear hybrid automata. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 187–200. Springer, Heidelberg (2008)
18. Ghosh, R., Tomlin, C.: Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch signalling. System Biology (2004)
19. Henzinger, T.A., Horowitz, B., Majumdar, R., Wong-Toi, H.: Beyond hytech: Hybrid systems analysis using interval numerical methods. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 130–144. Springer, Heidelberg (2000)
20. Kumar, R.: The Dynamics of Acute Inflammation. PhD thesis, University of Pittsburgh (2004)
21. Mitchell, I., Tomlin, C.: Level set methods for computation in hybrid systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 310–323. Springer, Heidelberg (2000)
22. Mitchell, I.M., Tomlin, C.J.: Overapproximating reachable sets by hamilton-jacobi projections. J. Symbolic Computation 19, 1–3 (2002)
23. Platzer, A., Clarke, E.M.: The image computation problem in hybrid systems model checking. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 473–486. Springer, Heidelberg (2007)

24. Pnueli, A.: The temporal logic of programs. In: Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS), pp. 46–57 (1977)
25. Polidori, D., Trimmer, J.: Bringing advanced therapies to marker faster: a role for bio-simulation? *Diabetes' Voice* 48(2), 28–30 (2003)
26. Reynolds, A., Rubin, J., Clermont, G., Day, J., Vodovotz, Y., Ermentrout, B.: A reduced mathematical model of the acute inflammatory response: I. derivation of model and analysis of anti-inflammation. *J. Theor. Biol.* 242(1), 220–236 (2006)
27. Serban, R., Hindmarsh, A.C.: Cvodes: the sensitivity-enabled ode solver in sundials. In: Proceedings of IDETC/CIE 2005, Long Beach, CA (September 2005)
28. Stursberg, O., Krogh, B.H.: Efficient representation and computation of reachable sets for hybrid systems. In: Maler, O., Pnueli, A. (eds.) *HSCC 2003*. LNCS, vol. 2623, pp. 482–497. Springer, Heidelberg (2003)

# Spatial Clustering of Multivariate Genomic and Epigenomic Information

Rami Jasichek and Amos Tanay

Department of Computer Science and Applied Mathematics  
The Weizmann Institute of Science, Rehovot, 76100 Israel

**Abstract.** The combination of fully sequenced genomes and new technologies for high density arrays and ultra-rapid sequencing enables the mapping of gene-regulatory and epigenetics marks on a global scale. This new experimental methodology was recently applied to map multiple histone marks and genomic factors, characterizing patterns of genome organization and discovering interactions among processes of epigenetic reprogramming during cellular differentiation. The new data poses a significant computational challenge in both size and statistical heterogeneity. Understanding it collectively and without bias remains an open problem. Here we introduce spatial clustering - a new unsupervised clustering methodology for dissection of large, multi-track genomic and epigenomic data sets into a spatially organized set of distinct combinatorial behaviors. We develop a probabilistic algorithm that finds spatial clustering solutions by learning an HMM model and inferring the most likely genomic layout of clusters. Application of our methods to meta-analysis of combined ChIP-seq and ChIP-chip epigenomic datasets in mouse and human reveals known and novel patterns of local co-occurrence among histone modification and related factors. Moreover, the model weaves together these local patterns into a coherent global model that reflects the higher level organization of the epigenome. Spatial clustering constitutes a powerful and scalable analysis methodology for dissecting even larger scale genomic dataset that will soon become available.

## 1 Introduction

The combination of fully sequenced genomes and new technologies for high density arrays (ChIP-chip) and ultra-rapid sequencing (ChIP-seq) enables the mapping of gene-regulatory and epigenetics marks on a global scale. Such mapping is being employed at an increasing pace to study genomic and epigenomic organization at different developmental stages [1-5]. The new massive experimental data has already revealed how genomes are programmed and reprogrammed by complex patterns of histone marks, transcription factors and regulatory complexes [6]. It was suggested that a mechanistic understanding of normal [7] and malignant [8, 9] differentiation processes can be facilitated through a comprehensive analysis of multiple marks and factors in multiple cell systems [10]. The great promise of the new datasets lies in their lack of bias and truly genomic scale. To fully exploit their potential, one must develop an adequate analysis methodology that can go beyond the study of a single histone mark or transcription factor. The goal is to comprehensively identify complex

genomic structures without an a-priori focus on known features (e.g., genes and promoters).

Current approaches for analyzing genomic information focus on the distributions of values relative to transcription start sites (TSS) or other genomic features (e.g., CTCF binding sites [11]). Computing such distributions is computationally easy and informative: one can depict the relations between TSSs and the profiled factors through graphs of the average factor/mark occupancy as a function of the distance to the nearest TSS. Comparing the profiles near active and inactive TSSs can highlight possible functional implications for the profiled distributions. Heat maps are used to show possible correlations between modification pairs [5]. While being easy to understand and very effective in mapping the organization around TSSs, the averaging methods provide little help when trying to understand the datasets as a whole. Focusing on the patterns around specific features does not enable the identification of novel genomic structures or higher level organization. The comprehensive and innovative nature of the experiments is therefore still unmatched analytically.

The problem of identifying patterns in large datasets is a hallmark of computational biology. Extensive literature is dealing with the analysis of gene expression data, dissecting it into clusters [12, 13] or biclusters [14] or in explaining the data by means of a complex regulatory model [15]. Clustering became the method of choice for analysis of gene expression, mostly due to its simple and unsupervised nature and since it allows effective visualization of the entire dataset by means of a few robust structures. The new generation of ChIP-chip and ChIP-seq datasets is however not easily approached using naïve clustering. At the most technical level, the datasets are huge and cannot be analyzed using the current algorithms' implementations. More fundamentally, the genomic datasets are spatially arranged over chromosomes and their analysis must account for this organization. Present clustering methodologies are inadequate for analysis of multi-track and heterogeneous ChIP-chip and ChIP-seq data.

In this paper we introduce the spatial clustering problem and describe our probabilistic algorithm for solving it. Our algorithm clusters a set of genomic profiles (tracks), representing epigenetic modifications, factor occupancy or other spatially distributed data. We model the data using an HMM which is being learned in an unsupervised fashion. The algorithm then infers the most likely coverage of the genome with contiguous spatial clusters. The HMM component of our model can flexibly express local structure (short genomic intervals with similar profiles) and global structure (groups of clusters that tend to co-occur and form larger domains). We demonstrate this by analyzing two combined whole genome datasets including epigenomic profiles of human cells and differentiating mouse stem cells. In both cases, our analysis provides a comprehensive map of epigenomic modes that extend beyond the reported patterns around TSSs. Spatial Clustering is a flexible and robust tool that is designed to meet the requirements of large genomic and epigenomic projects. It provides a powerful alternative to the limited and supervised analysis scheme which is currently in common use.

## 2 Methods

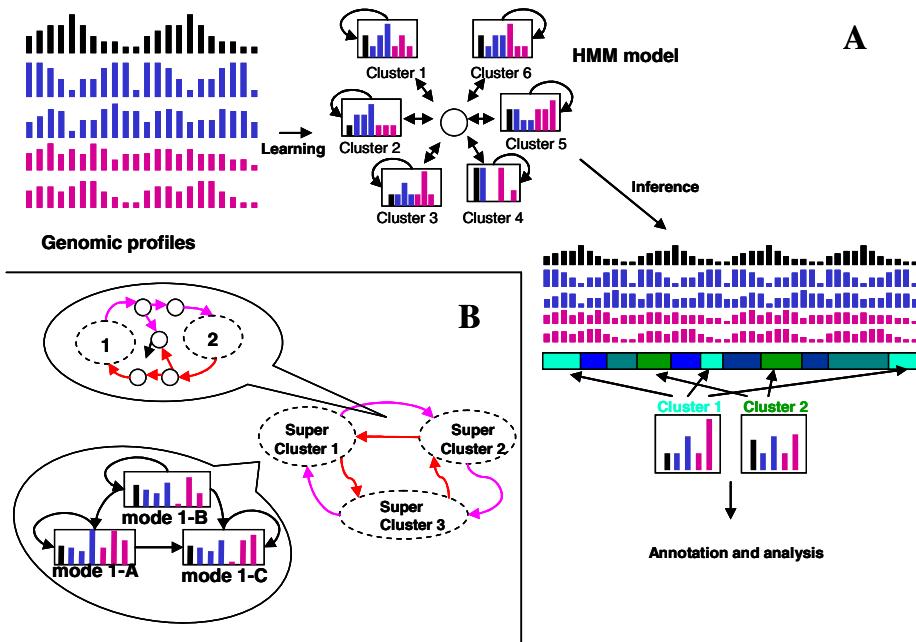
**The K-Spatial Clustering problem.** Given a multivariate genomic dataset, we seek a representation of the data using a limited repertoire of distinct behaviors that are

spatially organized. We formalize this notion as the K-Spatial clustering problem. Assume that we are studying a linearly organized (possibly on more than one interval) set of genomic measurement loci (or probes). Loci can be ordered along complete chromosomes at fixed distances (as in binned ChIP-seq data), or may tile selected parts of the genome at variable distances (as in ChIP-chip data). For simplicity, we will look only at the sequential position of each locus inside its surrounding contiguous segment, breaking chromosomes into segments wherever there is large gap between two adjacent probes. Assume also that we are given a vector of experimental measurements at each of the loci. The experimental data can come from different types of experiments or different cell types and conditions, but we disregard any available information on a-priori connections between vector entries. A K-Spatial clustering is a partition of the underlying genomic region into disjoint and contiguous intervals and a tagging of these intervals with cluster numbers [1..K]. In the most general settings, we introduce a quality function that scores K-Spatial clustering instances given the genomic dataset and defines the K-Spatial clustering problem as an optimization problem that seeks the maximal scoring K-Spatial clustering. We note that in the degenerate case, when the scoring function ignores the genomic layout of the data and is based solely on similarity of measured values per loci, the K-Spatial clustering problem is equivalent to a standard clustering problem. Needless to say, in any reasonable application, the quality function will take advantage of the genomic organization of the data to derive better solutions.

Spatial clustering can be considerably more informative and powerful than simple clustering of the probes. First, in most practical cases, the experimental values for adjacent probes are highly correlative as important biological features would typically span several measurement loci. These effects are making the common assumptions underlying most clustering frameworks incorrect (i.e., the a-priori independence among samples is not holding). A good K-Spatial clustering solution would therefore maximize the number of adjacent loci that are part of the same cluster (reducing the overall number of intervals) while not compromising the integrity and specificity of the clusters. A second source of spatial information works at a higher level. In many genomic datasets we can observe coupling between related phenomena. For example, transcription start sites (TSS) would often be followed by a transcribed region. Expressing the couplings between clusters, and introducing it into the quality function can have an important contribution to the quality of the results, and to our ability to understand them.

**Probabilistic K-Spatial clustering.** Our approach for deriving K-Spatial clustering is based on a probabilistic formulation of the problem (Fig. 1A). This is analogous to the standard approach that estimates a mixture model (for example, a mixture of multivariate Gaussians) to cluster large data sets [16]. We extend the naïve mixture model using an HMM structure that expresses the tendency of adjacent loci to remain in the same cluster and the spatial coupling among clusters. Specifically, the model is defined using K multivariate distributions over the experimental tracks and an HMM model connecting the clusters (with additional hidden states, see below). The topologies we shall consider for the HMM graph reflect constraints on higher level organization of clusters across the genome. Given a fixed topology, we apply the expectation maximization (EM) algorithm to learn the model and its parameters and

then compute posterior probabilities for the association between measurement loci and HMM clusters. The set of contiguous intervals associated with the same HMM state (with high posterior probability) are then used as our spatial clustering. We can also study the parameters of the distributions defining each cluster and the transition probabilities between clusters to get a more global picture of the model and its implications.



**Fig. 1. A) Spatial clustering of genomic profiles (tracks).** Multi-dimensional genomic datasets, including heterogeneous ChIP-seq and ChIP-chip experiments as well as other genomic sources of information are analyzed together using a probabilistic hidden Markov model. The model is then used to infer the most likely partition of the genome into spatial clusters, each representing a specific genomic or epigenomic behavior which is determined based on the distribution of all data tracks. **B) Hierarchical spatial clustering.** Shown is a schematic view of the HMM topology we use for building hierarchical spatial clustering. The model consists of a set of small complete graphs (*super clusters*, here on 3 states) that are connected through dedicated connector state pairs. Transition probabilities inside each connector pair are fixed throughout the learning process and add a penalty for crossing super cluster boundaries.

**Local distribution parameterization.** In our generative model, multi-track data is emitted from cluster states given a probability distribution associated with the cluster. To learn the model, we need to specify a family of distributions appropriate for the tracks we analyze. A good selection of distributions family is one that will be able to generate, given the correct parameters, a distribution of values that is as close as possible to the one observed, and would also allow robust learning with limited data. The simplest class of distributions assumes tracks are generated independently (once the cluster is determined). This simplification, which we use in the present work, is economical in terms of parameters and allows for learning robustly, even when the

number of tracks grows. More refined classes of distributions (e.g., multinormal distribution with arbitrary covariance matrices, [8]) are currently practical only for smaller number of tracks and should be further developed to allow robust learning in general.

The most common sources of comprehensive genomic data are ChIP-chip and ChIP-seq experiments. Results from ChIP-chip experiments are real valued binding ratios. ChIP-seq results are lists of sequenced reads which are normalized into some coverage statistics on genomic intervals. For ChIP-chip data, the combination of experimental and biological noise can be modeled using simple normal distributions. ChIP-seq results are by nature discrete, and should be theoretically distributed as a combination of samples from two fragment pools (false positives and enriched IP fragments [4, 17]). However, according to our analysis, the empirical distribution of ChIP-seq tracks cannot be effectively approximated as a mixture of noise and a geometric distribution, featuring a very heavy tail (data not shown). We therefore model ChIP-seq tracks using an a-parametric discrete distribution on variable sized bins. We generate the bins by identifying, for each track, the values of the  $(1-2^k)$  percentiles (for  $k=1$  to 10). The distributions considered for the track are now defined using a discrete distribution over 10 bins, where we map physical measurements to bins with values in the percentile intervals  $[(1-2^{k+1}):(1-2^k)]$ .

**The HMM structure.** We use the HMM structure to impose constraints on clustering solutions, demanding clusters would occupy contiguous genomic intervals and coupling together clusters that are frequently occurring next to each other. We use several structure families with increasing degrees of details. In its most simple form, the model has a star structure, which is associating all cluster states through a central hidden (non-emitting) state. The structure is in essence a simple mixture model (transitions probabilities from the central states to cluster states correspond to the mixture coefficients), but uses the states self transitions to increase the likelihood of solutions with contiguous clusters. The star topology imposes no constraint on transitions between specific emitting clusters pairs (all transitions must go through the central hidden node). Such transitions can be inferred from the posterior state probabilities in post process. Alternatively, an HMM over a clique topology (connecting all pairs of cluster states) is by far more expressive, but may be too detailed to allow robust learning, especially when the number of clusters increase. We note that even the clique topology can capture only coupling between pairs of clusters and cannot be used to represent higher order structures, which are frequently observed in genomic data.

To try and model higher order genomic structure, we are using a hierarchical topology (Fig. 1B). In this scheme, we construct small clique HMMs (super-clusters) to represent specific genomic structures that tend to co-occur over larger domains. We then connect super clusters using dedicated connector state pairs that implicate a probabilistic payment for each transition between super clusters. In the current framework we have worked with two levels of hierarchy, but these can be naturally generalized. More specific HMM topologies can be developed to model additional biological phenomena. For example, genomic structure is often polarized according to the direction of the nearby transcript, but our basic HMM implementation reads all data in the same order and can therefore learn only unidirectional couplings. To improve on that, we can form two copies of our model, each representing one genomic

polarization, such that all transition probabilities are reverse symmetric between the two copies.

**Model Learning.** Given an HMM topology, we learn the model parameterization using a standard Expectation-Maximization (EM) algorithm [18]. The success of the local optimization performed by the EM algorithm depends, as is often the case, on careful selection of initial conditions. The most critical initialization for the spatial clustering model is proper selection of initial cluster state emission distributions. In our implementation, we first preprocess the data to identify standard clusters (ignoring spatial information). We then use a repertoire of detected clusters to initialize cluster states over some initial topology. The clustering algorithm in the preprocessing stage can be selected arbitrarily. The implementation we report here consist of three phases: a) We discretize all data to three levels (-1,0 and 1) using predefined Z-score thresholds for each track. b) We group together all probes with the same discretized multi-track vector and count how many probes are classified for each vector. c) For each discretized vector with a sufficient number of associated probes, we estimate from the original data the (non discrete) multivariate distribution over the tracks and add it to the set of potential initial cluster states (or seeds). In case too few heavy clusters are available, we adjust the Z-score threshold and return to step a. Given a set of seed states, we generate an initial model by randomly selecting initial cluster states from the pool of generated seeds. Our analysis suggests that testing few dozens of initial conditions is performing effectively, and is comparable in performance to a greedy scheme in which we construct the model state by state, at each step testing EM after addition of each of the seeds and choosing the seed with the maximal likelihood gain (data not shown).

To learn a hierarchical model we work from top down. We first learn an HMM model on K states using a star topology. We then use inference with the derived model to partition the data into K sets, one for each cluster. We apply the initialization procedure described above separately for each of the K probes sets, and construct an initial model by combining together K clique models (each with L states) using the connector states described above (Fig. 1B).

Our unsupervised approach to the learning problem provides us with a robust and unbiased way to analyze the data. The approach can however lead to models that are optimized toward better representation of the experimental noise in the genomic background rather than models outlining meaningful biological effects. A large percentage of the genome may be showing little or no meaningful signal. Modeling these parts of the genome accurately yields a high likelihood gain – pushing the learning process to use additional clusters for refined background models. We can somewhat control this problem by increasing the total number of clusters in the model. More heuristically, we can also employ constraints to the learning process, limiting the variance of the emission distributions so that states cannot be too tuned for a specific background behavior (by having small variance), or too broad to absorb several biological phenomena (by having large variance).

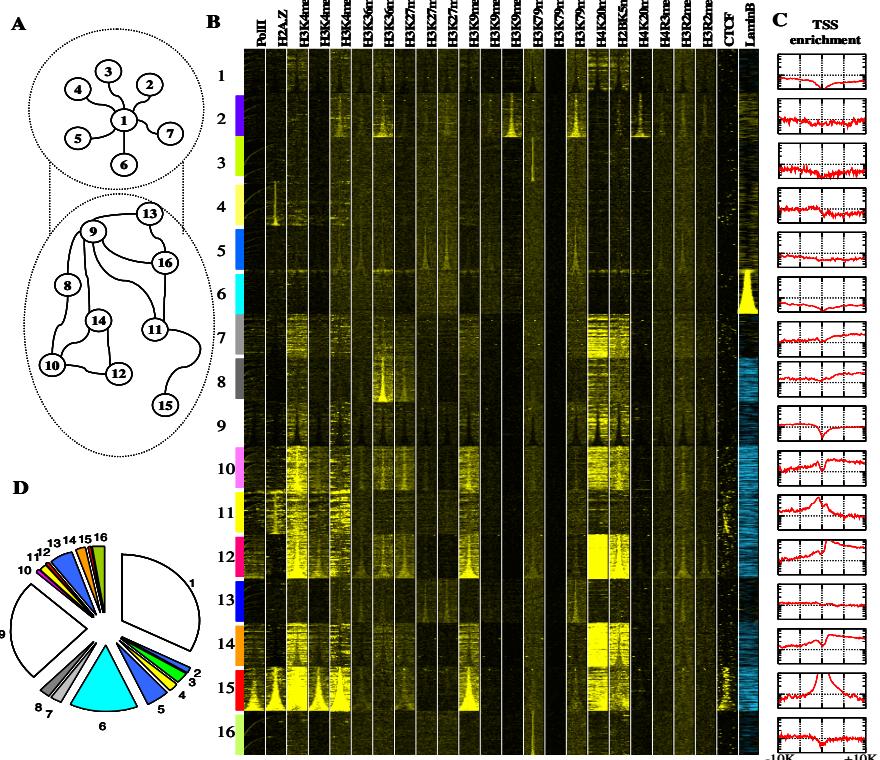
We note that the datasets our algorithm is handling are imposing new kinds of technical challenges on design and implementation. For example, the largest gene expression datasets analyzed thus far are limited to  $\sim 10^4$  genes and  $\sim 10^3$  conditions. The dataset we have analyzed here have  $\sim 10^8$  probes (tiling the human genome in

25bp-50bp resolution) and  $\sim 10^2$  tracks, making it 2-3 order of magnitudes larger. These numbers are expected to increase further. We are therefore forced to utilize considerable computational resources even if the algorithm is extremely efficient. In our implementation, the EM inference (Forward-Backward) step is massively parallelized over a computer cluster, allowing learning on large genomes to be completed quickly.

**Visualization.** The most direct visualization scheme for genomic data is as data tracks in genome browsers [19, 20]. When the data includes many tracks this scheme may prove difficult to follow, and at any rate can only provide local information on a specific locus and not global understanding on how the data is organized. Another common approach is to average the available genomic profiles with respect to a genomic feature (usually TSSs), but as argued above, this approach is strongly biased to a specific phenomenon and may miss important genomic structures. Our spatial cluster model opens the way to new visualization schemes of complex genomic data. We use a learned model to infer the most probable cluster state associated with each locus. We can then color code the genome according to the associated clusters in a way that summarizes all available experimental profiles in one color per locus. With appropriate selection of colors, this can be an effective way to identify both global and local behavior (see below). A complementary approach is attempting to visualize the entire data set in one figure. To do this we identify contiguous intervals associated with a cluster by looking at ranges of probes with consecutive high posterior probabilities for the same HMM state. We then pool together groups of intervals that were associated with the same cluster and plot the genomic profiles inside and around them in standard cluster-gram (each row represents an interval and its margins). From our experience, although the color coding approach is somewhat qualitative, it is currently the best way to rapidly understand the entire dataset in one view, providing a good starting point for further analysis.

### 3 Results

**Spatial clustering model for human T cells epigenetics.** Barks et al. [1] have used ChIP-seq and a collection of antibodies for 20 histone methylation marks, RNAP, the Histone variant H2A.Z and CTCF to globally map the epigenome of human T-cells. This constitutes the most comprehensive epigenomic dataset on a single mammalian cell type to date. TSS averaging analysis of the data confirmed and extended known principles of chromatin organization around active and repressed TSSs [5]. Recently, work from the Van Steensel group has put forward a genome wide map of chromatin interactions with the nuclear lamina [21], characterizing large lamina-associated domains that are very gene sparse and flanked by transcriptional units, insulators and H3K27 trimethylation. In an attempt to gain an unbiased view on the interactions among the profiled histone marks and other factors, we have combined these two datasets and applied spatial clustering analysis to them. We note that the two sets of experiments we used were derived from different cell types (T-cells vs. fibroblasts) and different technologies. To allow common analysis, we transformed all ChIP-seq data to coverage statistics on 50bp bins, assuming fragment length of 300bp



**Fig. 2. Spatial clustering of the human epigenome.** **A)** **HMM basic topology.** Shown are the model states, where we connect pairs of states that frequently follow each other. **B)** **Color coded visualization of clusters and their flanking region.** Each row segment represents the occupancy of one mark or factor over one spatial cluster and its flanking region, where we stack together loci that were associated to the same cluster (blocks). The color coding is providing a quick visualization of the complex model, for a more quantitative view consult <http://www.wisdom.weizmann.ac.il/~ramij/recomb2009.html>. **C)** **TSS enrichments.** The graph shown indicates for each cluster and offset from the TSS (X axis) the ratio between the fraction of loci associated with the cluster at that TSS offset to the overall fraction of genomic loci at that TSS offset (Y axis, 0.2 to 6, log scale). **D)** **Cluster coverage.** Shown are the fractions of genomic loci covered by each of the clusters. Color coding is similar to panel B.

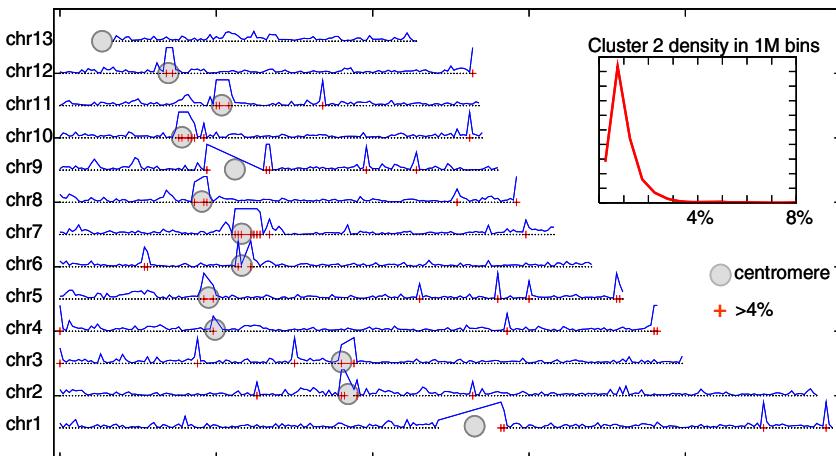
(as described in [4]). We also assumed that each Lamin B1 tiling array probe represents the occupancy at all 50bp bins in the range of 1000bp around the probe.

In Fig. 2A we show the major topological interactions in the spatial cluster model we derived and in Fig. 2B we present a color coded clustergram of the inferred clustering, depicting the profiles in and around the clusters (Each row represent one spatial cluster at one genomic locus, rows are grouped according to their associated cluster state, see Methods). A more quantitative summary of Fig. 2 is shown in <http://www.wisdom.weizmann.ac.il/~ramij/recomb2009.html>. We used the hierarchical learning process described above, with two clusters at the first phase and eight clusters per super-cluster in the second phase. The algorithm chose to first partition

the genome according to the strength of interaction with the nuclear lamina (upper clusters – strong interaction, lower clusters – no interaction) and then constructed a detailed model to describe different combinations of histone modification and factor occupancy and their couplings. To further illustrate the possible relationships between the clusters and genes we computed the enrichment of cluster associations as a function of the distance from the TSSs (Fig. 2C).

Several clusters we detected represent known structure around TSSs which the algorithm rediscovered in a completely unsupervised fashion and without using information on the TSSs locations. Cluster 15 is the only one with high levels of RNA PolII, and is further associated with very high H3K4me3 levels and significant enrichments of H2A.Z. The cluster is also enriched with H3K9me1, and is sometime observed with CTCF binding. CTCF binding was reported before to occur at alternative promoters sites [22], and this may explain the partial co-occurrence between RNAP and CTCF at cluster 15. Cluster 12 and cluster 10 represent a combination of mono methylation at multiple positions (H3K4me1, H3K36me1, H3K9me), all of which were associated before with the regions flanking the TSS. For cluster 12, there are very strong enrichments of H4K20me1 and H2BK5me1 and a detected preference for the regions downstream the TSS, while for cluster 10, little H4K20me1 and H2BK5me1 enrichment is detected. Each of the clusters are covering about 2% of the genome (Fig. 2D). The five monomethylation marks (at H3 K4, K9, K36, H4K20 and H2BK5) were associated before with active chromatin, but here we see that there are at least two modes of correlation among them. We note that since we observe such complex pattern, general monomethylation antibody specificity is unlikely to explain these patterns. The monomethylation marks may still share a mechanism that will explain their high degree of correlation. Cluster 8 is representing enriched trimethylation at H3K36, which is largely free of other marks and is found in the longer range downstream of transcribed genes. This modification was previously associated with transcription elongation. Cluster 11 shows insulator patterns, including hotspots of CTCF and H2A.Z, with a preference to the region upstream of the TSS.

Other clusters are not tightly associated to TSSs and represent weaker enrichments than the clear preferences discussed above. They still show a high degree of correlation between marks and can provide insights into the organization of chromatin out of the TSS context. Cluster 6 represents cores of Lamin B1 interaction and is not linked to any of the histone modifications. All clusters associated with the lamina are generally void of activation mark, as noted before, and even though the datasets compared two different cell types. Clusters 5 and 13 are characterized by the polycomb modifications, H3K27me3 and H3K27me2 and are occupying over 10% of genome, mostly in intergenic regions. Polycomb marks were characterized extensively in embryonic stem cells, and domains with strong H3K27me3 enrichment were linked to gene repression in genes poised for later activation upon differentiation. Cluster 5 and 13 may represent a broader and weaker pattern of polycomb marks, reminiscent of recent evidence on large H3K27me3 domains or large scale polycomb domains in flies [23]. A different type of repressive mark is H3K9me3 which is the main distinguishing feature of cluster 2. Interestingly, cluster 2 is also associated with H4K20me3 which was observed before to correlate with H3K9me3 [24] but was not suggested to co-occur with H3K9me3 in the original TSS-centric analysis of the data [1]. Here we also observe association of the H3K9me3 heterochromatic mark with trimethylation



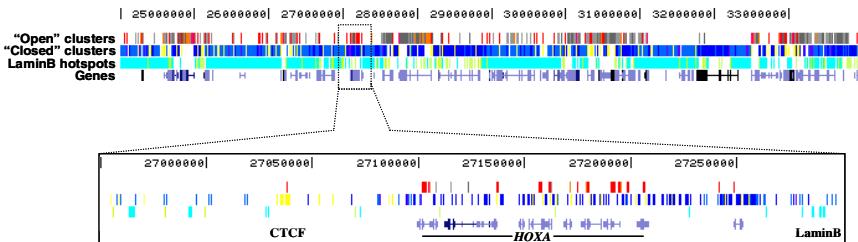
**Fig. 3. Hotspots of H3K9me3/H3K79me3/H4K20me3 (cluster 2) enrichment.** We computed the fraction the genome covered by cluster 2 in bins of 1MB. Shown is the overall distribution of these fractions (upper right) and how it is spatially organized in chromosome 1-13 (main figure, blue curves). Hotspots (fraction over 4%) are marked in red and are preferentially occurring in pericentromeric regions.

at H3K79, in concordance with recent evidence on H3K79me3 pericentromeric localization in mice [25] and with results derived from the same dataset using a local pattern search approach [26]. Finally, two clusters (3 and 16) are characterized by cryptic enrichment of H3K79me1 – with unclear functional or organizational specificity.

Beyond the set of clusters and their properties, the spatial cluster model also reflects higher level genomic organization. As shown in Fig. 3, cluster 2 occupancy (Representing trimethylation in H3K9, H3K79 and H4K20) is distributed in a highly non uniform fashion across the genome, with mean occupancy of less than 1%, but a significant number of 1MB genomic bins with more than four fold that number. As expected, the strongest cluster 2 hotspots are observed in pericentromeric regions, but many additional hotspots are apparent. Fig. 3 provides a general reference for the organization of H3K9me3 heterochromatin in CD4+ T-cells.

In Fig. 4 we illustrate the clustering of a specific chromosomal region around the HOXA gene family. The higher level view shows domains of "open" chromatin (red and gray bars) packed between larger domains of repressed ("closed") chromatin (blue bars: clusters 5,13 and 2, yellow bars: cluster 11). "Closed" clusters are frequently co-occurring with lamina clusters (light blue: cluster 6). The 100KB around the HOXA genes are unique in showing both active (red) and repressive (blue) marks. A higher resolution view (lower panel) shows activation domains at key HOXA gene promoters, and insulator/lamina domains flanking the entire region.

**Spatial clustering model for differentiation of mouse embryonic stem cells.** The genomes of embryonic stem cells (ESC) are uniquely organized to ensure pluripotency and maximize flexibility upon a differentiating signal. Genome wide maps of key histone marks, involving the trithorax- (H3K4) and Polycomb- (H3K27)

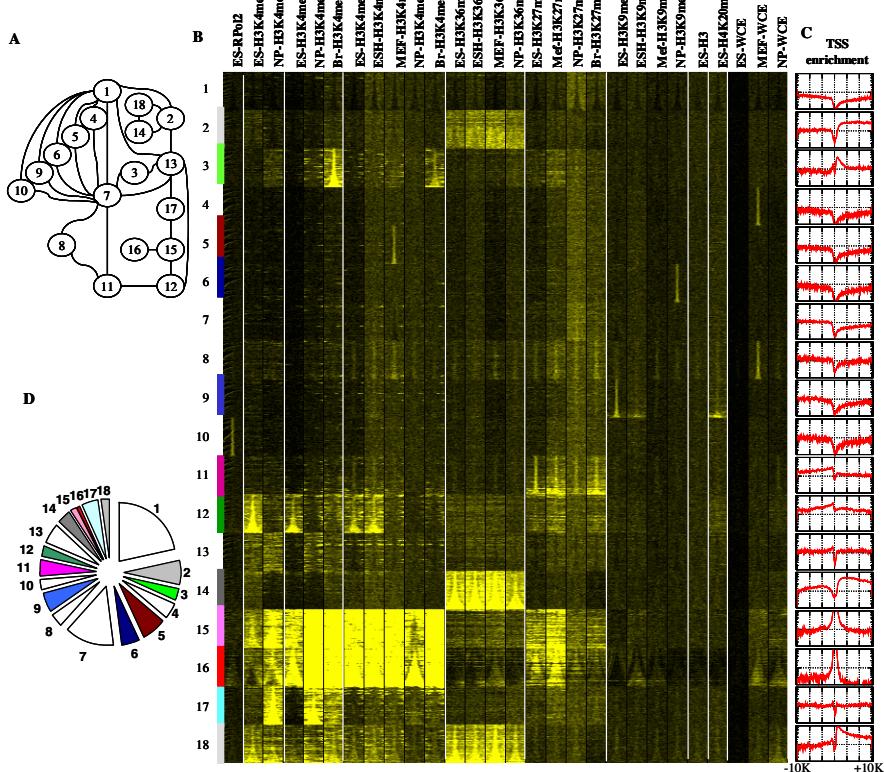


**Fig. 4. Spatial clustering in the HOXA region.** Shown is the clustering of some 10MB in the human chromosome 7, color coded according to the clustering shown in Fig 2. Genes are marked in the lower track.

associated modifications have revolutionized our understanding of the pluripotent epigenetic state [4, 27, 28]. To analyze the mouse ESC epigenomic state we applied spatial clustering to a combined ChIP-seq dataset [3, 4]. The data set included only few of the histone marks that were analyzed above for human T-cells, but allowed comparison between different cell types. To apply our algorithm, we used ChIP-seq coverage statistics in bins of 50bp as described [4] and derived a model including 3 super-clusters and 6 sub clusters in each of them. We also used DNA methylation data from the same experiments, but since their overall genomic coverage was very low, we omit them from the discussion.

As shown in Fig. 5 the spatial clustering reveals both cell-type conserved and cell type specific patterns (compare also <http://www.wisdom.weizmann.ac.il/~ramij/recomb2009.html> for a better quantitative resolution). As expected, and similarly to the pattern observed in somatic human cell (Fig. 2), the algorithm generated a cluster (#16) that is defined by strong H4Kme3 presence and is observed almost exclusively at annotated TSSs. Cluster 15, which is coupled by the HMM to cluster 16, represents a region with dominant H3K4me1 levels, which are conserved between cell types (see <http://www.wisdom.weizmann.ac.il/~ramij/recomb2009.html>) and clusters 18,14 (and to a lesser extent 2) are based on cell-type conserved H3K36me3 elongation marks. Clusters 17 and 12 represent H3K4me1 domains that are specific to mESC or NP cells. As noted before [29], the active chromatin state in embryonic stem cells is frequently co-occurring with high levels of H3K27me3, and this is reflected in cluster 16. On the other hand, cluster 11 represent domains in which H3K27me3 is the only significantly enriched mark, with very mild bias to promoters and conserved intensity between stem cells and derived lineages (in contrast to the decreasing H3K27me3 intensity in cluster 17).

As observed in the human datasets, we observe clusters of H3K9me3 activity. Cluster 9 shows H3Kme9 in mESC, with some matching H4K20me3 co-occurrence. Perhaps surprisingly, this pattern is not conserved in NP cells, which have their own H3K9me3 cluster (#6). It is unclear if this represents real plasticity of these heterochromatic marks, or experimental limitations (for example, cluster 8 and 4 isolate experimental artifacts by detecting clusters that are defined by elevated coverage in whole cell extract controls). These questions should be further addressed experimentally.



**Fig. 5. Spatial clustering of the mouse ESC epigenome.** A) **HMM topology.** Shown are the model states, where we connect pairs of states that frequently follow each other. B) **Global view.** Color coded visualization of clusters and their flanking region. Each row segment represents the occupancy of one mark or factor over one spatial cluster and its flanking region, where we stack together loci that were associated to the same cluster (left color coded blocks). The color coding is providing a quick visualization of the complex model, but may become saturated (e.g., in clusters 15 and 16). For a more quantitative view consult <http://www.wisdom.weizmann.ac.il/~ramij/recomb2009.html>. C) **TSS enrichments.** The graph shown indicates for each cluster and offset from the TSS (X axis) the ratio between the fraction of loci associated with the cluster at that TSS offset to the overall fraction of genomic loci at that TSS offset (Y axis, 0.2 to 6, log scale). D) **Cluster coverage.** Shown are the fractions of genomic loci covered by each of the clusters. Color coding is similar to panel B.

## 4 Discussion

We have presented spatial clustering as a new analysis methodology for dissecting large ChIP-chip and ChIP-seq datasets into defined clusters of common genomic or epigenomic behavior. The new method is allowing unsupervised and global modeling of the data, in a way that matches the unbiased and comprehensive nature of the experiments. It represents the entire genome as an organized set of contiguous clusters, and is capable of capturing both the nature of each cluster and the relations between

them, something that is not available in local views [26]. Spatial clustering does not assume any gene structure or information on TSSs for defining clusters, and can therefore be used to study both TSS-related and TSS-unrelated genomic phenomenon. The model is constructing patterns that are based on a combined behavior over all experimental tracks and therefore scales well with increasing number of experiments. Spatial clustering can be the first line of analysis for genomic data, serving as a starting point for more careful hypothesis testing in a way similar to that by which standard clustering is used for gene expression analysis.

The data we analyzed here is providing us with a comprehensive view of the epigenomic structure of human T-cells and differentiating mouse ESCs. The analysis concisely summarizes known chromatin modes and reveals some new testable associations between histone marks (e.g., H3K9me3 with H3K79me3). The results emphasize the need for an integrative and coherent model that can broadly combine multiple epigenomic datasets and derive architectural insights. Ultimately, such a model should be expanded to include more regulatory factors, in an attempt to explain how genomes organization is regulated. It should also take into account higher order chromatin structure, which may be critical for the understanding of genomic organization as already suggested by the remarkable nuclear lamina interaction data we used here. The Spatial clustering framework we introduced provides an immediate answer to problems with analysis of current data, as well as foundations for the development of more holistic models for genome organization.

## Acknowledgments

This work was supported by an Asher and Jeannette Alhadeff Research Award and by the Israeli science foundation converging technologies program. AT is an Alon fellow.

## References

1. Barski, A., et al.: High-resolution profiling of histone methylations in the human genome. *Cell* 129(4), 823–837 (2007)
2. Li, X.Y., et al.: Transcription factors bind thousands of active and inactive regions in the Drosophila blastoderm. *PLoS Biol.* 6(2), e27 (2008)
3. Meissner, A., et al.: Genome-scale DNA methylation maps of pluripotent and differentiated cells. *Nature* 454(7205), 766–770 (2008)
4. Mikkelsen, T.S., et al.: Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature* 448(7153), 553–560 (2007)
5. Wang, Z., et al.: Combinatorial patterns of histone acetylations and methylations in the human genome. *Nat. Genet.* 40(7), 897–903 (2008)
6. Liu, C.L., et al.: Single-nucleosome mapping of histone modifications in *S. cerevisiae*. *PLoS Biol.* 3(10), e328 (2005)
7. Guenther, M.G., et al.: A chromatin landmark and transcription initiation at most promoters in human cells. *Cell* 130(1), 77–88 (2007)

8. Gal-Yam, E.N., et al.: Frequent switching of Polycomb repressive marks and DNA hypermethylation in the PC3 prostate cancer cell line. *Proc. Natl. Acad. Sci. USA* 105(35), 12979–12984 (2008)
9. Kondo, Y., et al.: Gene silencing in cancer by histone H3 lysine 27 trimethylation independent of promoter DNA methylation. *Nat. Genet.* 40(6), 741–750 (2008)
10. Moving AHEAD with an international human epigenome project. *Nature* 454(7205), 711–715 (2008)
11. Fu, Y., et al.: The insulator binding protein CTCF positions 20 nucleosomes around its binding sites across the human genome. *PLoS Genet.* 4(7), e1000138 (2008)
12. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. *J. Comput. Biol.* 6(3-4), 281–297 (1999)
13. Eisen, M.B., et al.: Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95(25), 14863–14868 (1998)
14. Tanay, A., et al.: Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. USA* 101(9), 2981–2986 (2004)
15. Segal, E., et al.: Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat. Genet.* 34(2), 166–176 (2003)
16. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning. Series in Statistics*, p. 533. Springer, Heidelberg (2001)
17. Kuznetsov, V.A., et al.: Computational analysis and modeling of genome-scale avidity distribution of transcription factor binding sites in chip-pet experiments. *Genome Inform.* 19, 83–94 (2007)
18. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge (1998)
19. Hubbard, T., et al.: The Ensembl genome database project. *Nucleic. Acids Res.* 30(1), 38–41 (2002)
20. Kent, W.J., et al.: The human genome browser at UCSC. *Genome Res.* 12(6), 996–1006 (2002)
21. Guenel, L., et al.: Domain organization of human chromosomes revealed by mapping of nuclear lamina interactions. *Nature* 453(7197), 948–951 (2008)
22. Kim, T.H., et al.: Analysis of the vertebrate insulator protein CTCF-binding sites in the human genome. *Cell* 128(6), 1231–1245 (2007)
23. Schuettengruber, B., et al.: Genome regulation by Polycomb and trithorax proteins. *Cell* 128(4), 735–745 (2007)
24. Vakoc, C.R., et al.: Profile of histone lysine methylation across transcribed mammalian chromatin. *Mol. Cell Biol.* 26(24), 9185–9195 (2006)
25. Ooga, M., et al.: Changes in H3K79 methylation during preimplantation development in mice. *Biol. Reprod.* 78(3), 413–424 (2008)
26. Hon, G., Ren, B., Wang, W.: ChromaSig: a probabilistic approach to finding common chromatin signatures in the human genome. *PLoS Comput. Biol.* 4(10), e1000201 (2008)
27. Boyer, L.A., et al.: Polycomb complexes repress developmental regulators in murine embryonic stem cells. *Nature* 441(7091), 349–353 (2006)
28. Lee, T.I., et al.: Control of developmental regulators by Polycomb in human embryonic stem cells. *Cell* 125(2), 301–313 (2006)
29. Bernstein, B.E., et al.: A bivalent chromatin structure marks key developmental genes in embryonic stem cells. *Cell* 125(2), 315–326 (2006)

# How Many Bootstrap Replicates Are Necessary?

Nicholas D. Pattengale<sup>1</sup>, Masoud Alipour<sup>2</sup>, Olaf R.P. Bininda-Emonds<sup>3</sup>,  
Bernard M.E. Moret<sup>2,4</sup>, and Alexandros Stamatakis<sup>5</sup>

<sup>1</sup> Department of Computer Science,

University of New Mexico, Albuquerque NM, USA

<sup>2</sup> Laboratory for Computational Biology and Bioinformatics,

EPFL (École Polytechnique Fédérale de Lausanne), Switzerland

<sup>3</sup> AG Systematik und Evolutionsbiologie, Institut für Biologie und

Umweltwissenschaften, University of Oldenburg, Germany

<sup>4</sup> Swiss Institute of Bioinformatics, Lausanne, Switzerland

<sup>5</sup> The Exelixis Lab, Department of Computer Science,

Technische Universität München, Germany

`nickp@cs.unm.edu, masoud.alipour@epfl.ch, olaf.bininda@uni-oldenburg.de,`  
`bernard.moret@epfl.ch, stamatak@cs.tum.edu`

<http://icwww.epfl.ch/~stamatak/index-Dateien/Page443.htm>

**Abstract.** Phylogenetic Bootstrapping (BS) is a standard technique for inferring confidence values on phylogenetic trees that is based on reconstructing many trees from minor variations of the input data, trees called replicates. BS is used with all phylogenetic reconstruction approaches, but we focus here on the most popular, Maximum Likelihood (ML). Because ML inference is so computationally demanding, it has proved too expensive to date to assess the impact of the number of replicates used in BS on the quality of the support values. For the same reason, a rather small number (typically 100) of BS replicates are computed in real-world studies. Stamatakis *et al.* recently introduced a BS algorithm that is 1–2 orders of magnitude faster than previous techniques, while yielding qualitatively comparable support values, making an experimental study possible.

In this paper, we propose *stopping criteria*, that is, thresholds computed at runtime to determine when enough replicates have been generated, and report on the first large-scale experimental study to assess the effect of the number of replicates on the quality of support values, including the performance of our proposed criteria. We run our tests on 17 diverse real-world DNA, single-gene as well as multi-gene, datasets, that include between 125 and 2,554 sequences. We find that our stopping criteria typically stop computations after 100–500 replicates (although the most conservative criterion may continue for several thousand replicates) while producing support values that correlate at better than 99.5% with the reference values on the best ML trees. Significantly, we also find that the stopping criteria can recommend very different numbers of replicates for different datasets of comparable sizes.

Our results are thus two-fold: (i) they give the first experimental assessment of the effect of the number of BS replicates on the quality of support values returned through bootstrapping; and (ii) they validate

our proposals for stopping criteria. Practitioners will no longer have to enter a guess nor worry about the quality of support values; moreover, with most counts of replicates in the 100–500 range, robust BS under ML inference becomes computationally practical for most datasets. The complete test suite is available at <http://1cbb.epfl.ch/BS.tar.bz2> and BS with our stopping criteria is included in RAxML 7.1.0.

**Keywords:** Phylogenetic Inference, Maximum Likelihood, Bootstrap, Support Value, Stopping Criterion, Bootstrapping.

## 1 Introduction

Phylogenetic trees are used to represent the evolutionary histories of related organisms (as well, of course, as of any other units subject to evolutionary changes, from protein through genes and genomes to languages and ecologies). Most phylogenetic reconstructions for a collection of organisms take as input DNA or protein sequence alignments. (Others may take encoded morphological characters, although the end result remains a collection of aligned sequences.) These input sequences are placed at the leaves of the putative tree and reconstruction proceeds by searching for an optimal internal branching structure for the tree. Due to the rapid, and rapidly accelerating, growth of sequence data in the last few years, reconstruction of trees with more than 1,000 leaves has become increasingly common, often using sequence data from many genes (so-called multi-gene or phylogenomic alignments). Such practice represents a major departure from the typical practice of the last 20 years, in which trees of 10–100 organisms were inferred from the sequence of a few simple ribosomal genes. Scaling up inference in terms of the number of organisms, the length and complexity of the sequence data, and the diameter (largest pairwise distance among the organisms) is a very challenging issue [19]. The search space (all possible distinct branching structures) is notoriously large ( $(2n - 5)!! = (2n - 5) \cdot (2n - 7) \dots 5 \cdot 3 \cdot 1$ ) and unstructured. Both Maximum Parsimony and Maximum Likelihood approaches are known to be NP-hard, but both are preferred to the simpler distance methods, especially in the presence of more complex data or data with large diameters.

Significant progress has been achieved in the field of heuristic ML search algorithms with programs such as PHYML [12], GARLI [33], LeaPhy [32], and RAxML [29]. However, there is still a major bottleneck in computing bootstrap support (BS) values on these trees, which can require more than one month of sequential execution time for a likely insufficient number of 100 replicates [28] on a reasonably fast CPU. To date, it has proved infeasible to assess empirically the convergence properties of BS values, much less to evaluate means for dynamically deciding when a set of replicates is sufficiently large—at least on the size of trees where computing BS values is an issue.

Recently, Stamatakis *et al.* [30] introduced a fast BS algorithm that yields a run time acceleration of one to two orders of magnitude compared to other current algorithms while returning qualitatively comparable support values. This

improvement makes possible a large-scale experimental study on bootstrap stopping criteria, the results of which are the topic of this paper.

We propose two stopping criteria. Both split the set of replicates computed so far into two equal sets and compute statistics on the two sets. The frequency criterion (FC) is based on the observed frequencies of occurrences of distinct bipartitions; the more conservative weight criterion (WC) computes the consensus tree for each subset and scores their similarity. Both criteria can be computed efficiently and so a stopping test can be run every so many replicates until stopping is indicated. We test these criteria and the general convergence properties of BS values on 17 diverse real-world DNA, single-gene, as well as multi-gene datasets, that include between 125 and 2,554 sequences. We find that our stopping criteria typically stop computations after 100–500 replicates (although the most conservative criterion may continue for several thousand replicates) while producing support values that correlate at better than 99.5% with the reference values on the best ML trees. Unsurprisingly, differences tend to occur mostly on branches with poor support—on branches with support values of at least 0.75, over 98% of the values returned after early stopping agree with the reference values to within 5%.

Our results show that the BS convergence speeds of empirical datasets are highly dataset-dependent, which means that bootstopping criteria can and should be deployed to determine convergence on a per alignment basis. The criteria help to conduct as many BS replicates as *necessary* for a given accuracy level and thus help to reduce the computational costs for phylogenetic analyses. Practitioners will no longer have to enter a guess nor worry about the quality of support values; moreover, with most counts of replicates in the 100–500 range, robust BS under ML inference becomes computationally practical for most datasets.

The remainder of this paper is organized as follows: In Section 2 we review the bootstrap concept and related work on stopping criteria for (mostly non-phylogenetic) bootstrap procedures, including a brief overview of convergence criteria for MrBayes [26]. In Section 3 we describe our family of stopping criteria. In Section 4 we describe our experimental study, give detailed results, and discuss their implications.

## 2 Related Work on Bootstopping Criteria

### 2.1 The Phylogenetic Bootstrap

Phylogenetic bootstrapping is a fairly straightforward application of the standard statistical (nonparametric) bootstrap and was originally suggested by Felsenstein [9] as a way to assign confidence values to edges/clades in phylogenetic trees. Phylogenetic BS proceeds by generating perturbed BS alignments which are assembled by randomly drawing alignment columns from the original input alignment with replacement. The number of columns in the bootstrapped alignment is identical to the number of columns in the original alignment, but the column composition is different. Then, for each BS alignment, a tree is reconstructed independently. The procedure returns a collection of tree *replicates*.

The replicates can then be used either to compute consensus trees of various flavors or to draw confidence values onto a reference tree, e.g., the best-scoring ML tree. Each edge in such a reference tree is then assigned a confidence value equal to the number of replicates in which it appears. The question we address in this paper is—how many replicates must be generated in order to yield accurate confidence values? By accurate confidence values we mean relative accuracy of support values (the “true” support values are unknown for empirical datasets) with respect to support values obtained by a very large number ( $\geq 10,000$  in our experiments) of reference replicates. The extent to which the question about the appropriate number of BS replicates has been answered in other applications of the (non-phylogenetic) bootstrap is the subject of the following subsection.

## 2.2 General Bootstopping Criteria

Most of the literature addressing (whether theoretically or empirically) the issue of ensuring a sufficient number of replicates stems from the area of general statistics or econometrics. However, they are difficult to apply to phylogenetic BS due to the significantly higher computational and theoretical complexity of the estimator [17]. In addition, the problem is more complex since the number of entities (bipartitions) to which support values are assigned grows during the BS procedure, i.e., adding more BS replicates increases the number of unique bipartitions. This is not commonly the case for other application areas of the general Bootstrapping procedure and general bootstopping criteria that have recently been proposed (for instance see [13]).

Standard textbooks on Bootstrapping such as [6,8] suggest to choose a sufficiently large number  $B$  of BS replicates without addressing exact bounds for  $B$ . This does not represent a problem in most cases where the BS procedure is applied to simple statistical measures such as the mean or variance of univariate statistics. Efron and Tibshirani [8] suggest that  $B = 500$  is sufficient for the general standard bootstrap method in most cases. Manly *et al.* [18] propose a simple approach to determine  $B$  *a priori*, i.e., before conducting the BS analysis, based on a worst-case scenario by approximating the standard deviation of BS statistics. The analysis in [18] concludes that a general setting of  $B = 200$  provides a relatively small error margin in BS estimation. This approximation can only be applied to standard BS procedures, based on simple, univariate statistics. However, a larger number of BS replicates is required for other applications of the Bootstrap such as the computation of confidence intervals or tests of significance. P. Hall [14] proposes a general method for stopping the BS in a percentile- $t$  confidence interval. In the area of econometrics, Davidson and MacKinnon [7] propose a two-step procedure to determine  $B$  for BS P-values based on the most powerful test. Andrews *et al.* [11,23] propose and evaluate a general three-step algorithm to specify  $B$  in the bootstrap procedure. Andrews and Buchinsky [4] then further extend their algorithm to bootstrap BCA intervals.

With respect to phylogenetics Hedges [15] suggests a method to specify  $B$  *a priori* for a given level of significance. This approach does not take into account the number of sequences and hence the number of potential alternative

tree topologies, or the number of base-pairs or distinct patterns in the alignment. However, as underlined by our experimental results, important alignment-specific properties such as the “gappyness” (percentage of gaps) of the alignment, the quality of the alignment, and the respective phylogenetic signal strength greatly influence the estimator (the tree search algorithm) and hence the stability of BS replicates. We conclude that an adaptive stopping criterion which is computed on the fly at regular intervals during the actual BS search is best suited to take into account the particularities of real-world datasets and to determine a useful trade-off between accuracy and inference time. We are convinced that such trade-offs will become increasingly important for analysis on large phylogenomic datasets under computational resource constraints, as a current collaborative study with Biologists already requires 2,000,000 CPU hours on an IBM Blue-Gene/L supercomputer. Therefore, we assess our approach empirically, via a large number of computational experiments on diverse real datasets.

### 2.3 Bayesian Convergence Criteria and Tools

There exists some work on convergence criteria and tools for Bayesian phylogenetic analyses, most probably because the convergence of the actual search as opposed to a sufficient number of BS replicates in ML represents a more serious methodological problem for MCMC in general and phylogenetic MCMC searches in particular [20,27,31]. Gelman, Rubin, and Brooks [5,10] provide general frameworks to determine convergence of iterative simulations, with a focus on MCMC methods. MrBayes implements convergence diagnostics for multiple Metropolis-coupled MCMC chains that use the average standard deviation in partition frequency values across independent analyses. One potential drawback is that these statistics take into account all partition frequencies and not only the important, highly supported ones. In addition, there exist tools for graphical exploration of convergence such as AWTY [21] to visualize convergence rates of posterior split probabilities and branch lengths or Tracer [23] that analyzes time-series plots of substitution model parameters. AWTY also offers bivariate plots of split frequencies for trees obtained via independent chains. Note that both AWTY and Tracer require the user to visually inspect the respective output and determine whether the MCMC chains have converged. We are not aware of any computational experiments to assess the performance and accuracy of the above methods.

## 3 Bootstopping Criteria

In this section, we introduce stopping criteria for bootstrapping procedures, which we call “bootstopping” criteria. These are measures that are computed and used at run time, during the replicate generation phase, to decide when enough replicates have been computed. The frequency-based criterion (FC) is based upon Pearson’s correlation coefficient, whereas the Weighted Robinson-Foulds criterion (WC) is based upon the (weighted) symmetric topological difference widely used in phylogenetics.

### 3.1 Terminology and Definitions

A phylogenetic tree  $T$  is an unrooted binary tree; its leaves (also called tips) are labelled by the organism names of the input alignment, while its internal nodes represent hypothetical extinct common ancestors. Removing a branch between nodes  $a$  and  $b$  from a tree  $T$  disconnects the tree and creates two smaller trees,  $T_a$  and  $T_b$ . The trees  $T_a$  and  $T_b$  induce a *bipartition* (or split) of the set  $S$  of taxa (organism names at the leaves) of  $T$  into two disjoint taxon sets  $A$  and  $B$  ( $A \cup B = S$ ). We denote such a bipartition as  $A|B$ . Thus, there exists a one-to-one correspondence between the bipartitions of  $S$  and the branches of  $T$ , so that each tree is uniquely characterized by the set of bipartitions it induces. If  $|S| = n$ , then any (unrooted) multifurcating phylogenetic tree for  $S$  has at most  $2n - 3$  branches and so induces at most  $2n - 3$  bipartitions. If the tree is fully bifurcating the number of bipartitions is exactly  $2n - 3$ , while the number of non-trivial bipartitions, i.e., splits at branches that do not lead to a tip, is  $n - 3$ .

The Robinson-Foulds (RF) metric (sometimes referred to as symmetric difference) is a dissimilarity metric between two trees and counts the number of bipartitions that occur in one tree and not the other. The Weighted Robinson-Foulds (WRF) metric generalizes the RF metric by summing the weights of the bipartitions that contribute to the RF metric (and also, optionally, includes the sum of differences between the weights of shared bipartitions). Finally, consensus methods take a set of trees and return a single 'summary' tree. The majority rule consensus method (MR) returns a tree containing only bipartitions that exist in greater than half the input trees. The extended majority rules method (MRE, also known as *greedy consensus*) uses the MR consensus tree as a starting point and greedily adds bipartitions that occur in less than half the input trees by descending order of their frequency in the hopes (although not always possible) of obtaining a fully bifurcating (binary) tree.

### 3.2 Stopping Criteria

The two criteria we present in the following are both based on the same underlying mechanism. Initially, the set of replicates to be tested for convergence is randomly split into two equal halves. Then we compute statistics between the bipartition support values induced by these halves. If the difference between the splits of the replicates are small this indicates that adding more replicates will not significantly change the bipartition composition of the replicate set. In addition, we compute the statistics not only for one but for 100 random splits of the replicate sets, i.e., we draw a sample from all possible random splits of the replicates by applying a permutation test.

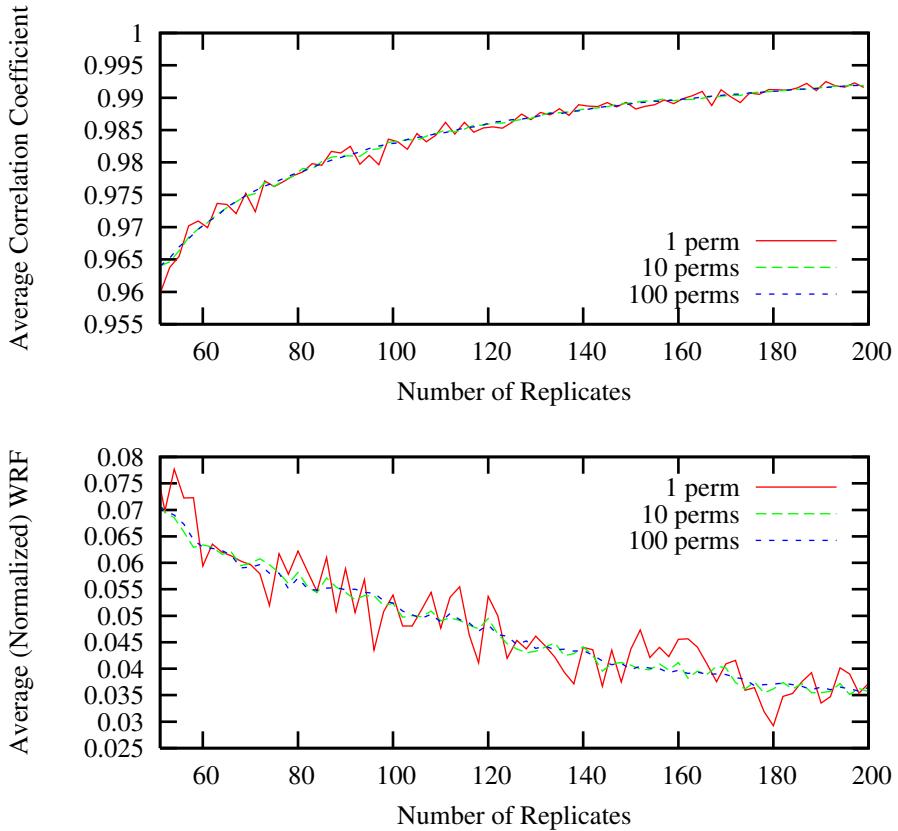
**Frequency Criterion (FC)** The frequency-based criterion uses the bipartition frequencies of all replicates computed up to the point at which the test is conducted, for example every 50 replicates, i.e., at 50, 100, 150, 200, ... replicates. One major design goal is to devise stand-alone criteria that do not rely on a previously computed best-known ML tree for the original alignment. This is partially due to the rapid BS algorithm (and future extensions thereof) in

RAxML that uses information gathered during the BS search to steer and accelerate the search for the best-scoring ML tree on the original alignment. Another important goal is to avoid a heavy dependency on the spacing (e.g., every 10, 20, or 50 replicates) of two successive steps of the test, i.e., we do not want to compute statistics that compare 20 with 30 replicates. Therefore, we have adopted a procedure, that is in some sense similar to the aforementioned convergence tests for MCMC chains implemented in MrBayes. There are two main differences though: (i) we do not use the test to determine convergence of the tree search itself, and (ii) we do not apply the test to only one single random or fixed split of the replicate tree set.

Our FC test works as follows: Assume that the test is conducted every 50 replicates, i.e., after the computation of 50, 100, 150, ... BS replicates. This spacing of 50 has been chosen empirically, in order to achieve a reasonable computational trade-off between the cost of the test and the cost for computing replicates (future work will cover the development of adaptive spacing strategies). The empirical setting also fits the typical range of bootstopped tree topologies, which range between 150 and 450 in our FC-based experiments, depending on the strength of the signal in the respective alignment. For the sake of simplicity, assume that we conduct the test for 50 replicates. At the top level of our procedure we perform a permutation test by randomly splitting up those 50 trees  $p = 100$  times ( $p = 100$  permutations) into disjoint sets  $s_1, s_2$  of equal size with 25 trees each. The advantage of 100 random splits over a single random split or a fixed split into, e.g., replicates with even and odd numbers, is that the curve is smoothed and depends to a far lesser degree on a by chance favorable or unfavorable single split of the data.

In Figure 11 we depict the impact of using  $p = 1, 10$ , and  $100$  permutations on the FC and WC criteria (see Sect. 3.2) for a dataset with 500 sequences. As expected the curve becomes smoother for larger  $p$  settings; a setting of  $p = 10$  appears to be sufficient to smooth the curve and reduce the cost of the test. Though statistically more stable, the disadvantage of this approach is clearly the significantly increased computational cost of the test. Nonetheless, an initial, highly optimized at a technical level, yet algorithmically naïve implementation requires only 1 minute to conduct all 6 tests on 50, 100, ..., 300 replicates on a 1,481 taxon dataset (2 minutes, 40 seconds for  $p = 1000$  random splits), compared to roughly 27 hours for the computation of 300 rapid BS replicates.

For each of the aforementioned 100 random splits we compute the support vectors  $v_1$  for  $s_1$  and  $v_2$  for  $s_2$  for all bipartitions  $b_{ALL}$  found in  $s_1 \cup s_2$ , i.e., all bipartitions contained in the original 50 trees. Note that both vectors  $v_1, v_2$  have length  $b_{ALL}$ . Given those two vectors for each permutation (random split)  $i$ , where,  $i = 0, \dots, 99$  we simply compute Pearson's correlation coefficient  $\rho_i$  on the vectors. Our procedure stops if there are at least 99  $\rho_i$  with  $\rho_i \geq 0.99$  (only one possible parameter setting). We henceforth denote the Pearson's threshold used as  $\rho_{FC}$ . A potential drawback of this method is that the support frequencies on the best-scoring tree or for all bipartitions found during the BS search might not follow a normal distribution. Nonetheless, the FC method appears to work



**Fig. 1.** FC (top) and WC (bottom) criteria for various  $p$  settings on dataset 500

reasonably well in practice (see Section 4). Another potential drawback is that the FC criterion is based on the bipartition frequencies of all bipartitions found. However, from a biological point of view, one is only interested in the “important” bipartitions, i.e., the bipartitions induced by the best-scoring ML tree or the bipartitions that form part of a strict, majority rule, or extended majority rule consensus tree. We address the design of a criterion that only takes into account important bipartitions in the next section. Nonetheless, the FC test can easily be extended in the future to take into account the important bipartitions by providing a user-defined best-scoring ML tree using either Pearson’s correlation or, e.g., the mean square error between corresponding bipartition support values.

**Weighted Robinson-Foulds distance-based Criterion (WC)** The Weighted Robinson-Foulds (WRF) distance criterion (WC) is employed similarly to the FC criterion (i.e. every 50 trees and uses  $p = 100$  permutations per test). Rather than computing a vector correlation, we compute the majority rules consensus trees for

$s_1$  and  $s_2$  and then assess the (dis)similarity between the two consensus trees. We then use the respective consensus trees, which only contain support values for “important” biologically relevant partitions, to calculate the WRF distance between the consensus tree  $c(s_1)$  of tree set  $s_1$  and the consensus tree  $c(s_2)$  of tree set  $s_2$ .

As a distance measure and hence convergence criterion we use the weighted Robinson-Foulds distance (WRF). This weighted topological distance measure between consensus trees takes into account the support values and penalizes incongruent subtrees with low support to a lesser extent. When RF distances are significantly larger than their weighted counterparts (WRF), this indicates that the differences in the consensus trees are induced by subtrees with low support. When  $\text{WRF} \approx \text{RF}$  this means that the differences in the tree topologies under comparison are due to differently placed clades/subtrees with high support. From a biological perspective the WRF distance represents a more reasonable measure since systematists are typically interested in the phylogenetic position of subtrees with high support. In real-world studies the typical empirical threshold is set to 75%, i.e., clades with a BS support of  $\geq 75\%$  are usually considered to be monophyletic (see [28] for a summary). As for the FC criterion, the WC stopping rule can be invoked with varying numbers of permutations and threshold settings. One might for example stop the BS procedure, if for  $p = 99$  out of 100 permutations, the relative WRF between  $c(s_1)$  and  $c(s_2)$  is  $\leq 5\%$ . For reasons of consistency we also denote the threshold parameter for WC as  $\rho_{WC}$ , a  $\rho_{WC}$  setting of 0.97 means that the BS search is stopped when  $p$  WRF distances are  $\leq 1.0 - 0.97 = 3\%$ .

## 4 Experimental Setup and Results

### 4.1 Experimental Setup

To test the performance and accuracy of FC and WC we used 17 real-world DNA alignments containing 125 up to 2,554 sequences. The number of distinct alignment patterns ranges between 348 and 19,436. For the sake of simplicity, alignments will henceforth be referenced by the number of taxa as provided in Table II. The experimental data spans a broad range of mostly hand-aligned sequences including rbcL genes (500, 2,554), mammalian sequences (125, 1,288, 2,308), bacterial and archaeal sequences (714, 994, 1,481, 1,512, 1,604, 2,000), ITS sequences (354), fungal sequences (628, 1,908), and grasses (404). The 10,000 reference BS replicates on each dataset were inferred on two AMD-based Linux clusters with 128 and 144 CPUs, respectively. All result files and datasets used are available for download at <http://1cbb.epfl.ch/BS.tar.bz2>. We make this data available in the hope that it will be useful as a basis for further exploration of stopping criteria as well as general properties of BS.

Computational experiments were conducted as follows. For each dataset we computed a minimum of 10,000 BS replicates using the rapid Bootstrapping (RBS [30]) algorithm implemented in RAxML. We then applied stand-alone bootstopping tests (either FC or WC) that take the set of 10,000 BS reference replicates as input and only execute the tests described in Section 3 without

**Table 1.** Performance analysis of FC ( $p = 99$ ,  $\rho_{FC} = 0.99$ ) vs. WC ( $p = 99$ ,  $\rho_{WC} = 0.97$ ) for three metrics: number of trees to converge, WRF between MRE consensus trees and Correlation Coefficient. Column # Patterns indicates the number of distinct column patterns in each alignment. The last line depicts the respective averages.

DATA	CON-FC	CON-WC	WRF-FC	WRF-WC	P-FC	P-WC	# Patterns
125	150	50	0	0	0.9997	0.9994	19,436
150	250	650	0.03	0.01	0.9984	0.9994	1,130
218	300	550	0.04	0.01	0.9977	0.9988	1,846
354	450	1200	0.03	0.01	0.9979	0.9992	348
404	250	700	0.04	0.01	0.9965	0.9988	7,429
500	200	400	0.03	0.01	0.9982	0.9991	1,193
628	250	450	0.03	0.01	0.9975	0.9987	1,033
714	200	400	0.03	0.02	0.9977	0.9989	1,231
994	150	300	0.04	0.02	0.9964	0.9974	3,363
1,288	200	400	0.03	0.02	0.9967	0.9985	1,132
1,481	300	450	0.04	0.02	0.9968	0.9979	1,241
1,512	250	350	0.03	0.02	0.9977	0.9983	1,576
1,604	250	600	0.04	0.02	0.9975	0.9990	1,275
1,908	200	400	0.03	0.02	0.9975	0.9987	1,209
2,000	300	600	0.03	0.01	0.9976	0.9989	1,251
2,308	150	200	0.03	0.02	0.9980	0.9985	1,184
2,554	200	500	0.03	0.01	0.9975	0.9991	1,232
1,102	238	482	0.03	0.01	0.9976	0.9987	2,771

performing the actual BS search. Returned is a file containing the first  $k$  trees from the full set, where  $k$  is determined by the stopping criterion (FC or WC, along with appropriate parameter values). We refer to these first  $k$  trees as the ‘bootstopped’ trees.

We then computed a number of (dis)similarity metrics between the reference replicates and the bootstopped replicates, including: correlation coefficient, RF between MRE consensus trees of the two sets, and WRF between the MRE consensus trees of the two sets. Additionally, support values from the bootstopped and full replicate sets were drawn on the best-scoring ML tree and the resulting support values compared.

## 4.2 Results for FC and WC Methods

In Table 1 we provide basic performance data for FC and WC. Column *DATA* lists the alignments, *CON-FC* the FC bootstop convergence number, and column *CON-WC* the WC bootstop convergence number. Columns *WRF-FC* and *WRF-WC* provide the WRF distance between the MRE consensus tree for the bootstopped trees and the MRE consensus tree induced by the reference replicates for FC and WC respectively. Finally, columns *P-FC* and *P-WC* provide Pearson’s correlation coefficient between support values from the bootstopped trees and the reference trees on the best-scoring ML tree for FC and WC respectively.

We observe that WC tends to be more conservative, i.e., stops the BS search after more replicates, except for dataset 125. Dataset 125 is a particularly long

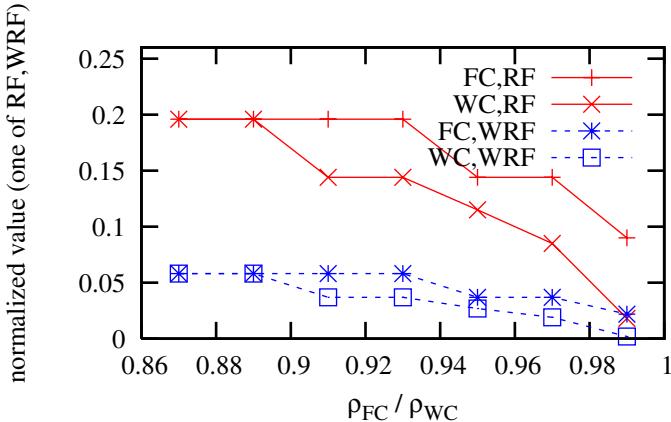
**Table 2.** Performance analysis of FC ( $p = 99$ ,  $\rho_{FC} = 0.99$ ) vs. WC ( $p = 99$ ,  $\rho_{WC} = 0.97$ ) for three metrics: mean error, mean squared error, and loss of support. The last line depicts the respective averages.

DATA	$\mu$ -FC	$\sigma^2$ -FC	$\mu$ -WC	$\sigma^2$ -WC	SUPPLOSS-FC	SUPPLOSS-WC
125	0.303279	0.637530	0.483607	1.807108	0.001066	0.004672
150	1.544218	2.941922	1.074830	1.402564	0.009252	0.003605
218	1.865116	3.205062	1.297674	1.836971	0.005070	0.004674
354	1.364672	1.912598	0.886040	0.864506	0.002009	0.002835
404	2.553616	6.626178	1.384040	2.386179	0.012357	0.007170
500	1.792757	3.532503	1.239437	1.936634	0.010020	0.006841
628	2.030400	4.531876	1.398400	2.175677	0.013400	0.008408
714	2.129395	4.973412	1.424754	2.396237	0.010858	0.008833
994	2.498486	11.178353	2.068618	9.014464	0.013895	0.010575
1,288	2.477821	8.308652	1.700389	3.752257	0.013899	0.009864
1,481	1.845061	5.082219	1.496617	3.243223	0.008562	0.007287
1,512	1.762094	3.958643	1.552684	3.176317	0.008403	0.006289
1,604	1.898813	3.891073	1.229232	1.746953	0.008120	0.005721
1,908	1.961680	4.209030	1.377528	2.298479	0.009711	0.007113
2,000	1.773160	3.323105	1.184276	1.504350	0.008488	0.005020
2,308	1.951410	6.626706	1.703254	4.919317	0.010330	0.009681
2,554	2.063897	4.639194	1.248530	1.793192	0.011319	0.006370
1,102	1.871522	4.681062	1.338230	2.720849	0.009221	0.006762

phylogenomic alignment of mammals and exhibits a surprisingly low variability for the bipartitions it induces. The 10,000 reference replicates only induce a total of 195 distinct bipartitions, which is extremely low given that a single BS tree for this dataset induces  $125 - 3 = 122$  nontrivial bipartitions. The WC method appears to capture this inherent stability of the BS trees sooner than FC, while the WRF to the MRE tree is 0 in both cases, i.e., the consensus trees for 50, 150, and 10,000 replicates are exactly identical. This also underlines our claim that our criteria help avoid needless computation (and needless energy expenditures, as large clusters tend to be power-hungry), in particular on such large and challenging phylogenomic datasets. Due to the general trend for WC to stop later, both WC metrics (P/WRF) are higher than the respective values for FC. For WC, a setting of  $\rho_{WC} = 0.97$  always returns a bootstopped set with a WRF  $< 2\%$  to the MRE consensus of the reference replicates. The results also clearly show that there is a significant alignment-dependent variability in the stopping numbers, as these range between 150 and 450 replicates for FC and between 50 and 1,200 for WC.

In Table 2 we provide additional metrics for the bootstopped trees. Columns  $\mu_x$  and  $\sigma_x^2$  provide the mean error and the mean squared error between support values induced by the  $x = \{\text{FC}, \text{WC}\}$ -bootstopped trees and by the reference trees on the best-scoring ML tree. Columns SUPPLOSS-FC and SUPPLOSS-WC quantify the deviations of support values in the best scoring ML tree.

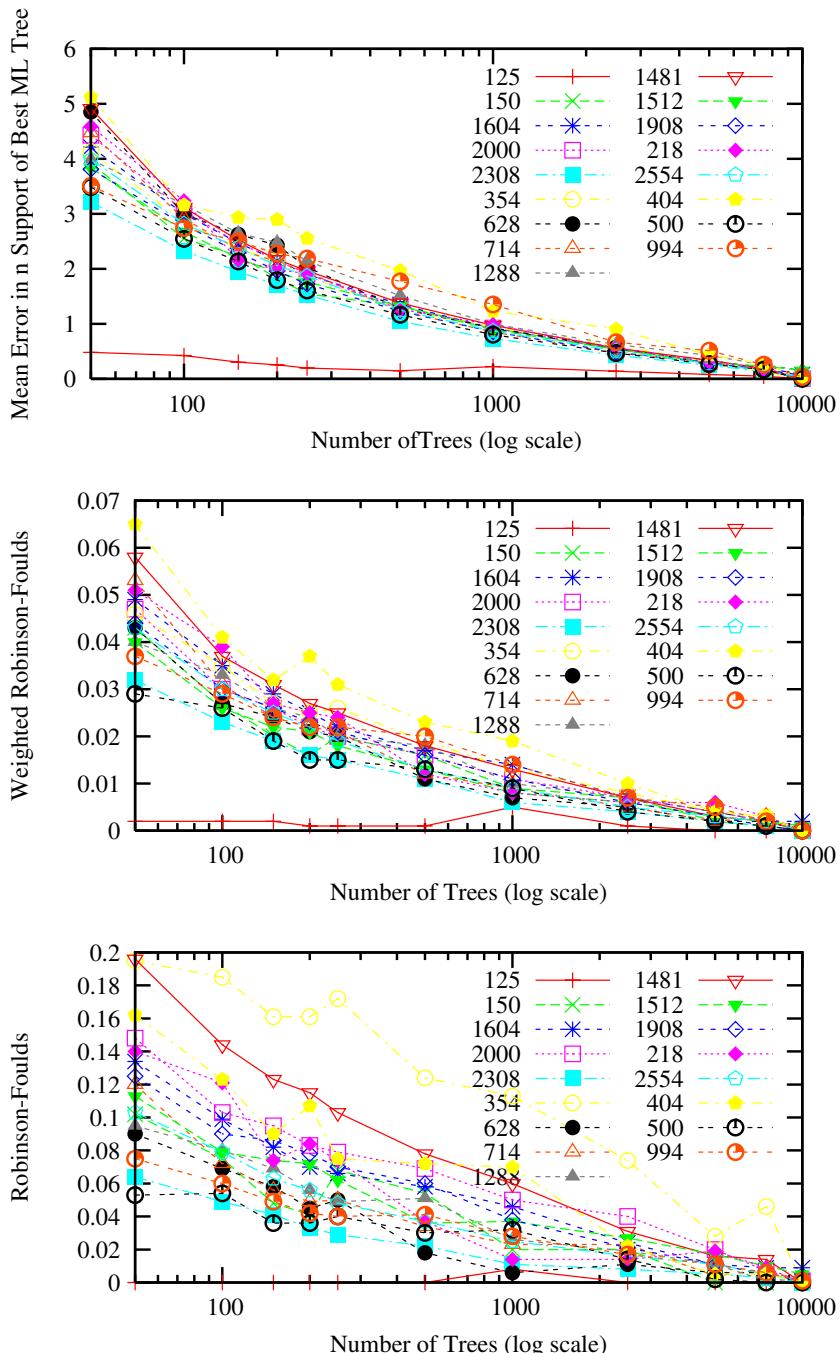
In Figure 2 we graphically depict, for one dataset (1481), the convergence of FC versus WC. We plot the RF and WRF distances between the MRE consensus of the bootstopped trees and reference trees over distinct settings (0.87, 0.88, ..., 0.99)



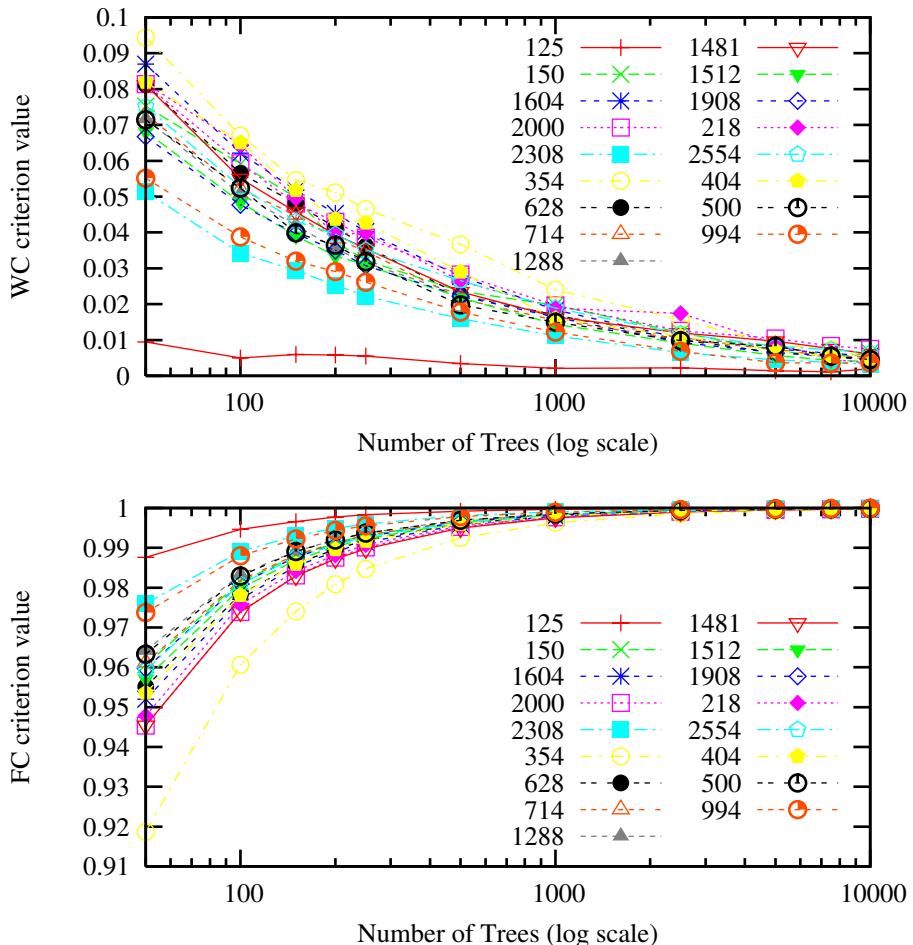
**Fig. 2.** Plot showing convergence of WC over FC for various threshold settings ( $\rho_{FC}$  and  $\rho_{WC}$  respectively) on dataset 1418

for  $\rho_{FC}$  and  $\rho_{WC}$ . For all but two datasets we observed that WC yielded a better convergence (while it required almost 50% more replicates on average) toward replicate sets whose consensi are more congruent (i.e., have lower RF and WRF distances) with the full replicate sets, as a function of  $\rho$ . This favorable property is due to the fact that WC is exclusively based on the “important” bipartitions. Therefore, WC allows to more precisely specify the desired degree of accuracy with respect to the biologically relevant information via an appropriate setting of  $\rho$ . As can be derived from Table II a setting of  $\rho = 0.97$  for WC induces a WRF toward the reference dataset consensus that is  $\leq 2\%$  in all cases for all of our datasets. Hence, the usage of a WC threshold will also be more meaningful, because it appears to be strongly correlated with the final WRF distance to the 10,000 reference replicates.

In addition to assessing our stopping criteria, we have also comprehensively assessed the inherent convergence properties of our replicate sets. Doing so has enabled us to understand a number of quantities that tend to reflect bootstrap support and may help in the design of improved stopping criteria. We have plotted a number of (dis)similarity measures between a subset (i.e., the first  $m$  trees) and full replicate ( $\geq 10,000$  trees) set. In Figure 3 we plot the RF and WRF (in the two lower plots) between the MRE consensus of each tree set restricted to the first  $m$  trees versus its respective full set of replicates ( $\geq 10,000$  trees). This plot shows the differences in convergence speeds among datasets. In addition, it underlines that WRF introduces less noise than RF as replicates are added, so that WRF is a more reliable measure for convergence. An extreme example for this is dataset 354, a short (348 alignment columns) alignment of maple tree sequences from the ITS gene that is known to be hard to analyze [1]. A comparison between the development of RF and WRF over the number of trees for this alignment shows that there are many sequences with low support that are placed in different parts of the tree and essentially reflect unresolved nodes. The slight increase of distance metrics around 1,000 replicates and consecutive decrease observed for dataset 125 might be minor artifacts of the RAxML RBS algorithm.



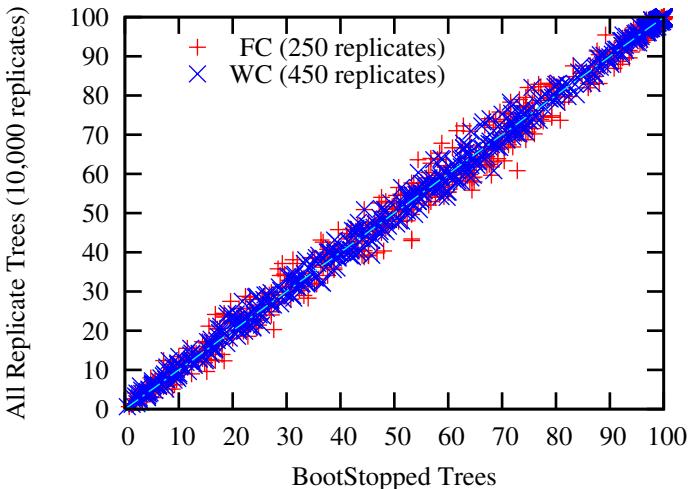
**Fig. 3.** Inherent convergence of replicate sets scored by (top) error in support of best ML tree (middle) WRF and (bottom) RF distances between the first  $m$  trees and the entire (10,000 tree) set



**Fig. 4.** Values of FC and WC criteria for tree subsets consisting of the first  $m$  trees

Also in the upper part of Figure 3 we plot the development of the mean error between support values of  $m$  replicates and all replicates on the best-scoring tree. The three plots in Figure 3 clearly show that the development of WRF distances over the number of replicates is highly congruent to the development of the mean error on the best-scoring tree. Thus, WRF can be used as a criterion to determine convergence without an external reference tree. Designing such a criterion has been a major goal of the phylogenetic community; WRF is the first good answer. Moreover, the plots can help to determine an appropriate threshold setting for  $\rho_{WC}$ , depending on the desired degree of accuracy.

Finally, in Figure 5 we plot the support values of FC/WC-bootstopped trees against the support values from the reference replicates on the best-scoring ML tree for dataset 628. The comparison clearly shows a decrease in deviations from the diagonal for the WC criterion.



**Fig. 5.** Support values drawn on the best ML tree for FC (blue) and WC (red) versus full replicate set, for data set 628

## 5 Conclusion and Future Work

We have conducted the first large-scale empirical ML-based study of the convergence properties of bootstrapping, using biological datasets that cover a wide range of input alignment sizes and a broad variety of organisms and genes. In addition, we have developed and assessed two bootstopping criteria that can be computed at run time and do not rely on externally provided reference trees to determine convergence. The criteria have been designed so as to capture a stopping point that provides sufficient accuracy for an unambiguous biological interpretation of the resulting consensus trees or best-known ML trees with support values. The correlation between bootstopped support values and support values from 10,000 reference trees exceeds 99.5% in all cases, while the relative weighted tree distance (used with the WC criterion) is smaller than the specified threshold value in all cases. We conclude that the WC criterion yields better performance and higher accuracy than FC while it correlates very well with the mean error of support values on the best-scoring tree. We advocate the use of WC over FC because it only takes into account the BS support of “important” bipartitions which are subject to biological interpretation. We have also shown that the number of replicates required to achieve a certain level of accuracy is highly dataset-dependent for real data, so that, by using our criteria, an investigator need only compute as many replicates as necessary, thus avoiding the waste of scarce computational resources, in particular for future large-scale phylogenomic analyses.

Future work entails the full integration of the bootstopping criteria into the forthcoming release of RAxML (RAxML 7.1.0). We will assess whether performance gains can be obtained by applying embedding techniques in the

calculation of RF/WRF[22]. We will also devise ways to dynamically adapt the spacing of FC/WC criteria (which is currently fixed at 50) to the convergence speed of the BS replicates, i.e., use a more sparse spacing for the initial phase and a denser spacing for the later phase of the BS search.

## Acknowledgments

We would like to thank Derrick Zwickl and Bret Larget for useful discussions on this manuscript. We are also thankful to Andrew Rambaut for discussions on Trace and AWTY. We would also like to thank the following colleagues for providing real-world datasets: N. Poulakakis, U. Roshan, M. Gottschling, M. Göker, G. Grimm, C. Robertson, N. Salamin. Part of this work was funded under the auspices of the Emmy Noether program by the German Science Foundation (DFG).

## References

1. Andrews, D.W.K., Buchinsky, M.: On the Number of Bootstrap Repetitions for Bootstrap Standard Errors, Confidence Intervals, and Tests. Cowles Foundation Paper 1141R (1997)
2. Andrews, D.W.K., Buchinsky, M.: A Three-Step Method for Choosing the Number of Bootstrap Repetitions. *Econometrica* 68(1), 23–51 (2000)
3. Andrews, D.W.K., Buchinsky, M.: Evaluation of a Three-step Method for Choosing the Number of Bootstrap Repetitions. *J. of Econometrics* 103(1-2), 345–386 (2001)
4. Andrews, D.W.K., Buchinsky, M.: On The Number of Bootstrap Repetitions for BCa Confidence Intervals. *Econometric Theory* 18(4), 962–984 (2002)
5. Brooks, S.P., Gelman, A.: General Methods for Monitoring Convergence of Iterative Simulations. *J. of Computational and Graphical Statistics* 7(4), 434–455 (1998)
6. Davidson, A.C., Hinkley, D.V.: *Bootstrap Methods and Their Application*. Cambridge University Press, Cambridge (2003)
7. Davidson, R., MacKinnon, J.G.: Bootstrap Tests: How Many Bootstraps? *Econometric Reviews* 19(1), 55–68 (2000)
8. Efron, B., Tibshirani, R.J.: *An Introduction to the Bootstrap*. Chapman and Hall, New York (1993)
9. Felsenstein, J.: Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution* 39(4), 783–791 (1985)
10. Gelman, A., Rubin, D.B.: Inference from Iterative Simulation using Multiple Sequences. *Stat. Sci.* 7, 457–511 (1992)
11. Grimm, G.W., Renner, S.S., Stamatakis, A., Hemleben, V.: A Nuclear Ribosomal DNA Phylogeny of acer Inferred with Maximum Likelihood, Splits Graphs, and Motif Analyses of 606 Sequences. *Evolutionary Bioinformatics Online* 2, 279–294 (2006)
12. Guindon, S., Gascuel, O.: A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Sys. Biol.* 52(5), 696–704 (2003)
13. Guo, W., Peddada, S.: Adaptive Choice of the Number of Bootstrap Samples in Large Scale Multiple Testing. *Stat. Appl. in Genetics and Mol. Biol.* 7(1) (2008)
14. Hall, P.: On the Number of Bootstrap Simulations Required to Construct a Confidence Interval. *The Annals of Statistics* 14(4), 1453–1462 (1986)

15. Hedges, S.B.: The Number of Replications Needed for Accurate Estimation of the Bootstrap P Value in Phylogenetic Studies. *Mol. Biol. Evol.* 9(2), 366–369 (1992)
16. Hillis, D.M., Heath, T.A., John, K.S.: Analysis and Visualization of Tree Space. *Sys. Biol.* 54(3), 471–482 (2005)
17. Holmes, S.: Bootstrapping Phylogenies *Statistical Science*, 18(2), 241–255
18. Manly, B.F.J., et al.: Randomization, Bootstrap and Monte Carlo Methods in Biology. CRC Press, Boca Raton (1997)
19. Moret, B.M.E.: Large-scale Phylogenetic Reconstruction. In: Brown, J.R. (ed.) Comparative Genomics: Basic and Applied Research, pp. 29–48. CRC Press/Taylor & Francis (2007)
20. Mossel, E., Vigoda, E.: Limitations of Markov Chain Monte Carlo Algorithms for Bayesian Inference of Phylogeny. *Ann. Appl. Probab.* 16(4), 2215–2234 (2006)
21. Nylander, J.A.A., Wilgenbusch, J.C., Warren, D.L., Swofford, D.L.: AWTY (are we there yet?): A System for Graphical Exploration of MCMC Convergence in Bayesian Phylogenetics. *Bioinformatics* (2007) (advance access, published August 30)
22. Pattengale, N.D., Gottlieb, E.J., Moret, B.M.E.: Efficiently Computing the Robinson-Foulds Metric. *J. of Computational Biology* 14(6), 724–735 (2007)
23. Rambaut, A., Drummond, A.: Tracer MCMC Trace Analysis Tool version 1.3 (2004)
24. Robinson, D.F., Foulds, L.R.: Comparison of Weighted Labelled Trees. *Lecture Notes in Mathematics* 748, 119–126 (1979)
25. Robinson, D.F., Foulds, L.R.: Comparison of Phylogenetic Trees. *Math. Biosc.* 53(1), 131–147 (1981)
26. Ronquist, F., Huelsenbeck, J.P.: MrBayes 3: Bayesian Phylogenetic Inference under Mixed Models. *Bioinformatics* 19(12), 1572–1574 (2003)
27. Soltis, D.E., Gitzendanner, M.A., Soltis, P.S.: A 567-taxon Data Set for Angiosperms: The Challenges Posed by Bayesian Analyses of Large Data Sets. *Int'l J. Plant Sci.* 168(2), 137–157 (2007)
28. Soltis, D.E., Soltis, P.S.: Applying the Bootstrap in Phylogeny Reconstruction. *Statist. Sci.* 18(2), 256–267 (2003)
29. Stamatakis, A.: RAxML-VI-HPC: Maximum Likelihood-based Phylogenetic Analyses with Thousands of Taxa and Mixed Models. *Bioinformatics* 22(21), 2688–2690 (2006)
30. Stamatakis, A., Hoover, P., Rougemont, J.: A Rapid Bootstrap Algorithm for the RAxML Web Servers. *Sys. Biol.* (2008) (in press)
31. Stamatakis, A., Meier, H., Ludwig, T.: New Fast and Accurate Heuristics for Inference of Large Phylogenetic Trees. In: Proc. of IPDPS 2004, HICOMB Workshop, Proceedings on CD, Santa Fe, New Mexico (2004)
32. Whelan, S.: New Approaches to Phylogenetic Tree Search and Their Application to Large Numbers of Protein Alignments. *Sys. Biol.* 56(5), 727–740 (2007)
33. Zwickl, D.: Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion. PhD thesis, University of Texas at Austin (April 2006)

# A Robust Bayesian Two-Sample Test for Detecting Intervals of Differential Gene Expression in Microarray Time Series

Oliver Stegle<sup>1</sup>, Katherine Denby<sup>2</sup>, David L. Wild<sup>2</sup>,  
Zoubin Ghahramani<sup>1</sup>, and Karsten M. Borgwardt<sup>1,3</sup>

<sup>1</sup> University of Cambridge, UK

<sup>2</sup> University of Warwick, UK

<sup>3</sup> Max-Planck-Institutes for Developmental Biology and Biological Cybernetics,  
Tübingen, Germany  
`os252@cam.ac.uk, karsten.borgwardt@tuebingen.mpg.de`

**Abstract.** Understanding the regulatory mechanisms that are responsible for an organism’s response to environmental changes is an important question in molecular biology. A first and important step towards this goal is to detect genes whose expression levels are affected by altered external conditions. A range of methods to test for differential gene expression, both in static as well as in time-course experiments, have been proposed. While these tests answer the question *whether* a gene is differentially expressed, they do not explicitly address the question *when* a gene is differentially expressed, although this information may provide insights into the course and causal structure of regulatory programs. In this article, we propose a two-sample test for identifying *intervals* of differential gene expression in microarray time series. Our approach is based on Gaussian process regression, can deal with arbitrary numbers of replicates and is robust with respect to outliers. We apply our algorithm to study the response of *Arabidopsis thaliana* genes to an infection by a fungal pathogen using a microarray time series dataset covering 30,336 gene probes at 24 time points. In classification experiments our test compares favorably with existing methods and provides additional insights into time-dependent differential expression.

## 1 Introduction

Understanding regulatory mechanisms, in particular related to the response to changing external conditions, is of great interest in molecular biology. Changes in external conditions include environmental influences or treatments that an organism is exposed to, ranging from parasitic infections studied in plant biology to drug responses that are of interest in pharmacogenomics. A first step towards understanding these mechanisms is to identify genes that are involved in a particular response. This task can be reduced to a decision problem where we want to tell whether a gene is differentially expressed or not. In the past, most available datasets were static, such that this decision was based on measurements of

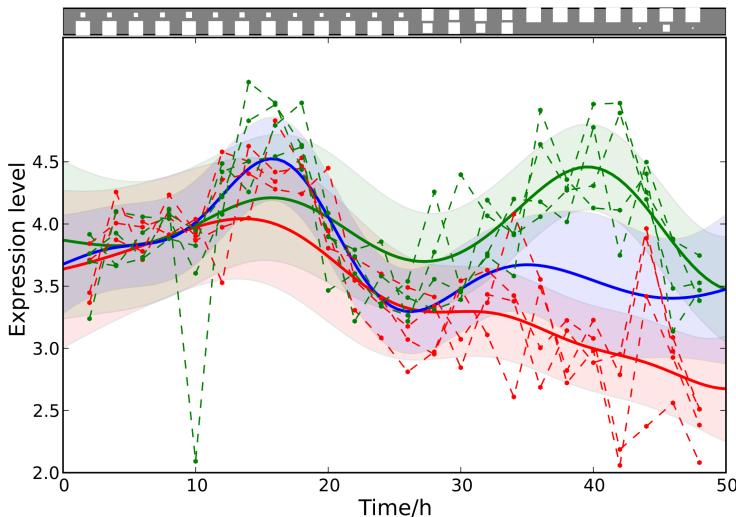
gene expression at a single time point (e.g. (1)). Increasingly, studies are being carried out that measure the expression profiles of large sets of genes over a time course rather than a single static snapshot.

The basic task however remains the same: given an observed time series in two conditions (treatment and control), the goal is to determine whether the observations originate from the same biological process or whether they are better described by means of independent processes specific to each condition. This is referred to as a two-sample problem in statistics. In the bioinformatics and statistics community, a wide range of methods have been proposed to test for differential gene expression, both from static microarray experiments (1; 2; 3; 4; 5) as well as from time series microarray data (6; 7; 8; 9). Among desirable properties of a useful test for time series data are the ability to handle multiple replicates of the same condition and robustness with respect to outliers in measured expression levels. While most tests can be applied to multiple replicates, only few of them are robust to outliers due to non-Gaussian errors (9). Furthermore, existing methods often make strong assumptions on the time series, for instance, gene expression levels being described by a linear model or over a finite basis (7; 8).

In this article, we propose a test for differential gene expression based on Gaussian processes (GP), a nonparametric prior over functions. The GP machinery allows attractive properties of existing two-sample tests to be combined: the capability to handle arbitrary numbers of replicates, robustness to outliers and a flexible model basis. In addition, our method is able to identify patterns of local differential expression, where gene expression levels are only differential in subintervals of the full time series. This feature can be used to understand *when* differential expression occurs. Such information is important in molecular biology, because it provides insights on the temporal order in which genes are activated or inhibited by environmental stimuli. For example, it allows to study whether there is a delay in response, whether the effect of the treatment is only temporary, or to identify a cascade of genes that trigger each other's activation during the response. The detection of *intervals* of differential expression can be considered the second central step towards uncovering gene regulatory mechanisms, which follows the first step of detecting differentially expressed genes. This main contribution is illustrated in Figure 1 (Top), where in addition to a score of differential expression, our test also allows to pinpoint the intervals in which a gene exhibits differential expression, as indicated by the Hinton diagrams in the top panel.

The remainder of this article is organized as follows. In Section 2 we describe our Gaussian process based two-sample test for microarray time series data. In Section 2.2 we show how a heavy-tailed noise model can be incorporated to gain additional robustness with respect to outliers. Section 3 concludes the methodological development by introducing a mixture model that can detect differential expression over parts of the time course. In our experimental evaluation, we compare our model to two state-of-the-art two-sample tests from the literature. On time series data for 30,336 probes from *Arabidopsis thaliana*, we assess the predictive performance (Section 2.4) and demonstrate that the

CATMA3A53880: 50.1201



**Fig. 1.** An example result produced by the GPTwoSample temporal test. **Bottom:** Dashed lines represent replicates of gene expression measurements for control (green) and treatment (red). Thick solid lines represent Gaussian process mean predictions of the latent process traces;  $\pm 2$  standard deviation error bars are indicated by shaded areas. **Top:** Hinton diagrams illustrate the probability of differential expression for different time points. Size of upper bars indicates the probability of the genes being differentially expressed, size of lower bars that of being non-differentially expressed.

detection of differential expression in intervals is useful to gain insights in the response of *Arabidopsis* to a fungal pathogen infection (Section 3.1).

## 2 GPTwoSample - A Robust Two-Sample Test for Time Series Using Gaussian Processes

In line with previous approaches to test for differential expression, our test compares two alternative hypotheses: **Either** the time series measured in two conditions,  $A$  and  $B$ , can be described by a *shared* underlying process ( $f(t)$ ), **or** they are better described by means of two *independent* processes, one for each condition ( $f^A(t)$ ,  $f^B(t)$ ). Figure 2 shows a Bayesian network representation of both hypotheses. We assume that in both conditions, expression levels of  $R$  biological replicates are measured at discrete time points  $t_1, \dots, t_N$ . For notational convenience, we assume that measurements from both conditions and for all replicates are synchronized, i.e. share a common time discretization. However, this is not a requirement of the Gaussian process framework which can deal with arbitrary time representations and missing values.

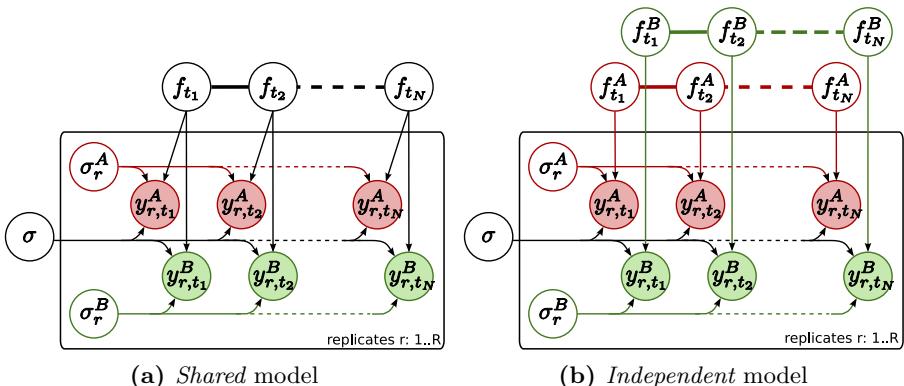
The *Bayes factor* has been previously applied to test for differential expression [9; 10]. Following this idea we score the two alternative hypotheses using the logarithm of the ratio of the corresponding model evidences (i.e. the logarithm of the *Bayes factor*)

$$\text{Score} = \log \frac{P(\mathcal{D}_A | \mathcal{H}_{\text{GP}}, \hat{\theta}_I) P(\mathcal{D}_B | \mathcal{H}_{\text{GP}}, \hat{\theta}_I) P(\hat{\theta}_I)}{P(\mathcal{D}_A, \mathcal{D}_B | \mathcal{H}_{\text{GP}}, \hat{\theta}_S) P(\hat{\theta}_S)}, \quad (1)$$

where  $\mathcal{D}_{A,B}$  represent observed expression levels in both conditions  $A$  and  $B$ . The notation  $\mathcal{H}_{\text{GP}}$  indicates that both models are Gaussian process models, where  $P(\hat{\theta}_I)$  and  $P(\hat{\theta}_S)$  are prior distributions over the model hyperparameters. The *Bayes factor* is computed conditioned on hyperparameters  $\hat{\theta}_I$  and  $\hat{\theta}_S$  of the *independent* (I) and *shared* (S) model respectively. Hyperparameters are set to their most probable value as described in the following.

The accuracy of this score crucially depends on the model used to represent biological processes. A good model should provide sufficient flexibility to allow for noise and variation between biological replicates, but at the same time be able to detect true differential expression. In addition, there are microarray specific requirements. Firstly, observations are likely to be sparse, i.e. only very few observations are made and potentially in irregular intervals. Secondly, expression data is highly susceptible to noise and prone to outliers which can obscure the test result, if not modeled accurately.

All these requirements can be accommodated by a Gaussian process (GP). This nonparametric prior over functions yields the required flexibility while still



**Fig. 2.** Bayesian network for the two alternative models compared in the GP-TwoSample test: **a)** *Shared* model where both conditions are explained by means of a single process  $f(t)$ , **b)** *Independent* model with processes  $f^A(t)$  and  $f^B(t)$  for each condition. Expression levels  $y_{r,t}^{A,B}$  of a given gene are observed in two biological conditions  $A, B$  with  $r : 1, \dots, R$  biological replicates and at discrete time points  $t : t_1, \dots, t_N$ . Observation noise is split into a global noise level  $\sigma$  and a per-replicate noise level  $\sigma_r^{A,B}$ . The smoothness induced by the Gaussian process priors is indicated by the thick band coupling the latent function values at different time points.

allowing specific beliefs, for instance about smoothness and length scales of the process, to be incorporated. Previously, Gaussian processes have been used to model gene expression time dynamics in the context of transcriptional regulation [11] and for biomarker discovery [12]. In the context of hypothesis testing, Gaussian processes have been applied to gene expression profiles by Yuan [10].

## 2.1 Gaussian Process Model

Let us first consider the *shared* model (Figure 2a), where observations from both conditions are described by a single biological process  $f(t)$ . We split up the observation noise into a global noise component and a per-replicate noise component by introducing latent replicate observations  $g_{r,t}^c$ . The joint posterior distribution over unobserved function values  $\mathbf{f}$  and the replicate observations  $g_{r,t}^c$  for conditions  $c \in \{A, B\}$  follows as

$$\begin{aligned} P(\mathbf{f}, \{g_{r,t}^c\} | \mathcal{D}_A, \mathcal{D}_B, \boldsymbol{\theta}_S) &\propto \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \times \\ &\prod_{c=A, B} \prod_{r=1}^R \prod_{n=1}^N \mathcal{N}(g_{r,t_n}^c | f_{t_n}, \sigma_r^c) P_L(y_{r,t_n}^c | g_{r,t_n}^c, \boldsymbol{\theta}_L), \end{aligned} \quad (2)$$

where  $\boldsymbol{\theta}_S = \{\boldsymbol{\theta}_K, \boldsymbol{\theta}_L, \{\sigma_r^c\}\}$  denotes the set of all hyperparameters for kernel, likelihood and the replicate noise levels respectively. The covariance matrix  $K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)$  is derived from the covariance function  $k(t, t' | \boldsymbol{\theta}_K)$  which specifies how function values at two time points  $t$  and  $t'$  covary. We use a covariance function that decays exponentially with squared time distance,  $k_{SE}(t, t') = A \exp\{-\frac{1}{2} \frac{(t-t')^2}{L^2}\}$ , which yields smooth functions with a typical squared amplitude  $A$  and a typical length-scale  $L$ . These kernel hyperparameters are summarized as  $\boldsymbol{\theta}_K$ .

For simplicity, let us first assume Gaussian observation noise with variance  $\sigma$ ,  $P_L(y_{r,t}^c | g_{r,t}^c, \boldsymbol{\theta}_L) = \mathcal{N}(y_{r,t}^c | g_{r,t}^c, \sigma)$ . Integrating out the latent replicate process observations  $g_{r,t}^c$  results in a standard Gaussian process with an effective noise variance per replicate and condition

$$P(\mathbf{f} | \mathcal{D}_A, \mathcal{D}_B, \boldsymbol{\theta}_S) \propto \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{c=A, B} \prod_{r=1}^R \prod_{n=1}^N \mathcal{N}(y_{r,t_n}^c | f_{t_n}, \sigma_r^c), \quad (3)$$

where  $\sigma_r^c = \sqrt{\sigma_r^{c2} + \sigma^2}$ . Predictions from this model can be obtained by considering the joint distribution over training data and an unseen test input  $t_*$ . Completing the square leads to a Gaussian predictive distribution (see [13]) of the corresponding function value  $f_* \sim \mathcal{N}(\mu_*, v_*)$

$$\begin{aligned} \mu_* &= K_{*, \mathbf{T}} [K_{\mathbf{T}, \mathbf{T}} + \Sigma]^{-1} \mathbf{y} \\ v_* &= K_{*, *} - K_{*, \mathbf{T}} [K_{\mathbf{T}, \mathbf{T}} + \Sigma]^{-1} K_{\mathbf{T}, *}, \end{aligned} \quad (4)$$

where  $\Sigma$  is a diagonal matrix constructed from the noise levels  $\{\sigma_r^c\}$  of the observed expression levels. Note that the dependence of the covariance matrices on hyperparameters  $\boldsymbol{\theta}_K$  is omitted for clarity. The *Bayes factor* in Eqn. 11 requires the evaluation of the log marginal likelihood. Again, this quantity can be calculated in closed form

$$\log P(\mathcal{D}_A, \mathcal{D}_B | \mathcal{H}_{GP}, \boldsymbol{\theta}_S) = -\frac{1}{2} \log \det K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K) - \frac{1}{2} \mathbf{y}^\top K_{\mathbf{T}, \mathbf{T}}^{-1} \mathbf{y} - \frac{N}{2} \log 2\pi. \quad (5)$$

The most probable parameter settings  $\hat{\boldsymbol{\theta}}_S$  are determined by finding the maximum of the posterior probability

$$\hat{\boldsymbol{\theta}}_S = \arg \max_{\boldsymbol{\theta}_S} [\log P(\mathcal{D}_A, \mathcal{D}_B | \mathcal{H}_{GP}, \boldsymbol{\theta}_S) + \log P(\boldsymbol{\theta}_S)], \quad (6)$$

where  $P(\boldsymbol{\theta}_S)$  are priors on the hyperparameters. Prior distributions are set to incorporate *a priori* beliefs about parameter values. The prior on the amplitude  $A$  is uninformative and set to a broad gamma distribution  $A \sim \Gamma(0.001, 1000)$ . To ensure that noise is not explained by extremely short length scales, we set the prior on  $L$  such that the expectation value of the gamma prior corresponds to one fifth of the total length of the time series with a standard deviation of 50%. The noise hyperparameters are set to  $\sigma \sim \Gamma(0.1, 10)$  and  $\sigma_r^c \sim \Gamma(0.01, 10)$ , which favors that noise variance is explained by the shared noise variance where possible.

The optimized marginal likelihood of the alternative hypothesis, assuming *independent* biological processes,  $\log P(\mathcal{D}_A | \mathcal{H}_{GP}, \hat{\boldsymbol{\theta}}_I) + \log P(\mathcal{D}_B | \mathcal{H}_{GP}, \hat{\boldsymbol{\theta}}_I) + \log P(\hat{\boldsymbol{\theta}}_I)$ , can be obtained analogously. Hyperparameters of the *independent* model are optimized jointly for both processes  $f^A(t)$  and  $f^B(t)$  where kernel parameters  $\boldsymbol{\theta}_K$  and the global noise variance  $\sigma$  are shared and hence the number of explicit hyperparameters is identical for both models.

## 2.2 Robustness with Respect to Outliers

The presentation of the Gaussian process model so far makes a crucial simplification, namely that observation noise is Gaussian. However, for our full model we use a heavy-tailed noise model to acknowledge that a small fraction of the data points can be extremely noisy (outliers) while others are measured with considerably more precision. To reflect this belief we use a mixture model (14)

$$P_L(y_{r,t}^c | g_{r,t}^c, \boldsymbol{\theta}_L) = \pi_0 \mathcal{N}(y_{r,t}^c | g_{r,t}^c, \sigma) + (1 - \pi_0) \mathcal{N}(y_{r,t}^c | g_{r,t}^c, \sigma_{\text{inf}}), \quad (7)$$

where  $\pi_0$  represents the probability of the datum being a regular observation and  $(1 - \pi_0)$  of being an outlier. The variance of the outlier component  $\sigma_{\text{inf}}$  is much larger than for regular observations and hence allows outliers to be discarded. Unfortunately when using this likelihood model the posterior in Eqn. 2 is no longer computable in closed form. To overcome this problem we use Expectation Propagation (EP) (15), a deterministic approximate inference algorithm. EP approximates the true posterior by a Gaussian process and is efficient enough to

allow the algorithm to be applied on large scale datasets. EP for non-Gaussian likelihoods in Gaussian process models is discussed in (13); robust Gaussian process regression has been previously applied to biological data in (16). The derivation of EP for the robust likelihood and further references can be found in Appendix A.

### 2.3 Runtime

The computational complexity of a Gaussian process models scales with  $(RN)^3$ , where  $N$  is the number of observations per condition and  $R$  the number of replicates. Since microarray time series datasets are typically small in the sense that they cover few time points per gene this is not prohibitive. The robust Gaussian process method requires multiple cycles of EP updates which result in constant factor of additional computation. For the datasets studied below, including 24 time points with 4 replicates, the robust test takes approximately 10 seconds per gene on a standard desktop machine.

### 2.4 Differential Gene Expression in *Arabidopsis Thaliana* after Fungal Infection

We applied GPTwoSample to study plant response to biotic stress on a dataset of microarray time series. Plant stress responses involve a significant degree of transcriptional change, with different stress stimuli activating common signalling components (17).

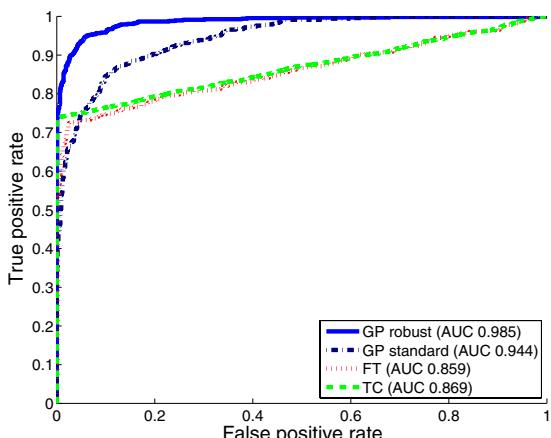
In this particular experiment, the stress response of interest is an infection of *Arabidopsis thaliana* by the fungal pathogen *Botrytis cinerea*. The ultimate goal is to elucidate the gene regulatory networks controlling plant defense against this pathogen. Finding differentially expressed genes and intervals of differential gene expression are important steps towards this goal.

Data were obtained from an experiment in which detached *Arabidopsis* leaves were inoculated with a *B. cinerea* spore suspension (or mock-inoculated) and harvested every 2h up to 48h post-inoculation (i.e. a total of 24 time points). *B. cinerea* spores (suspended in half-strength grape juice) germinate, penetrate the leaf and cause expanding necrotic lesions. Mock-inoculated leaves were treated with droplets of half-strength grape juice. At each time point and for both treatments one leaf was harvested from four plants in identical conditions (i.e. 4 biological replicates). Full genome expression profiles were generated from these whole leaves using CATMA arrays (18). Data preprocessing and normalization was carried out using a pipeline based on the MAANOVA package (19). The experimental design is longitudinal in that subsequent time points should show related expression patterns, but also cross-sectional in that the biological replicates are all from independent plants. Due to this specific study design we expect particularly noisy observations and outliers within the time course of a single replicate plant. For each probe in the dataset, we applied our Gaussian process based test, including the robust noise model (GP robust) to the time courses measured in both conditions and all four replicates. As comparison we also applied

two state-of-the-art methods from the literature, the *timecourse* method (TC) of Tai and Speed [8], and the F-Test (FT) as implemented in the MAANOVA package [19]. For each of the three methods, we rank all probes based on their likelihood of being differentially expressed in descending order.

On a subset of 2000 randomly selected probes we asked a human expert to manually label each probe as either ‘differentially expressed’, ‘not differentially expressed’, or ‘dubious case’. After removing the dubious cases, we used the remaining 1890 labeled probes as gold standard to benchmark the three methods. Figure 3 shows the area under the ROC curve for each method. To check the impact of our outlier-robust model, we also computed the area under the ROC curve for a variant of GPTwoSample that is not robust to outliers and instead uses a standard Gaussian noise model (GP standard). The area under the curve can be interpreted as the probability that a classifier ranks a randomly chosen positive instance (a differentially expressed gene) higher than a randomly chosen negative example (a non-differentially expressed gene). Hence a ‘perfect’ test would reach an AUC of 1, while a consistently failing test would yield an AUC of 0. On this randomly selected set, GPTwoSample with robust noise model (GP robust, AUC 0.986) and the simpler non-robust variant (GP standard, AUC 0.944) outperformed both benchmark models, F-Test (FT, AUC 0.859) and the *timecourse* method (TC, AUC 0.869). The model GP robust achieved an additional improvement over GP standard, showing the merits of a robust noise model.

As a second evaluation, we wanted to get an idea of how the different methods perform on reference datasets of ‘*controlled difficulty*’ (rather than a *random* subset). For this purpose we created reference sets with 400 genes for every method. In each of these sets 100 genes were correctly classified as differentially



**Fig. 3.** Predictive accuracy of four different methods measured by the area under the ROC curve. Each method has been evaluated on the random benchmark dataset of 1890 genes as described in the text.

**Table 1.** AUC scores of three methods on reference datasets created from genome-wide results on GP robust, TC, and FT

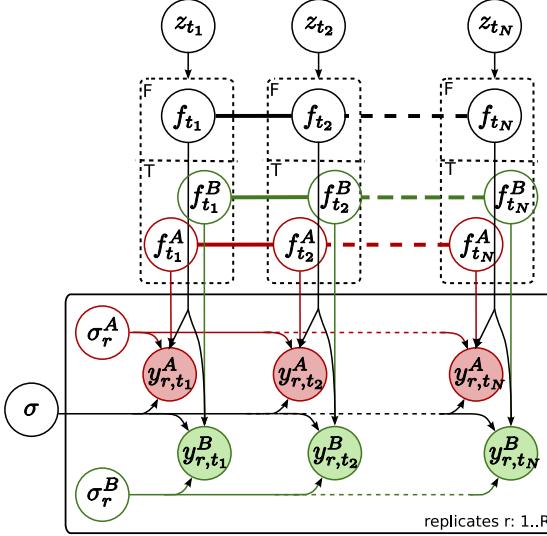
Dataset / Method	GP robust	TC	FT
GP robust dataset	—	0.937	0.929
TC dataset	0.959	—	0.805
FT dataset	0.986	0.956	—

expressed and 100 correctly as non-differentially expressed. The remaining two hundred genes were false positives and false negatives to equal proportions. Following this procedure, we obtained three labeled reference datasets of 400 genes for GP robust, FT and TC. On each of these datasets, we assessed the predictions of the remaining two methods by computing AUC scores (see Table 1). On the *timecourse* reference dataset, our GP robust achieved a higher area under the curve than the F-Test, and it outperformed *timecourse* on the F-Test reference dataset as well. Hence its ability to correctly detect differential gene expression on these reference datasets is again more than competitive with that of the state-of-the-art two-sample tests F-Test [7] and *timecourse* [8].

To further validate the quality of the gene list produced by GPTwoSample we clustered genes considered to be differentially expressed using the SplineCluster method of Heard et al. [20; 21]. We analyzed the resulting clusters for statistically significantly over-represented Gene Ontology(GO) annotations related to a given cluster of genes. The probability that this over-representation is not found by chance can be calculated by the use of a hypergeometric test, implemented in the R/Bioconductor package *GOSTats* [22]. Because of the effects of multiple testing, a subsequent correction of the *p*-values is necessary. We apply a Bonferroni correction, which gives a conservative (and easily calculated) correction for multiple testing. In the supplementary material [23] we show the GO annotations for the clusters which are significant at Bonferroni-corrected *p*-values of 0.01 and 0.05. These GO groupings of the clusters derived from GPTwoSample are intuitively meaningful in the context of plant-pathogen interactions.

### 3 Detecting Intervals of Differential Gene Expression

On knowing that a particular gene is differentially expressed, it is interesting to ask in which time intervals this difference in expression is present and in which time intervals the time series are similar. To tackle this questions we use a mixture model, switching between the two hypotheses, corresponding either to the *shared* model (Figure 2a) or the *independent* model (Figure 2b) as a function of time. Figure 4 shows the Bayesian network representation of this temporal two-sample test. This model is related to mixtures of Gaussian process experts, which have been studied previously [24; 25]. In our setting, we have a fixed number of two experts, where one expert is a single Gaussian process describing both conditions, while the second expert models each condition with a separate process. In order to retain the computational speed required to apply this algorithm on large



**Fig. 4.** Bayesian network for the time local mixture model. At each observed time point  $t_n$  binary indicator variables  $z_{t_n}$  determine whether the observation is explained by the single Gaussian process expert ( $f(t)$ ) or the expert corresponding to the *independent* model ( $f^A(t)$  and  $f^B(t)$ ). This switch is graphically represented as dotted boxes around the processes  $f(t)$  and  $f^A(t)$ ,  $f^B(t)$  respectively. If the switch is true (T) the *independent* expert is used, if the switch is false (F) the *shared* expert.

scale, performing thousands of tests, we use a simplistic gating network. Binary switches  $z_{t_n}$  at every observed time point determine which expert describes the expression level at this particular time point. *A priori* the indicator variables are independent Bernoulli distributed,  $P(z_{t_n}) = \text{Bernoulli}(z_{t_n} | 0.5)$ , assigning both experts equal probability.

The joint probability of both experts and all model parameters, conditioned on the observed data from both conditions, can be written as

$$\begin{aligned}
 P(\mathbf{f}, \mathbf{f}^A, \mathbf{f}^B, \mathbf{Z} | \mathcal{D}_A, \mathcal{D}_B, \boldsymbol{\theta}_S, \boldsymbol{\theta}_I) &\propto P(\mathbf{f} | \boldsymbol{\theta}_K)P(\mathbf{f}^A | \boldsymbol{\theta}_K)P(\mathbf{f}^B | \boldsymbol{\theta}_K) \times \\
 &\prod_{r=1}^R \prod_{n=1}^N [\mathcal{N}(f_{t_n} | y_{r,t_n}^A, \sigma_r^A) \mathcal{N}(f_{t_n} | y_{r,t_n}^B, \sigma_r^B)]^{(z_{t_n}=0)} \times \\
 &[\mathcal{N}(f_{t_n}^A | y_{r,t_n}^A, \sigma_r^A) \mathcal{N}(f_{t_n}^B | y_{r,t_n}^B, \sigma_r^B)]^{(z_{t_n}=1)}, \tag{8}
 \end{aligned}$$

where  $P(\mathbf{f} | \boldsymbol{\theta}_K)P(\mathbf{f}^A | \boldsymbol{\theta}_K)P(\mathbf{f}^B | \boldsymbol{\theta}_K)$  denotes the independent Gaussian process priors on all three processes. Again we simplify the presentation by considering a Gaussian noise model.

Inference in this model is achieved using a variational approximation (26). The joint posterior distribution (Eqn. 8) is approximated by a separable distribution of the form  $Q(\mathbf{f})Q(\mathbf{f}^A)Q(\mathbf{f}^B)\prod_{n=1}^N Q(z_{t_n})$ .

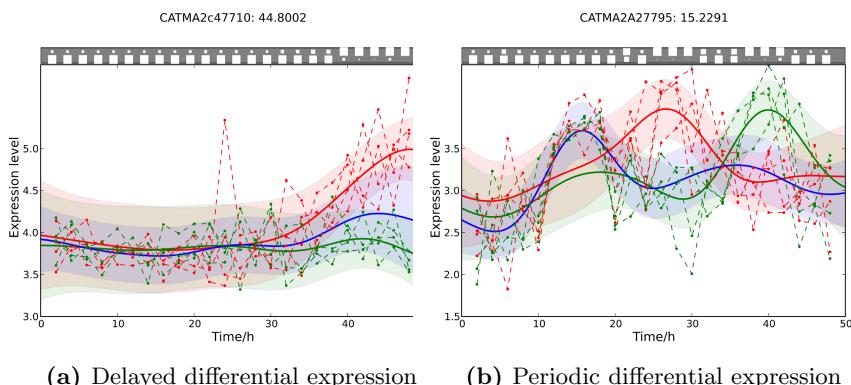
Iterative variational inference updates the approximate posteriors over the latent processes  $Q(\mathbf{f})$ ,  $Q(\mathbf{f}^A)$ ,  $Q(\mathbf{f}^B)$  given the current state of  $Q(\mathbf{Z})$  and vice versa, until convergence is reached. A variational approximation per se is not suited to perform inference in a mixture of Gaussian process model, due to the coupling of target values induced by the GP priors. However, in this specific application, the approximate posteriors over the indicator variables are sufficiently accurate. Finally, to decide whether a time point is differentially expressed, we use the inferred mixing state  $Q(z_{t_n})$  with a threshold value of 0.5.

### 3.1 Detecting Transition Points in the *Arabidopsis* Time Series Data

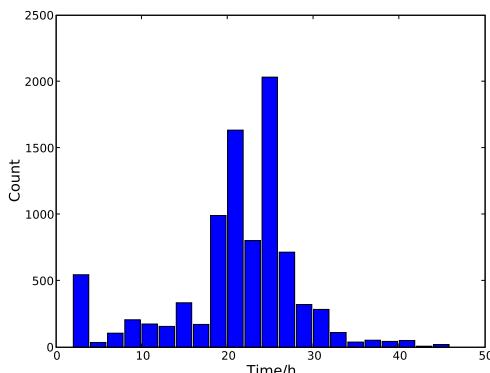
We applied the temporal GPTwoSample model to detect intervals of differential expression of genes from the same *Arabidopsis* time series dataset as in Section 2.4. Figure 5 shows raw data and the inference results for two selected example genes.

**Delayed differential expression.** Having the inferred time intervals of differential and non-differential expression at hand, it is possible to analyze the time information and its distribution over genes.

Here we took the top 9000 genes which have a score suggesting significant differential expression and determined the first time point where the probability of differential expression exceeded 0.5. Figure 6 shows the histogram of



**Fig. 5.** Two example results of the temporal GPTwoSample model on the *Arabidopsis* data. The bottom panel in each plot illustrates the inferred posterior distributions from the Gaussian processes (blue: the process describing the *shared* biological behavior; red and green: the two separate processes modelling differential gene expression). The Hinton diagrams in the top panel indicate whether at a given point in time the gene is likely to be differentially expressed or not. The size of the dots in each row is proportional to the probability of differential expression (top row) and of no differential expression (bottom row).



**Fig. 6.** Histogram of the most likely start of differential expression for the top 9000 differentially expressed genes

this start time. Identification of transition points for individual gene expression profiles shows that a significant change in the transcriptional program begins around 20h post inoculation. This program of gene expression change appears to have two waves peaking around 22h and 26h after inoculation. We expect transcription factors (if regulated by differential expression) to be expressed at earlier time points than the downstream genes whose expression they control. Hence transcription factor genes whose expression first changes in the 22h wave (or earlier) would be of particular interest when designing further experiments to elucidate transcriptional networks mediating the defense response against *B. cinerea*.

## 4 Conclusion

Detecting differential gene expression and patterns of its temporal dynamics are important first steps towards understanding regulatory programs on a molecular level. In this paper, we proposed a Gaussian process framework which provides answers to these problems. Our test not only determines which genes are differentially expressed, but also infers subintervals of differential expression over time. The analysis carried out on the *Arabidopsis thaliana* expression datasets demonstrates that this additional knowledge can be used to gain an understanding of pathways and the timing in which, as in this example, the effect of a fungus infection spreads. Source code and additional information about the used dataset is available online [23].

The natural next question to ask is in which manner these genes interact as part of a regulatory program. The algorithmic task is here to infer a network of regulatory interactions from gene expression measurements and prior knowledge. In future work, we will study how the detection of differential expression can be combined with regulatory network inference.

**Acknowledgments.** The authors would like to thank Andrew Mead and Stuart McHattie for data preprocessing. We acknowledge support from the Cambridge Gates Trust (OS), grants BBSRC BB/F005806/1 (KD and DW), EU Marie Curie IRG 46444 (DW) and NIH GM63208 (KB).

## References

- [1] Kerr, M., Martin, M., Churchill, G.: Analysis of Variance for Gene Expression Microarray Data. *Journal of Computational Biology* 7(6), 819–837 (2000)
- [2] Dudoit, S., Yang, Y.H., Callow, M.J., Speed, T.P.: Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Statistica Sinica* 12, 111–140 (2002)
- [3] Efron, B., Tibshirani, R., Storey, J.D., Tusher, V.: Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association* 96, 1151–1160 (2001)
- [4] Ishwaran, H., Rao, J.: Detecting differentially expressed genes in microarrays using Bayesian model selection. *Journal of the American Statistical Association* 98, 438–455 (2003)
- [5] Lonnstedt, I., Speed, T.: Replicated microarray data. *Statistica Sinica* 12, 31–46 (2002)
- [6] Bar-Joseph, Z., Gerber, G., Simon, I., Gifford, D.K., Jaakkola, T.S.: Comparing the continuous representation of time-series expression profiles to identify differentially expressed genes. *Proceedings of the National Academy of Sciences of the United States of America* 100, 10146–10151 (2003)
- [7] Storey, J.D., Xiao, W., Leek, J.T., Tompkins, R.G., Davis, R.W.: Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America* 102, 12837–12842 (2005)
- [8] Tai, Y.C., Speed, T.P.: A multivariate empirical Bayes statistic for replicated microarray time course data. *Annals of Statistics* 34, 2387–2412 (2006)
- [9] Angelini, C., De Canditiis, D., Mutarelli, M., Pensky, M.: A Bayesian Approach to Estimation and Testing in Time-course Microarray Experiments. *Statistical Applications in Genetics and Molecular Biology* 6 (September 2007)
- [10] Yuan, M.: Flexible temporal expression profile modelling using the Gaussian process. *Computational Statistics and Data Analysis* 51, 1754–1764 (2006)
- [11] Lawrence, N.D., Sanguinetti, G., Rattray, M.: Modelling transcriptional regulation using Gaussian Processes. In: *Advances in Neural Information Processing Systems*, vol. 19, pp. 785–792. MIT Press, Cambridge (2007)
- [12] Chu, W., Ghahramani, Z., Falciani, F., Wild, D.: Biomarker discovery in microarray gene expression data with Gaussian processes. *Bioinformatics* 21(16), 3385–3393 (2005)
- [13] Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge (2006)
- [14] Kuss, M., Pfingsten, T., Csato, L., Rasmussen, C.E.: Approximate Inference for Robust Gaussian Process Regression. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen (2005)
- [15] Minka, T.: Expectation propagation for approximate Bayesian inference. *Uncertainty in Artificial Intelligence* 17, 362–369 (2001)
- [16] Stegle, O., Fallert, S.V., MacKay, D.J.C., Bräge, S.: Gaussian process robust regression for noisy heart rate data. *IEEE Trans. Biomed. Eng.* 55, 2143–2151 (2008)

- [17] Fujita, M., Fujita, Y., Noutoshi, Y., Takahashi, F., Narusaka, Y., Yamaguchi-Shinozaki, K., Shinozaki, K.: Crosstalk between abiotic and biotic stress responses: a current view from the points of convergence in the stress signaling networks. *Current Opinion in Plant Biology* 9, 436–442 (2006)
- [18] Allemeersch, J., Durinck, S., Vanderhaeghen, R., Alard, P., Maes, R., Seeuws, K., Bogaert, T., Coddens, K., Deschouwer, K., Hummelen, P.V., Vuylsteke, M., Moreau, Y., Kwekkeboom, J., Wijfjes, A.H., May, S., Beynon, J., Hilson, P., Kuiper, M.T.: Benchmarking the catma microarray. a novel tool for arabidopsis transcriptome analysis. *Plant Physiol.* 137, 588–601 (2005)
- [19] Wu, H., Kerr, M., Cui, X., Churchill, G.: MAANOVA: a software package for the analysis of spotted cDNA microarray experiments. *The Analysis of Gene Expression Data: Methods and Software*, pp. 313–341
- [20] Heard, N., Holmes, C., Stephens, D., Hand, D., Dimopoulos, G.: Bayesian coclustering of Anopheles gene expression time series: Study of immune defense response to multiple experimental challenges. *Proceedings of the National Academy of Sciences* 102(47), 16939–16944 (2005)
- [21] Heard, N.A., Holmes, C.C., Stephens, D.A.: A Quantitative Study of Gene Regulation Involved in the Immune Response of Anopheline Mosquitoes: An Application of Bayesian Hierarchical Clustering of Curves. *Journal of the American Statistical Association* 101(473), 18 (2006)
- [22] Falcon, S., Gentleman, R.: Using GOstats to test gene lists for GO term association. *Bioinformatics* 23(2), 257 (2007)
- [23] Stegle, O., Denby, K., Wild, D.L., Ghahramani, Z., Borgwardt, K.: Supplementary material: A robust Bayesian two-sample test for detecting intervals of differential gene expression in microarray time series (2009), <http://www.inference.phy.cam.ac.uk/os252/projects/GPTwoSample>
- [24] Yuan, C., Neubauer, C.: Variational Mixture of Gaussian Process Experts. In: *Advances in Neural Information Processing Systems*, vol. 19. MIT Press, Cambridge (2008)
- [25] Rasmussen, C.E., Ghahramani, Z.: Infinite Mixtures of Gaussian Process Experts. In: *Advances in Neural Information Processing Systems*, vol. 19, pp. 881–888. MIT Press, Cambridge (2001)
- [26] Jordan, M., Ghahramani, Z., Jaakkola, T., Saul, L.: An introduction to variational methods for graphical models. *Machine Learning* 37, 183–233 (1999)
- [27] Kullback, S., Leibler, R.: On Information and Sufficiency. *The Annals of Mathematical Statistics* 22(1), 79–86 (1951)
- [28] Seeger, M.: Expectation Propagation for Exponential Families. Technical report, University of California at Berkeley (2005)

## A Expectation Propagation for Robust Gaussian Process Regression

Predictions (Eqn. 4) and the log marginal likelihood (Eqn. 5) are only available in closed form for a Gaussian likelihood model  $P_L$ . When using a complicated likelihood function, such as the mixture model in Eqn. 7, Expectation Propagation (EP) (15) can be used to obtain a tractable approximation.

In our application the exact posterior distribution over latent functions  $f(t)$  for a given dataset  $\mathcal{D} = \{t_n, y_n\}_{n=1}^N$  is

$$\begin{aligned} P(\mathbf{f} | \mathcal{D}, \boldsymbol{\theta}) &\propto \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N P_L(y_n | f_n, \boldsymbol{\theta}_L) \\ &= \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N [\pi_0 \mathcal{N}(y_n | f_n, \sigma) + (1 - \pi_0) \mathcal{N}(y_n | f_n, \sigma_{\text{inf}})], \end{aligned} \quad (9)$$

where again we define  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_K, \boldsymbol{\theta}_L\}$ . The goal of EP is to approximate this exact posterior with a tractable alternative

$$Q(\mathbf{f} | \mathcal{D}, \boldsymbol{\theta}) \propto \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N g_n(f_n), \quad (10)$$

where  $g_n(f_n)$  denote approximate factors. Following (14) we choose unnormalized Gaussians as approximate factors

$$g_n(f_n | C_n, \tilde{\mu}_n, \tilde{\nu}_n) = C_n \exp \left( -\frac{1}{2\tilde{\nu}_n} (f_n - \tilde{\mu}_n)^2 \right), \quad (11)$$

which leads to an approximate posterior distribution of  $f(t)$  that is a Gaussian process again. Evaluated at the training inputs the distribution over function values is a multivariate Gaussian

$$Q(\mathbf{f} | \mathcal{D}, \boldsymbol{\theta}_K, \boldsymbol{\theta}_L) \propto \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N g_n(f_n | C_n, \tilde{\nu}_n, \tilde{\nu}_n) \quad (12)$$

$$= \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \mathcal{N}(\mathbf{f} | \tilde{\boldsymbol{\mu}}, \tilde{\Sigma}), \quad (13)$$

where we define  $\tilde{\boldsymbol{\mu}} = \{\nu_1, \dots, \nu_N\}$  and  $\tilde{\Sigma} = \text{diag}(\{\nu_1^2, \dots, \nu_N^2\})$ .

The idea of EP is to iteratively update one approximate factor leaving all other factors fixed. This is achieved by minimizing the Kullback–Leibler (KL) divergence, a distance measure for distributions (27). Updates for a single approximate factor  $i$  can be derived by minimizing

$$\begin{aligned} \text{KL} \left[ \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n \neq i} q_n(f_n | C_n, \tilde{\mu}_n, \tilde{\nu}_n) \overbrace{P_L(y_i | f_i, \boldsymbol{\theta}_L)}^{\text{exact factor}} || \right. \\ \left. \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n \neq i} q_n(f_n | C_n, \tilde{\mu}_n, \tilde{\nu}_n) \underbrace{g_i(f_i | C_i, \tilde{\mu}_i, \tilde{\nu}_i)}_{\text{approximation}} \right] \quad (14) \end{aligned}$$

with respect to the  $i$ th factor's parameters  $\tilde{\mu}_i, \tilde{\nu}_i$  and  $C_i$ . This is done by matching the moments between the two arguments of the KL divergence which can then be translated back into an update for factor parameters. It is convenient

to work in the natural parameter representation of the distributions where multiplication and division of factors are equivalent to addition and subtraction of the parameters.

There is no convergence guarantee for EP, but in practice it is found to converge for the likelihood model we consider [14]. The fact that the mixture of Gaussians likelihood is not log-concave is problematic, as it may cause invalid EP updates, leading to a covariance matrix that is not positive definite. We avoid this problem by damping the updates [14, 28].

After EP converged, we obtain a Gaussian process as approximate posterior distribution again and hence can evaluate a predicted mean and variance as for the Gaussian noise model (Eqn. 1).

By capturing the zeroth moment of the exact distribution with the explicit normalization constant  $C_n$ , we obtain an approximation to the log marginal likelihood

$$\begin{aligned} \log P(\mathcal{D} | \boldsymbol{\theta}_K, \boldsymbol{\theta}_L) &= \ln \int d\mathbf{f} \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N P_L(f_n | y_n, \boldsymbol{\theta}_L) \\ &\approx \ln \int d\mathbf{f} \mathcal{N}(\mathbf{f} | 0, K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K)) \prod_{n=1}^N g_n(f_n | C_n, \tilde{\mu}_n, \tilde{\nu}_n) \end{aligned} \quad (15)$$

$$\begin{aligned} &= \frac{1}{2} \sum_{n=1}^N (\ln \tilde{\nu}_n^2 + \ln C_n) - \frac{1}{2} \ln |K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K) + \tilde{\Sigma}| \\ &\quad - \frac{1}{2} \tilde{\mu}^\top (K_{\mathbf{T}, \mathbf{T}}(\boldsymbol{\theta}_K) + \tilde{\Sigma}) \tilde{\mu}. \end{aligned} \quad (16)$$

This log marginal likelihood approximation enables us to optimize hyperparameters of the kernel  $\boldsymbol{\theta}_K$ , as well as the from likelihood  $\boldsymbol{\theta}_L$  and serves as approximation when evaluating the *Bayes factor* in Eqn. 11.

# Incorporating Nucleosomes into Thermodynamic Models of Transcription Regulation

Tali Raveh-Sadka\*, Michal Levo\*, and Eran Segal\*\*

Computer Science and Applied Mathematics Department, Weizmann Institute of Science,  
Rehovot, 76100, Israel  
[eran.segal@weizmann.ac.il](mailto:eran.segal@weizmann.ac.il)

Transcriptional control is central to many cellular processes and consequently, much effort has been devoted to understanding its underlying mechanisms. Recently, it has become evident that the organization of nucleosomes along promoter regions has an important role in transcriptional control, since most transcription factors cannot bind to sequences bound by nucleosomes, and thus compete with nucleosomes for DNA access. This competition is governed by the relative concentrations of nucleosomes and transcription factors and by their respective sequence binding preferences. Even though competition of nucleosomes and transcription factors may have significant effects on transcription, a mechanistic understanding of its quantitative consequences for gene expression is still missing. Here we employ a thermodynamic framework based on fundamental principles of statistical mechanics to theoretically explore the effect that different nucleosome organizations along promoters have on the activation dynamics of promoters in response to varying concentrations of the regulating transcription factors. We show that even simple landscapes of nucleosome organization reproduce experimental results regarding the effect of nucleosomes as general repressors and as generators of obligate binding cooperativity between transcription factors. Our modeling framework also allows us to characterize the effects that various sequence elements of promoters will have on the induction threshold and on the shape of the promoter activation curves.

---

\* These authors contributed equally to this work.

\*\* Correspondence Author.

# Combinatorial Algorithms for Structural Variation Detection in High Throughput Sequenced Genomes

Fereydoun Hormozdiari<sup>1,\*\*</sup>, Can Alkan<sup>2,\*\*</sup>, Evan E. Eichler<sup>2,\*</sup>,  
and S. Cenk Sahinalp<sup>1,\*</sup>

<sup>1</sup> Lab for Computational Biology, Simon Fraser University, Burnaby, BC, Canada

<sup>2</sup> Dept. of Genome Sciences, University of Washington, and Howard Hughes Medical Institute,  
Seattle, WA, USA

eee@gs.washington.edu, cenk@cs.sfu.ca

**Motivation.** Recent studies show that, along with single nucleotide polymorphisms and small indels, larger structural variants among human individuals are common. These studies have typically been based high-cost library generation and Sanger sequencing; however, recent introduction of next-generation sequencing (NGS) technologies is changing how research in this area is conducted in a significant way. High-throughput sequencing technologies such as 454, Illumina, Helicos, and AB SOLiD produce shorter reads than the traditional capillary sequencing, yet they reduce the cost (and/or the redundancy) by a factor of 10 – 100 and perhaps even more. Those NGS technologies with the capability of sequencing paired-ends (or *matepairs*) of a clone insert (which follows a tight length distribution) have made it feasible to perform detailed and comprehensive genome variation and rearrangement studies. Unfortunately, the few existing algorithms for identifying structural variation among individuals using paired-end reads have not been designed to handle the short read lengths and the errors implied by these platforms. Here, we describe, for the first time, algorithms for identifying various forms of structural variation between a paired-end NGS sequenced genome and a reference genome.

**Methods.** We present two alternative formulations and corresponding algorithms for the structural variation detection problem between a set of paired-end short reads from an individual genome and a reference genome. Both methods initially map each paired-end read to all potential locations that are within a user defined edit distance to the paired-end read. The first formulation then aims to find the most parsimonious set of structural variants that can be associated with these potential mappings. This problem turns out to be *NP*-hard; however, we provide an approximation algorithm (with  $O(\log n)$  approximation factor) based on the classical Set-Cover method from the combinatorial algorithms literature. We call this method *VariationHunter-Set-Cover* (or in short *VariationHunter-SC*). The second formulation on the other hand, aims to compute the probability of each "potential" structural variant, given the probability of each potential map location (for every paired-end read) and vice versa. We describe an iterative

---

\* Corresponding Authors.

\*\* These authors contributed equally.

method (similar to EM methods) to find a solution for this problem, which we call *VariationHunter-Probabilistic* (or in short *VariationHunter-Pr*).

**Results.** We have applied our algorithms to identify structural variation (deletions, insertions, and inversions) between the genome of a human individual recently sequenced with the Illumina Genome Analyzer platform and the reference genome. We show that our method significantly outperforms "naive" methods that initially map each paired-end read to a *single "best"* location, and then greedily cluster such mappings that support a structural variant. The details of this paper can be found in the full version that will appear in the Genome Research Special Issue for RECOMB'09.

# Optimizing PCR Assays for DNA Based Cancer Diagnostics

Ali Bashir<sup>1</sup>, Qing Lu<sup>1</sup>, Dennis Carson<sup>1</sup>, Benjamin Raphael<sup>2</sup>, Yu-Tseng Liu<sup>1</sup>, and Vineet Bafna<sup>1</sup>

<sup>1</sup> University of California, San Diego, La Jolla, CA 92093, USA

{abashir, vbafnna}@ucsd.edu

<sup>2</sup> Brown University, Providence, RI 02912, USA

braphael@brown.edu

**Abstract.** Somatically acquired DNA rearrangements are characteristic of many cancers. The use of these mutations as diagnostic markers is challenging, because tumor cells are frequently admixed with normal cells, particularly in early stage tumor samples, and thus the samples contain a high background of normal DNA. Detection is further confounded by the fact that the rearrangement boundaries are not conserved across individuals, and might vary over hundreds of kilobases. Here, we present an algorithm for designing PCR primers and oligonucleotide probes to assay for these variant rearrangements. Specifically, the primers and probes tile the entire genomic region surrounding a rearrangement, so as to amplify the mutant DNA over a wide range of possible breakpoints and robustly assay for the amplified signal on an array. Our solution involves the design of a complex combinatorial optimization problem, and also includes a novel alternating multiplexing strategy that makes efficient detection possible. Simulations show that we can achieve near-optimal detection in many different cases, even when the regions are highly non-symmetric. Additionally, we prove that the suggested multiplexing strategy is optimal in breakpoint detection.

We applied our technique to create a custom design to assay for genomic lesions in several cancer cell-lines associated with a disruption in the *CDKN2A* locus. The *CDKN2A* deletion has highly variable boundaries across many cancers. We successfully detect the breakpoint in all cell-lines, even when the region has undergone multiple rearrangements. These results point to the development of a successful protocol for early diagnosis and monitoring of cancer.

## 1 Introduction

Cancers are characterized by somatically acquired DNA mutations. While these include point mutations, they often involve rearrangements of large genomic regions [1]. Recurrent events such as “gene-fusion” were originally characterized in leukemias, and uncommon solid tumors. This changed with the discovery of recurrent gene fusions in prostate cancers [2]. This has motivated multiple sequencing surveys on primary tumors and tumor cell lines using high-throughput paired-end mapping (PEM) and conventional BAC End Sequence Profiling (ESP) [3][4]. In parallel, computational techniques have been developed to analyze PEM/ESP data to detect gene disruptions and

gene fusions that impact tumor development [5]. These technological advancements will be key to developing new treatments and diagnostic tests. Nevertheless, there are substantial barriers to developing diagnostic tests based on identification of a characteristic somatic rearrangement.

Primarily, the aforementioned studies require relatively pure DNA samples. In the early stage of tumor development, the DNA samples contain a high background of normal DNA (below 1:100, tumor:normal). This makes a sequence based approach impractical; the depth of sequence coverage required to reliably detect a rearrangement would be enormous. PCR is used to amplify targeted genomic regions. Previous studies have focused on optimizing primer multiplexing strategies to target multiple genomic regions simultaneously [67]. PCR can similarly be used to amplify a rearranged region, if the rearrangement boundaries are known. However, many rearrangements have highly variable boundaries across patients, often hundreds of kb in size, which means that naive PCR approaches would require a custom assay for each patient. Precisely to address these issues of heterogeneity and detection of variable genomic lesions in a high background of normal DNA, some of us designed a novel multiplex PCR based assay, *Primer Approximation Multiplex PCR (PAMP)* [8]. We have shown success in optimizing primer designs that can detect variable genomic lesions in a highly heterogeneous background of normal DNA [9]. The basic approach is outlined here for deletions, but is applicable to any rearrangement.

**PAMP overview:** See Figure 1. The genomic region of interest is tiled by forward (denoted by  $p_r$ ) and reverse ( $p_\gamma$ ) primers. All of the primers are incorporated into a multiplex tube (or tubes), along with the query DNA. The primers are spaced so that deletion at any boundary will bring a primer pair close enough to be amplified. In Figure 1, the deletion at  $(x, y)$  brings  $p_{1r}$  and  $p_{\gamma 3}$  close together. The amplified product is hybridized to a set of probes denoted by locations  $b_{1r}, b_{2r}, \dots$ , and detected on an array. Successful hybridization confirms the deletion event, and also resolves the deletion boundary to within 1kb.

In spite of its apparent simplicity, successful deployment of PAMP requires the solving of a challenging combinatorial optimization problem relating to the design of primers. Intuitively, the goal is to *maximize the detection of any possible deletion while minimizing the number of multiplex reactions such that no primer-primer interactions occur within a reaction*. The optimization is critical, as each missed deletion is a potentially undetected tumor in an individual. Likewise, we cannot overstate the need for computation, as the design can involve the selection of thousands of primers from tens of thousands of candidate primers, and even a single primer-primer interaction within a multiplex reaction nullifies all signals [9].

**Primer design for PAMP:** Formally, a *primer-design* is described by a set of forward and reverse primers

$$\mathcal{P} = (\mathcal{P}_r, \mathcal{P}_\gamma) = \{(p_{n^r}, \dots, p_{2^r}, p_{1^r}), (p_{\gamma 1}, p_{\gamma 2}, \dots, p_{\gamma \nu})\}$$

The genomic locations of the primers are denoted by

$$l_{n^r} < \dots < l_{1^r} < l_{\gamma 1} < \dots < l_{\gamma \nu}$$

Consider a design,  $\mathcal{P} = (\mathcal{P}_r, \mathcal{P}_\eta)$ . Define a *breakpoint* as a pair of genomic coordinates  $(x, y)$  that come together because of a mutation. Breakpoint  $(x, y)$  is *amplifiable* by a primer pair  $(p_{i^r}, p_{\eta_j})$  if  $(x - l_{i^r}) + (l_{\eta_j} - y) \leq d$ , where  $d$  ( $\approx 2000$ ) is a known parameter constrained by limitations of PCR. For each breakpoint  $(x, y)$ , denote its *coverage-cost*,  $C^{\mathcal{P}}(x, y) = 1$  if it is not amplifiable by a pair in  $\mathcal{P}$ .  $C^{\mathcal{P}}(x, y) = 0$  otherwise. This allows us to define *two-sided* cost of tiling a genomic region by primers as  $\sum_x \sum_y C^{\mathcal{P}}(x, y)$ . The critical constraint in selecting primers is *dimerization*: no pair of primers in a single multiplex reaction must hybridize to each other. One way to solve the dimerization problem is simply to put each pair of primers in their own multiplex tube. With 500 forward, and 500 reverse primers in a typical design, this is not tenable. We consider the following:

### The PAMP design problem

**Input:** Positive integer  $m$ , and a forward and reverse candidate region, with a collection of corresponding forward and reverse primers.

**Output:** A primer design over  $m$  multiplex tubes, so that no primer pair dimerizes in a multiplex reaction, and the coverage cost is minimized ( $\sum_x \sum_y C^{\mathcal{P}}(x, y)$ ).

We will add additional constraints to this formulation in the next section. We previously addressed this problem using a heuristic approach with a simplified coverage function which performed naive multiplexing (see [9]), hereafter termed PAMP-1D. PAMP-1D optimizes a one-sided variant of cost by penalizing each pair of adjacent primers if they were greater than half of the amplifiable distance. The advantage is computational expediency, as the change in coverage due to primer addition/deletion is easily computed. Even though PAMP-1D was able to handle a larger number (tens of thousands) of input primers and converge in a reasonable amount of time (minutes to hours), its coverage function underestimated true breakpoint coverage. Especially for non-symmetric regions, the designs are greatly improved by allowing longer primer gaps on the larger side, and vice-versa. The two-sided problem is very difficult to optimize; previous approaches were impractical for most data-sets only handling  $< 100$  input primers [10]. In our application, even simple input sets would entail thousands of potential primers.

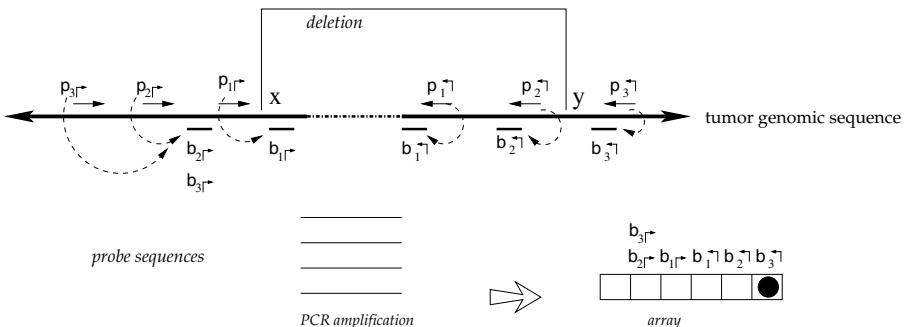
Finally, initial experimental data showed fundamental flaws in these approaches (see Results). The initial formulation makes the reasonable, but *incorrect* assumption that an amplified PCR product can be readily detected by a corresponding probe. Second, it does not carefully account for other interactions, such as ‘mispriming’ (described later), which may also lead to false negatives.

In this manuscript, we seek to redress these shortcomings with PAMP-2D. In Section 2.1, we describe the flaw in the PAMP-1D design and solve it using a novel alternating multiplexing scheme. We follow this by describing a generic framework for optimal multiplexed primer design. The actual optimization is performed using simulated annealing. In Section 2.2, we describe a data-structure to speed-up the simulated annealing iteration. Simulations on multiple genomic region demonstrate the efficacy and efficiency of the proposed approach (Section 3). To test the overall methodology, we assayed events in 3 cell-lines, CEM, A549, and MOLT4. We show that in each case, we can detect the lesion of interest, even in regions with multiple rearrangements.

## 2 Methods: A Multiplexed Approach to PAMP Design

### 2.1 Amplification $\neq$ Detection

We begin by noting that amplification of a deleted genomic region is *not* synonymous with detection of the region. Figure 1 shows an example of this. Each primer is associated with a proximal downstream probe on the array, in order to detect the amplified product. Note the probe and the primer locations cannot match, because residual primers in the solution will hybridize to the array, giving a false signal. This leads to an ‘orphan’ region between the primer and the probe. As an example, the point  $x$  in Figure 1 lies in between primer  $p_{1r}$  and its probe  $b_{1r}$ . When the region  $(x, y)$  is deleted,  $p_{1r}$  and  $p_{3r}$  come together, and are amplified. However, the amplified region does not contain sequence complementary to any probe on the left hand side. Consequently, there is no signal corresponding to the left breakpoint. The solution to this *detection* problem



**Fig. 1. Schematic of PAMP detection failure.** The breakpoint,  $(x, y)$  results in an amplified product, but is not detected on the left side.

lies in recognizing that had the product  $(p_{2r}, p_{3r})$  been amplified, the left breakpoint would be detected by hybridization to probe  $b_{2r}$ . However, even when  $(p_{2r}, p_{3r})$  is close enough to be amplified, it is out-competed by  $(p_{1r}, p_{3r})$  and will not amplify. One possible solution is to add  $p_{1r}$  and  $p_{2r}$  in different *multiplex* tubes. Multiple multiplex reactions is a practical necessity in most cases, as it is challenging to amplify products with a larger number primers in a single tube [11]. Indeed, the experimental design for PAMP, as first proposed by Liu and Carson [8], consists of selecting groups of adjacent primers upstream and downstream of a putative breakpoint, in which each set is one half the desired multiplex size. Nevertheless, increased multiplexing increases the complexity and cost of the experiment, and must be controlled. We address these issues through a novel *Alternating multiplexing* strategy.

**Alternating multiplexing for primer design:** Consider a primer design  $\mathcal{P} = (\mathcal{P}_r, \mathcal{P}_\gamma)$ . Each forward primer  $p_{ir}$  is associated with a downstream probe located at  $b_{ir}$  ( $l_{ir} < b_{ir}$ ). We abuse notation slightly by using  $b_{ir}$  to denote both the probe, and its location. Also  $b_{ir} \leq b_{(i-1)r}$  for all  $i$ , with equality indicating that  $p_{ir}$  and  $p_{(i-1)r}$  share the same

probe. In this notation, probes  $b_{i^r}$  and  $b_{j^r}$  are *adjacent* (on the genome) if there exists  $i \geq k > j$  such that

$$b_{i^r} = b_{(i-1)^r} = \dots b_{k^r} < b_{(k-1)^r} = \dots = b_{j^r}$$

As an example, probes  $b_{3^r}$  and  $b_{1^r}$  are adjacent in Figure 1 as also  $b_{2^r}$  and  $b_{1^r}$ , but not  $b_{3^r}$  and  $b_{2^r}$ . Analogous definitions apply for reverse primers. The probes are located at  $b_{\gamma_1} \leq b_{\gamma_2} \leq \dots b_{\gamma_\nu}$  where  $b_{\gamma_i} < l_{\gamma_i}$ , for all  $i$ .

**Definition 1.** *Breakpoint*  $(x, y)$  is left-detectable (respectively, right-detectable) by a primer-design  $\mathcal{P}$  if it is amplifiable by some primer pair  $(p_{i^r}, p_{\gamma_j})$ , and  $l_{i^r} < b_{i^r} < x$ , (respectively,  $l_{\gamma_j} > b_{\gamma_j} > y$ ).

The set of *left-detectable* and *right-detectable* breakpoints is denoted by

$$\mathcal{S}_X(\mathcal{P}) = \{(x, y) | (x, y) \text{ is left-detectable by } \mathcal{P}\},$$

$$\mathcal{S}_Y(\mathcal{P}) = \{(x, y) | (x, y) \text{ is right-detectable by } \mathcal{P}\}$$

As a breakpoint might not be detected even when it is detectable, we define

**Definition 2.** *Breakpoint*  $(x, y)$  is left-detected (respectively, right-detected) by  $\mathcal{P}$  if it is left-detectable (respectively, right detectable), and the following are satisfied: (a) no pair of primers  $p, p' \in \mathcal{P}$  dimerize (non-dimerization); and (b)  $(x, y)$  is not amplified by any  $p_{i^r}, p_{\gamma_j} \in \mathcal{P}$  with  $l_{i^r} < x < b_{i^r}$ , or  $l_{\gamma_j} > y > b_{\gamma_j}$  (non-detection).

When  $\mathcal{P}$  is used in a single reaction, we denote the set of left-detected (respectively, right-detected) breakpoints as  $\mathcal{S}_X^*(\mathcal{P}) \subseteq \mathcal{S}_X(\mathcal{P})$  (respectively,  $\mathcal{S}_Y^*(\mathcal{P}) \subseteq \mathcal{S}_Y(\mathcal{P})$ ). By partitioning  $\mathcal{P}$  into multiple multiplexing tubes it may be possible to obtain better coverage. Simply, one could run each forward and reverse primer in its own multiplex reaction so that

$$\cup_{i,j} \mathcal{S}_X^*(p_{i^r}, p_{\gamma_j}) = \mathcal{S}_X(\mathcal{P})$$

The idea behind alternate multiplexing is simple: Primers  $p_{i^r}, p_{j^r}$  (correspondingly,  $p_{\gamma_i}, p_{\gamma_j}$ ) can be added to the same multiplex set only if their downstream probes  $b_{i^r}$  and  $b_{j^r}$  (correspondingly,  $b_{\gamma_i}, b_{\gamma_j}$ ) are not adjacent. A similar rule applies for reverse primers. Formally, first order the all unique probes by their genomic position (independently for the left and right sides), and then number them in increasing order. Partition  $\mathcal{P}_r$  into  $\mathcal{P}_r^0, \mathcal{P}_r^1$ , where  $\mathcal{P}_r^0$  contains all primers whose probe is “even” numbered, and the sets  $\mathcal{P}_r^1$  contains all primers whose probe is “odd” numbered. Similarly, define  $\mathcal{P}_{\gamma}^0, \mathcal{P}_{\gamma}^1$ . The multiplexing reactions are given by choosing each of the 4 forward reverse partitions. In Figure 1 this scheme would place  $p_{2^r}, p_{3^r}$  in different multiplex sets. Before we show that alternating multiplexing optimizes detection, we define a technical term. A design of forward (likewise, reverse) primers is *non-trivial* if there exists at least one pair of primers  $p_{i^r}, p_{j^r}$  with  $0 < l_{i^r} - l_{j^r} < d$ , and  $b_{i^r} \neq b_{j^r}$ . The definition is introduced for technical reasons only, as any useful design must be non-trivial.

**Theorem 1.** Let  $\mathcal{P}$  be a design with no dimerizing pairs. Then, alternating multiplexing allows us to detect all detectable breakpoints. In other words

$$\cup_{a,b \in \{0,1\}} \mathcal{S}_X^*(\mathcal{P}_r^a \times \mathcal{P}_\gamma^b) = \mathcal{S}_X(\mathcal{P}), \quad \text{and} \quad \cup_{a,b \in \{0,1\}} \mathcal{S}_Y^*(\mathcal{P}_r^a \times \mathcal{P}_\gamma^b) = \mathcal{S}_Y(\mathcal{P})$$

Further, if  $\mathcal{P}_r, \mathcal{P}_\gamma$  are non-trivial then the multiplexing is the minimal necessary to achieve detectability.

*Proof.* See Appendix.

Note that Theorem 1 works only for non-dimerizing sets. In earlier work, we have shown that the achievable coverage diminishes with larger sets of primers because of dimerizations. To connect dimerization, detection, and multiplexing, we start by defining a *primer-dimer-adjacency graph*  $G$ , whose nodes are defined by the set of primers. A primer-pair is edge-connected if either of the following is true: a) the primers dimerize, or b) they are adjacent and in the same orientation. Theorem 2 is based on the property that a coloring of this graph partitions primers into non-dimerizing, alternatingly multiplexed sets.

**Theorem 2.** Let each forward (reverse) primer in  $\mathcal{P}$  be edge connected with at most  $\Delta_r$  ( $\Delta_\gamma$ ) other forward (reverse) primers. Further, there is no dimerization between a forward-reverse pair. Then, there exists a design with no more than  $\Delta_r \cdot \Delta_\gamma$  multiplex reactions for which all breakpoints in  $\mathcal{S}_X(\mathcal{P})$  (respectively,  $\mathcal{S}_Y(\mathcal{P})$ ) are left-detected (respectively, right detected).

*Proof.* Proof based on Brooks' theorem [12]. See Appendix.

Theorems 1 and 2 guide an overall iterative strategy that provides for the best coverage, given a bound on the number of multiplex reactions. Assume for now that we have a procedure *OptCoverage*( $G, \Delta_r, \Delta_\gamma$ ) that takes  $G$  and numbers  $\Delta_r, \Delta_\gamma$  as input, and returns an optimized design,  $(\mathcal{P}_r, \mathcal{P}_\gamma)$ . Specifically, it returns the sub-graph induced by  $(\mathcal{P}_r, \mathcal{P}_\gamma)$  in which a) each forward (respectively, reverse) primer is edge-connected to  $\leq \Delta_r$  ( $\leq \Delta_\gamma$ ) forward (reverse) primers; and, b) no forward-reverse primers are edge-connected. We can use Theorems 1, 2 to obtain the same coverage using  $\leq \Delta_r \cdot \Delta_\gamma$  multiplexing. The following section will describe the *OptCoverage* procedure. The overall algorithm is motivated by the fact Theorem 2 only provides a weak upper bound of  $\Delta_r \Delta_\gamma$  on the available multiplexing. If the actual number of multiplex reactions is smaller than available, we adjust and iterate.

### Procedure PrimerDesign( $m, G, L_r, L_\gamma$ )

(\*  $L_r, L_\gamma$  represent lengths of the two regions \*)

1. Let  $\Delta_r = \sqrt{\left(m \frac{L_\gamma}{L_r}\right)}, \Delta_\gamma = \sqrt{\left(m \frac{L_r}{L_\gamma}\right)}$  (\* Initial estimate \*)

2.  $(\mathcal{P}_r, \mathcal{P}_\gamma) = \text{OptCoverage}(G, \Delta_r, \Delta_\gamma)$ .

(\* Return sub-graph induced by  $(\mathcal{P}_r, \mathcal{P}_\gamma)$  with  $\Delta_r$  ( $\Delta_\gamma$ ) edges per forward (reverse) primer and optimal coverage. \*)

3. Compute  $\Delta_c = \text{MaxAdjacency}(\mathcal{P}_r, \mathcal{P}_\gamma)$

4. Use Welsh and Powell algorithm [13] to color  $\mathcal{P}_r$  (and  $\mathcal{P}_\gamma$ ). Return  $m_r$  (and,  $m_\gamma$ ) colors.

5. If  $|m - m_r \cdot m_\gamma|$  is large, adjust  $\Delta_r, \Delta_\gamma$ ; Go to Step 2.

Figure 3c shows that a large gap often exists between  $\Delta_r \Delta_\gamma$  and true multiplexing levels, especially as  $\Delta_r \Delta_\gamma$  gets large. In the following section, we describe the use of simulated annealing to optimize the design of primers.

## 2.2 Simulated Annealing for Optimization

The computational goal is to choose a design  $\mathcal{P}$  that minimizes coverage-cost  $\mathcal{C}^{\mathcal{P}}$ . The optimal design is chosen from candidate solutions  $\mathcal{P}$  in which each forward primer (reverse primer) is edge-connected with at most  $\Delta_r$  ( $\Delta_\gamma$ ) other primers. We use an established simulated annealing approach to perform the optimization [14].

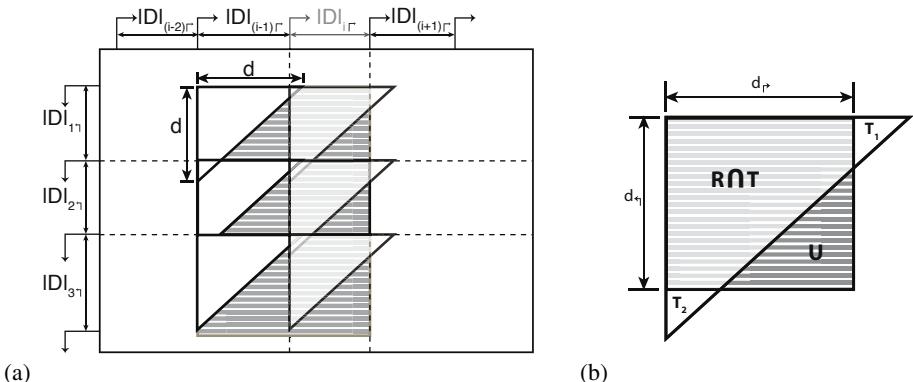
For any candidate solution  $\mathcal{P}$ , define its neighborhood as the set of candidate solutions that are obtained by adding or removing a primer from  $\mathcal{P}$ . In other words,  $\mathcal{P}' \in N_{\mathcal{P}}$  iff  $|\mathcal{P} - \mathcal{P}'| \leq 1$ . Let  $\delta = \mathcal{C}^{\mathcal{P}'} - \mathcal{C}^{\mathcal{P}}$  denote the change in coverage cost in moving from  $\mathcal{P}$  to  $\mathcal{P}'$ . Following the s.a. paradigm, we move to  $\mathcal{P}'$  if  $\delta < 0$ . If  $\delta \geq 0$ , we choose to move to  $\mathcal{P}'$  with probability  $\exp(-\frac{\delta}{T})$ , in order to escape local minima. While the basic paradigm has been explored in our earlier work, and elsewhere, we extend this here by addressing two key issues: 1) Incorporation of probe distances/interactions into the optimization; and, 2) Rapid calculation of the 2D coverage-cost.

**Incorporating Probes:** We address the first problem by noting the direct relationship between primers and probes. Specifically, we do not need to separately iterate over primers and probes; anytime a primer is added we attempt to add its proximal downstream probe. This probe is added unless it is already present in the set (in which case no additional probe is added) or it causes a mispriming signal (in which case the next most proximal probe is examined). As the first condition is trivial, we focus our attention on the second.

There have previously been rigorous attempts at identifying such mispriming pairs in a computational framework [7]. The mispriming problem in PAMP is somewhat unique; it is only problematic when it leads to a “false positive” signal. These signals occur when a primer pair anneals to another region of the genome *and* the amplified sequenced hybridizes to a *probe* on the array. This allows us to create a novel formulation for the probe/mispriming problem. Define a collection of *probe-misprime-triads*  $T_m$  on  $\mathcal{P}$  as follows:  $(p_r, p_\gamma, b) \in T_m$  if and only if primers  $p_r$  and  $p_\gamma$  misprime to genomic sequence  $s$  *and* probe  $b$  anneals to  $s$ . Checking this could be costly if many such mispriming triads exist. In practice, enforcing this criteria has no measurable effect on time complexity as most probes and primer-pairs are unique and  $|T_m|$  is small. This effect is, in fact, minimized prior to simulated annealing optimization by preferentially selecting probes in the input set which do not create probe-misprime-triads.

**Updating Coverage:** For exposition, we focus on coverage (detection will naturally follow). Recall that a breakpoint  $(x, y)$  is covered if there exists some pair of primers  $(p_{i^r}, p_{\gamma j})$  such that  $(x - l_{i^r}) + (l_{\gamma j} - y) \leq d$ . All such breakpoints,  $(x, y)$ , can be considered as points in a two-dimensional space.

Consider a step in the simulated annealing when primer  $p_{i^r}$  is being added, and we need to compute the change in cost. See Figure 2a. We only need consider breakpoints  $(x, y)$ , where  $l_i < x \leq l_{i+1}$ . To check if  $(x, y)$  was previously covered, we only need



**Fig. 2. Schematic of uncovered breakpoint computation.** a) Diagram of a small primer set in which there are initially 4 forward and 4 reverse primers. An additional forward primer (lightly shaded) is added at position  $i$ , which reduces the uncovered space. The difference in uncovered space between  $p_{(i-1)r}$  to  $p_ir$  is seen as the difference in total shaded area compared to darkly shaded area. b) shows uncovered space for the added primer at a specific reverse pair location, given by the dotted lines in a). Uncovered breakpoints are the coordinates contained in the rectangle not contained in the triangle. The total number of such breakpoints is given by,  $|U| = |R - R \cap T| = |R| - (|T| - |T_1| - |T_2|)$ .

to examine the most proximal upstream primer. This suggests a naive algorithm that scales with the length of the opposing region,  $L$ , and the amplifiable range of a PCR product,  $d$ , yielding a time complexity of  $O(Ld)$ .

To make the computation more efficient, we partition the space into forward intervals  $D_{ir} = (l_i, l_{i+1}]$ , and reverse intervals, using adjacent pairs of forward and reverse primers. In Figure 2a these intervals correspond to regions on the  $x$  and  $y$  axes respectively. In adding primer  $p_{ir}$ , coverage is changed only in the forward interval  $D_{ir}$ . The algorithm proceeds by examining rectangles  $R_{ij} = D_{ir} \times D_{j\gamma}$ , one for each reverse interval  $D_{j\gamma}$ . Denote the set of *uncovered* breakpoints in an arbitrary rectangle,  $R$ , as  $U$  (ignoring subscripts  $i$  and  $j$ ), as in Figure 2b. Let  $T$  denote the total space covered by the corresponding primer pair. Observe that

$$U = R - (R \cap T) = R - (T - T_1 - T_2)$$

where  $T_1, T_2$  represents portions of  $T$  not in  $R$  (Note that  $T_1$  and  $T_2$  can be equal to  $\emptyset$ ). Let  $d_r = \min(|D_r|, d)$ , and  $d_\gamma = \min(|D_\gamma|, d)$ . Then,

$$|T_1| = \frac{1}{2}(d - d_r)^2, \quad |T_2| = \frac{1}{2}(d - d_\gamma)^2$$

This leads to a simple equation for calculating the amount of uncovered space  $|U|$ , as

$$|U| = |D_r| |D_\gamma| - \left( \frac{1}{2}d^2 - \frac{1}{2}(d - d_r)^2 - \frac{1}{2}(d - d_\gamma)^2 \right)$$

This update reduces the time complexity several orders of magnitude to  $O(n)$ , where  $n$  is the total number of opposing primers.

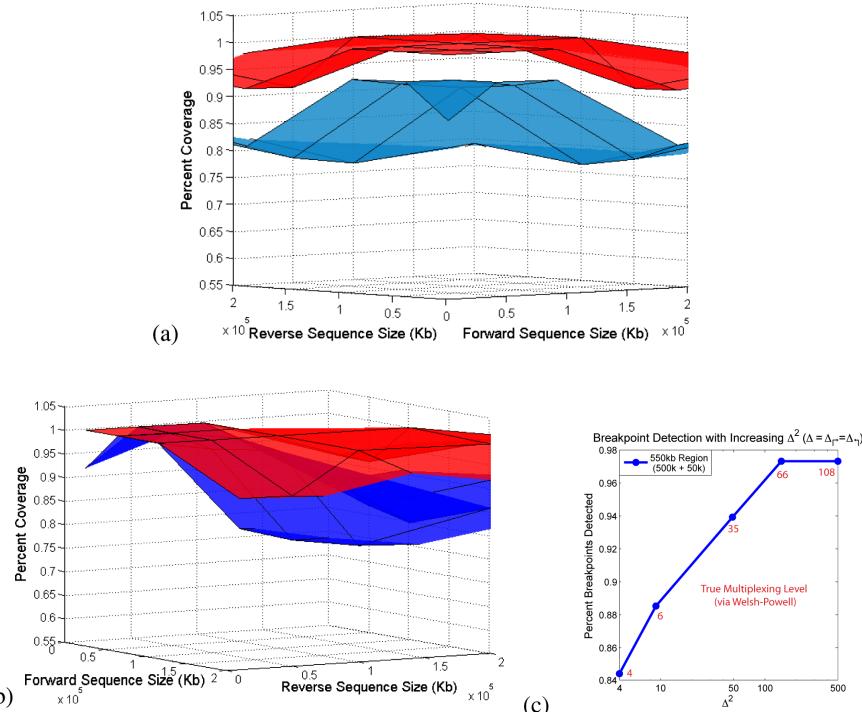
Even so, the update remains expensive to compute, and must be improved further. In adding forward primer  $p_{i^*}$ , the set of values  $d_{\gamma_j}$  do not change. If for some  $d_{\gamma_j}$ ,  $d_{i^*} + d_{\gamma_j} < d$ , then  $D_{i^*} \times D_{\gamma_j}$  is entirely covered. To exploit this, we store all  $d_{\gamma_j}$  in a MAX-HEAP $_{\gamma}$ , and all  $d_{i^*}$  in MAX-HEAP $_{\gamma}$ . When considering a forward primer, we scan MAX-HEAP $_{\gamma}$  using a BFS to get all  $d_{\gamma_j}$  for which  $d_{\gamma_j} \geq d - d_{i^*}$ , for a total of  $O(k)$  steps. If we add the forward primer, we need to make  $O(1)$  updates to MAX-HEAP $_{\gamma}$ . Total time is  $O(k) + O(\lg n)$  per iteration. In most cases, there is either very little uncovered space or the uncovered space is relegated to a few specific regions (as in Figure 4), implying  $k \ll n$ . When optimizing for coverage and detection, we need to maintain two additional heaps with primer-probe distances, with a slightly more complex algorithm. However, the update time remains the same.

### 3 Results

We simulated data-sets from two genomic regions that have been implicated in cancers. A homozygous deletion in the CDKN2A region (9p21) is an important genetic marker for multiple cancers. The lesion has been observed in multiple cell-lines and primary tumor samples, including glioblastomas, leukemias, and lung, pancreatic and breast cancers [15]. However, the boundaries of the deletion are known to vary over a large region. Recently, the deletion was assayed and confirmed in 25/54 (46%) of adolescent ALL patients [16]. These results are based on array-CGH, which would not detect small deletions, or be useful for early diagnosis when the tumor cells are rare compared to wild-type ones. The observed deletions varied in size from 25kb all the way to the loss of an entire arm (52Mb), with a variety of intermediate sized deletions. The overlap among all specimens was 12.5kb. Thus the CDKN2A region is a prototypical case for PAMP. The second example comes from recently mapped deletions in the TMPRSS2 region (21q22.3), that fuse the 5' UTR of the TMPRSS2 gene with ETS transcription factors (ERG, ETV1, or ETV4), resulting in over-expression of these genes, and progression of prostate cancer [17][18]. The two regions are also good test cases in that the CDKN2A fusing regions are symmetric, while the TMPRSS2 is non-symmetric, and not as amenable to a one-sided cost function.

#### 3.1 Simulations

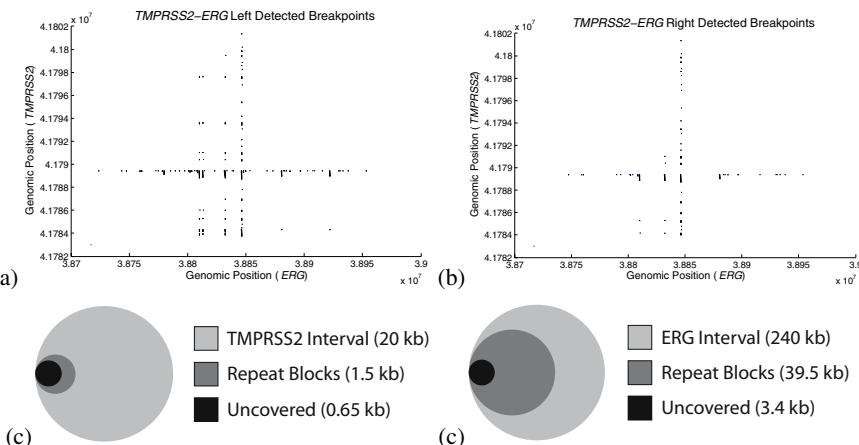
The detectability achieved varies considerably according to the genomic regions in question. Therefore, we bench-marked the overall performance of PAMP-2D across a spectrum of sequence sizes (10, 50, 100, 200kb) for both the forward and reverse primer regions. For each pair of sizes, 10 corresponding pairs of genomic regions were randomly selected, making 160 unique input sets. PAMP-1D, and PAMP-2D were run on each of these sets. Figure 3a shows that PAMP-2D is superior to PAMP-1D over all input sample sizes. Much of the improvement is in detection due to the use of alternating multiplexing. However, the performance remains superior to PAMP-1D even when



**Fig.3. Performance of PAMP-2D .** (a) The surface plot shows that a significant benefit in detectable coverage is seen when comparing PAMP-1D (blue) to PAMP-2D (red). (b) Applying the alternating strategy to PAMP-1D significantly improves its coverage. However PAMP-2D consistently obtains better coverage (especially in non-symmetric regions). (c) As allowed multiplexing,  $\Delta^2$ , in the final primer set increases, the resulting coverage increases. Red values represent the ‘true’ number of multiplex reactions at each data point (as predicted via Welsh-Powell algorithm).

alternating multiplexing is incorporated in PAMP-1D , particularly in non-symmetric regions (Figure 3b, raw results available in online supplement).

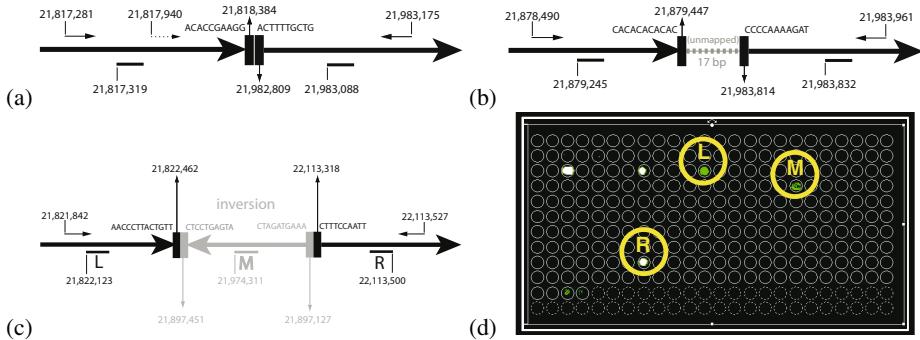
The performance of all methods degrades for large regions ( $\geq 500\text{kb}$ ) due to increased dimerizations. To improve detection for these large regions, increased multiplexing is important. Figure 3c shows the improvement observed in transition to an increasing number of multiplexing sets (represented by  $\Delta_r \cdot \Delta_\gamma$ ) for a non-symmetric  $500 \times 50\text{kb}$  region. Saturation occurs prior to reaching complete coverage, partially because in some regions it is simply not feasible to add in primers and probes. In this region, we also performed a specific optimization for 50 multiplex sets using the aforementioned **PrimerDesign** procedure. A symmetric strategy ( $\Delta_r = 7, \Delta_\gamma = 7$ ) provides only 94% coverage (Figure 3c). The non-symmetric initial solution ( $\Delta_r = 22, \Delta_\gamma = 2$ ) provided 97% coverage with a “true” multiplexing level  $m = 38$ . Iterating with adjusted values, we achieved 98% coverage with  $m = 50$  ( $\Delta_r = 11, \Delta_\gamma = 6$ ). A significantly more complex multiplexing strategy could help further improve of coverage and will be explored in future research.



**Fig. 4. Left and right detectability.** The shaded regions in (a) and (b) represent all undetected breakpoints ( $x, y$ ) from the joining of TMPRSS2 and ERG. a) Left detected breakpoints resulting from fusion of the TMPRSS2-ERG region. b.) Right detected breakpoints. In both cases, the fraction of undetectable breakpoints is less than 5% of all possible breakpoints. c) and d) Venn diagrams (to scale)- showing the overlap of missed regions with highly-conserved repeats. The outermost circle represents the entire length of the corresponding axis. The “Repeat Block” shaded area corresponds to the sum of lengths for all repeat regions that are over 75% conserved, continuous (or with  $< 100$  bp between conserved repeats), and  $> 500$  bp in length. “Uncovered” corresponds to any positions in TMPRSS2, (c), and ERG, (d), that *may* be undetected (i.e.  $y$  coordinates that appear in shaded regions in figure (a), and  $x$  coordinates in figure (b), respectively). In both diagrams “Uncovered” is completely contained within “Repeat Blocks”.

### 3.2 Left vs. Right Breakpoint Detection

Figure 4 shows the results of our design on the TMPRSS2-ERG region (240kb  $\times$  20kb), with the obvious conclusion that less than 5% of the breakpoints remain undetected on either side (overall coverage 98%). Interestingly, the figure also differentiates between coverage (amplification), which is symmetric, and detection, which is not. To explain, note that more breakpoints are not detected on the left (*ERG*) side compared to the right (*TMPRSS2*). This is largely due to the presence of several large, highly conserved, LINE elements, in *ERG* introns (corresponding to vertical bands of uncovered breakpoints in Figure 4a). While it was possible to design primers in these regions, it was nearly impossible to design unique probes. The primers allowed breakpoints to be amplified and right-detected (by *TMPRSS2* probes), but not left-detected. In some repetitive regions, it was difficult to even design unique primers. When the sequence is not amplified, neither the left, nor the right end-point is detected, observable as shaded regions at the same breakpoints in Figure 4a and 4b. Figure 4c,d shows the total length of uncovered sequenced on each axis (1 dimensional) contained within highly-conserved “Repeat Blocks”. We see that as a fraction of the total sequence length, these regions are quite small. In the case of *ERG*, only a small fraction of total “Repeat Block” space is uncovered. A similar coverage was obtained for *CDKN2A* region (data not shown).



**Fig. 5. Detecting rearrangements in cell-lines with complex rearrangements.** Sequencing results confirming the breakpoint locations in a) CEM b) MOLT4, and c) A549 cell-lines. The presence of multiple forward primers in CEM requires the use of alternating multiplexing. d) Array results for the A549 cell line. Note that the array not only captures the left and right breakpoints, but also an inserted inversion. The remainder of the spots correspond to non-specific background signals (corresponding to repeat locations) present across runs.

### 3.3 Experimental Confirmation of CDKN2A

We had previously reported a design for the CDKN2A region, optimized using the single-sided scoring function. The design spanned 600kb, incorporating 600 primers [9]. Our PAMP-1D design successfully verified the deletion breakpoint in the Detroit 562 cell-line, but could not detect the lesion in 3 other cell-lines (Molt-4, CEM, and A549). Here, we report positive results on the 3 cell lines using a novel design that includes alternating multiplexing (See Appendix for modifications to experimental design). Note that two of three (CEM and MOLT4) resulted in experimental failure using PAMP-1D designs. In each case, the tests confirmed breakpoint boundaries previously described in the literature [8][19][16]. See Figure 5 for an overview of the results. Confirmatory sequencing validated the breakpoint boundaries (Figure 5a-c). Full sequencing results of the amplified primer products can be found in the online supplement.

Note that in the absence of alternating multiplexing, the forward primer at 21,817,940 precludes left-detection of the CEM breakpoint (Figure 5a). Interestingly, The A549 cell-line has a discontinuous 290kb *CDKN2A* deletion within which there is an internal 325 bp inversion. The array design successfully captured both left and right breakpoints as well as an internal event (Figure 5d), indicating that the technique can be successful in detecting breakpoint boundaries even in complex regions with multiple rearrangements.

### 3.4 Running Time

The update operation described in Section 2.2 is critical to the success of PAMP-2D . A naive computation of coverage for an  $200 \times 10\text{kb}$  region requires  $> .05$  CPU-min per iterations [1]. In contrast, our optimized computation runs in  $< 1.5 \times 10^{-7}$  CPU-min per iteration (including both coverage computation and updates). Both tests were run

---

<sup>1</sup> This represents the largest region for which it was possible to complete even a short test run.

on a 3.4 Ghz Pentium 4 with 1 GB RAM. Even so, the designs involve a very complex optimization. The simulations required a total of  $\sim$  7000 CPU Hours (ranging from as little as 2 minutes on average for  $10 \times 10$ kb regions to 26 hours for  $500 \times 500$ kb regions).

## 4 Discussion

Our results provide solid evidence of the feasibility of this approach for early diagnosis of cancer. The alternating primer scheme ensures that all breakpoints that can possibly be detected are detected. This scheme, in the non-dimerizing case, represents the minimal number of multiplexing reactions possible to achieve this optimal breakpoint coverage. We provide the number of multiplexing reactions as a parameter to be chosen by the experimentalist. This allows a trade-off between coverage and experimental cost/complexity. Other important trade-offs can factor into the decision making process. If one simply seeks to determine the presence of a rearrangement, then detection on either side is acceptable. In some cases, it is important to have positional information for both the left and right breakpoint coordinate. For example, the amplifying primer pair could be used individually in follow-up tests for the individual (thereby, saving cost and making a more reliable assay). Also, the predicted breakpoint can be validated via sequencing or being run on a gel. In both cases, simply amplifying the event is insufficient.

A key point of debate is the choice of relatively older technologies (PCR and hybridization), given the rapid development of new parallel sequencing technologies. To explain our choice, note that there are two facets to our strategy: a) PCR allows for the amplification of weak signals from the cancer sequence; and, b) oligonucleotide arrays allow for a cost-effective and reliable detection. On the face of it, high-throughput sequencing approaches appear to be a good alternative, as per base, such approaches are cost-effective. However, without amplification one would be primarily sequencing background DNA, not the cancerous signal. An enormous depth of coverage (and therefore cost) would be necessary to ensure detection of a weak cancerous signal. Additionally once a mutation is detected in the individual, resequencing is a costly follow-up, while PAMP returns a custom pair of primers specific to that lesion event.

Second, hybridization yields an unambiguous detection of the PCR amplification. Sequencing could be used in lieu of hybridization to detect PCR-amplified mutants, but this is more challenging than it appears. There is always the possibility of amplifying background DNA (returning to the mispriming problem) or sequencing non-amplified DNA (especially if no true lesion exists). These would not hybridize to the probe, but would confound sequence based analyses and the reconstruction of the breakpoint. Such problems are magnified by artifacts inherent to multiplexing which could lead to several non-specific amplifications in addition to the targeted breakpoint. Moreover, there is a fixed cost (several thousand dollars) for running a single sample, which makes for an expensive early diagnostic, or even regular follow-up exam, to see cancer progression or remission in a single individual, whereas custom arrays are fairly cost-effective.

A significant remaining challenge is that our coverage drops off for larger regions ( $\geq 500$ kb). The primary reason for this is an inherent requirement in our design that

each forward primer must be multiplexed with every reverse primer, and therefore cannot dimerize with it. With increased sizes, each forward primer is constrained to not dimerize with many reverse primers, which severely reduces the number of primers, and coverage. One way around this is to use a flexible multiplexing scheme. Subsets of forward primers can be permitted to dimerize with subsets of reverse primers as long as they are never in the same multiplex reaction. While this works in principle, optimizing such designs would require a substantial increase in the total number of primers (as multiple primers spanning the same genomic region would be necessary), the number of multiplexing sets, and the overall experimental complexity. As these approaches move to a more industrial or automated setting, it will become increasingly important to solve these more complex optimization problems.

## Acknowledgements

This work was supported in part by NIH grant CA119335 to the UCSD NanoTumor Center of Excellence for Cancer Nanotechnology and CA113634. Computational analysis was supported in part by the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622.

## References

1. Campbell, P., Stephens, P., Pleasance, E., O'Meara, S., Li, H., Santarius, T., Stebbings, L., Leroy, C., Edkins, S., Hardy, C., et al.: Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nature Genetics* 40(6), 722 (2008)
2. Mitelman, F., Johansson, B., Mertens, F.: The impact of translocations and gene fusions on cancer causation. *Nat. Rev. Cancer.* 7, 233–245 (2007)
3. Raphael, B., Volik, S., Yu, P., Wu, C., Huang, G., Waldman, F., Costello, J., Pienta, K., Mills, G., Bajcarowicz, K., Kobayashi, Y., Sridharan, S., Paris, P., Tao, Q., Aerni, S., Brown, R., Bashir, A., Gray, J., Cheng, J.F., Jong, P., Nefedov, M., Padilla-Nash, H., Collins, C.: A sequence based survey of the complex structural organization of tumor genomes. *Genome Biology* 9(3) (2008)
4. Ruan, Y., Ooi, H., Choo, S., Chiu, K., Zhao, X., Srinivasan, K., Yao, F., Choo, C., Liu, J., Ariyaratne, P., et al.: Fusion transcripts and transcribed retrotransposed loci discovered through comprehensive transcriptome analysis using Paired-End diTags (PETs). *Genome Res.* 17(6), 828–838 (2007)
5. Bashir, A., Volik, S., Collins, C., Bafna, V., Raphael, B.: Evaluation of Paired-End Sequencing Strategies for Detection of Genome Rearrangements in Cancer. *PLoS Computational Biology* 4(4) (2008)
6. Beigel, R., Alon, N., Apaydin, S., Fortnow, L., Kasif, S.: An Optimal Multiplex PCR Protocol for Closing Gaps in Whole Genomes. In: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB) (2001)
7. Lipson, D.: Optimization problems in design of oligonucleotides for hybridization based methods. Master's thesis, Technion - Israel Institute of Technology (2002)
8. Liu, Y., Carson, D.: A novel approach for determining cancer genomic breakpoints in the presence of normal DNA. *PLoS ONE* 2(4), e380 (2007)

9. Bashir, A., Liu, Y., Raphael, B., Carson, D., Bafna, V.: Optimization of primer design for the detection of variable genomic lesions in cancer. *Bioinformatics* 23(21), 2807 (2007)
10. Dasgupta, B., Jun, J., Mandoiu, I.: Primer Selection Methods for Detection of Genomic Inversions and Deletions via PAMP. In: *Proceedings of the 6th Asia-Pacific Bioinformatics Conference*. Imperial College Press (2008)
11. Fan, J., Chee, M., Gunderson, K., et al.: Highly parallel genomic assays. *Nature Reviews Genetics* 7(8), 632 (2006)
12. Brooks, R.: On colouring the nodes of a network. *Proc. Cambridge Phil. Soc.* 37, 194–197 (1941)
13. Welsh, D., Powell, M.: An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal* 10(1), 85–86 (1967)
14. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. *Science* 220(4598), 671–680 (1983)
15. Rocco, J., Sidransky, D.: p16 (MTS-1/CDKN2/INK4a) in Cancer Progression. *Experimental Cell Research* 264(1), 42–55 (2001)
16. Sasaki, S., Kitagawa, Y., Sekido, Y., Minna, J., Kuwano, H., Yokota, J., Kohno, T.: Molecular processes of chromosome 9p21 deletions in human cancers. *Oncogene* 22, 3792–3798 (2003)
17. Tomlins, S., Rhodes, D., Perner, S., Dhanasekaran, S., Mehra, R., Sun, X., Varambally, S., Cao, X., Tchinda, J., Kuefer, R., et al.: Recurrent Fusion of TMPRSS2 and ETS Transcription Factor Genes in Prostate Cancer. *Science* 310(5748), 644–648 (2005)
18. Wang, J., Cai, Y., Ren, C., Ittmann, M.: Expression of variant TMPRSS2/ERG fusion messenger RNAs is associated with aggressive prostate cancer. *Cancer Res.* 66, 8347–8351 (2006)
19. Kitagawa, Y., Inoue, K., Sasaki, S., Hayashi, Y., Matsuo, Y., Lieber, M.R., Mizoguchi, H., Yokota, J., Kohno, T.: Prevalent Involvement of Illegitimate V (D) J Recombination in Chromosome 9p21 Deletions in Lymphoid Leukemia. *Journal of Biological Chemistry* 277(48), 46289–46297 (2002)
20. Wang, D., Urisman, A., Liu, Y., Springer, M., Ksiazek, T., Erdman, D., Mardis, E., Hickenthal, M., Magrini, V., Eldred, J., et al.: Viral discovery and sequence recovery using DNA microarrays. *PLoS Biol.* 1(2), E2 (2003)
21. Lu, Q., Nunez, E., Lin, C., Christensen, K., Downs, T., Carson, D., Wang-Rodriguez, J., Liu, Y.: A sensitive array-based assay for identifying multiple TMPRSS2: ERG fusion gene variants. *Nucleic Acids Research* (2008)

## A Appendix

### A.1 Experimental Methods

The microarray procedure followed that of Liu and Carson [8] with the following modification. Briefly, the 5'-ends of each designed primers have the sequence of primer B (GTTTCCCAAGTCACGATC) for the subsequent step of PCR labeling with a single primer B as described previously [20][21].

### A.2 Proofs

Theorem I makes 2 assertions, which we will prove independently.

1. Let  $\mathcal{P}$  be a design with no dimerizing pairs. Using alternating multiplexing, we detect all detectable breakpoints.

$$\cup_{a,b \in \{0,1\}} \mathcal{S}_X^*(\mathcal{P}_\gamma^a \times \mathcal{P}_\gamma^b) = \mathcal{S}_X(\mathcal{P}), \cup_{a,b \in \{0,1\}} \mathcal{S}_Y^*(\mathcal{P}_\gamma^a \times \mathcal{P}_\gamma^b) = \mathcal{S}_Y(\mathcal{P})$$

2. Further, if  $\mathcal{P}_r, \mathcal{P}_\gamma$  are non-trivial then alternating multiplex yields the minimum number (4) of multiplex reactions necessary to achieve detectability.

**Proof. 1 (By contradiction).** Consider  $(x, y) \in \mathcal{S}_X(\mathcal{P})$  that is not left-detected by the strategy. All other cases are symmetric. By definition of detectability, there exists a proximal ('good') amplifiable primer pair  $p_{i^r}, p_{j^r}$ , with left probe  $b_{i^r}$ , such that  $l_{i^r} < b_{i^r} < x$ . The primer is not left-detected only if there exists a ('bad') primer  $p_{k^r}$  with  $b_{i^r} < l_{k^r} < x < b_{k^r}$ , which amplifies  $(x, y)$  but does not left-detect. Among all good and bad left primers for  $x$ , choose the pair  $(p_{i^r}, p_{k^r})$  with the most proximal probes on either side of  $x$ . By definition,  $p_{i^r}, p_{k^r}$  are adjacent, and cannot be in the same multiplex reaction, a contradiction.

**2:** In a non-trivial forward design  $\mathcal{P}_r$ , there exists pair of primers  $p_{i^r}, p_{k^r}$  with  $0 < l_{i^r} - l_{k^r} < d$ , and  $b_{i^r} \neq b_{k^r}$ . As primers and probes do not overlap in sequence, we can find a point  $x$  with  $l_{j^r} < x < b_{j^r}$ . Consider an arbitrary reverse primer  $p_{j^r}$  and choose  $y$  so that  $b_{j^r} < y < l_{j^r}$ . Consider a breakpoint  $(x, y)$ . When  $(p_{i^r}, p_{k^r})$  are in the same multiplex tube,  $p_{i^r}, p_{j^r}$  gets preferentially amplified, but not left-detected. For left-detection,  $(p_{i^r}, p_{k^r})$  must be in separate multiplex-tubes. An analogous argument can be used for non-trivial design  $\mathcal{P}_\gamma$ . As the set of forward and reverse primers is partitioned into at least two tubes each, a total of 4 reactions is needed to cover every pair.

**Proof. (Theorem 2):** Brooks' theorem states that a graph can be colored using  $\Delta$  colors provided  $\Delta \geq 3$ , and it is not a complete graph. Consider the primer-dimer-adjacency graph  $G$ . By definition, no forward (reverse) vertex has degree greater than  $\Delta_r$  ( $\Delta_\gamma$ ). We will impose the constraint that the first and last primer cannot dimerize. Note that this constraint can *always* be imposed on the primer set by the addition of dummy nodes at the beginning and end, which are adjacent to some primers (and edge-connected), but do not dimerize with any primer. This formulation is outlined in our original optimization [9]. Therefore, the graph cannot be either an odd-cycle, or a complete graph, and Brooks' theorem applies. In practice, the graphs are very sparse and we apply the Welsh-Powell algorithm to obtain good colorings.

## Online Supplement

<http://bioinf.ucsd.edu/~abashir/Recomb2009/>

# The Multi-State Perfect Phylogeny Problem with Missing and Removable Data: Solutions via Integer-Programming and Chordal Graph Theory

Dan Gusfield

Department of Computer Science, University of California, Davis  
[gusfield@cs.ucdavis.edu](mailto:gusfield@cs.ucdavis.edu)

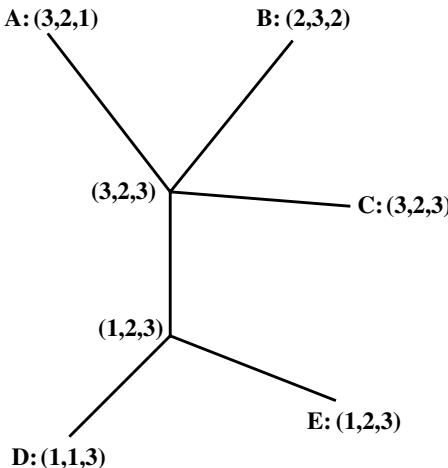
**Abstract.** The *Multi-State Perfect Phylogeny Problem* is an extension of the *Binary Perfect Phylogeny Problem*, allowing characters to take on more than two states. In this paper we consider three problems that extend the utility of the multi-state perfect phylogeny model: **The Missing Data (MD) Problem** where some entries in the input are missing and the question is whether (bounded) values for the missing data can be imputed so that the resulting data has a multi-state perfect phylogeny; **The Character-Removal (CR) Problem** where we want to minimize the number of characters to remove from the data so that the resulting data has a multi-state perfect phylogeny; and **The Missing-Data Character-Removal (MDCR) Problem** where the input has missing data and we want to impute values for the missing data to *minimize* the solution to the resulting Character-Removal Problem.

We detail Integer Linear Programming (ILP) solutions to these problems for the special case of three permitted states per character and report on extensive empirical testing of these solutions. Then we develop a general theory to solve the MD problem for an *arbitrary* number of permitted states, using chordal graph theory and results on minimal triangulation of non-chordal graphs. This establishes new necessary and sufficient conditions for the existence of a perfect phylogeny with (or without) missing data. We implement the general theory using integer linear programming, although other optimization methods are possible. We extensively explore the empirical behavior of the general solution, showing that the methods are very practical for data of size and complexity that is characteristic of many current applications in phylogenetics. Some of the empirical results for the MD problem with an arbitrary number of permitted states are very surprising, suggesting the existence of additional combinatorial structure in multi-state perfect phylogenies.

**Keywords:** computational biology, phylogenetics, perfect phylogeny, integer programming, chordal graphs, graph triangulation.

## 1 Introduction and Background

In the *k*-state Perfect Phylogeny Problem, the input is an  $n$  by  $m$  matrix  $M$  whose values are integers from the set  $Z(k) = \{1, 2, \dots, k\}$ . We refer to each



**Fig. 1.** A three-state perfect phylogeny with  $n = 5, m = 3$ . The input  $M$  consists of the five vectors that label the leaves of the tree. The subtree  $T_3(3)$  contains the leaves labeled C,D,E, and the two interior nodes.

row of  $M$  as a *taxon* (plural *taxa*), to each column of  $M$  as a *character*, and to each value in a column  $c$  as a *state* of character  $c$ . A  *$k$ -state Perfect Phylogeny* for  $M$  is a tree  $T$  with  $n$  leaves, where each leaf is labeled by a distinct taxon of  $M$ , and each internal node of  $T$  is labeled by a vector in  $Z(k)^m$  (which need not be in  $M$ ), such that for every character  $c$  and every state  $i$  of  $c$ , the subgraph of  $T$  induced by the nodes labeled with state  $i$  for character  $c$  (which we denote by  $T_c(i)$ ) must be a *connected subtree* of  $T$ . The requirement that subgraph  $T_c(i)$  be a subtree is called the *convexity* requirement. Clearly, for any character  $c$  and states  $i \neq j$ , the subtrees  $T_c(i)$  and  $T_c(j)$  of perfect phylogeny  $T$  are node disjoint. An example is shown in Figure 1.

Another way to view convexity is to arbitrarily designate a node in  $T$  as the root; direct all the edges in  $T$  away from the root, and consider this directed tree as giving a history of character mutations. The convexity requirement is then equivalent to saying that for any character/state pair  $(c, i)$ , there is at most one edge in  $T$  where the state of character  $c$  mutates to  $i$ .

The  **$k$ -state Perfect Phylogeny Problem** is to determine, for input  $M$ , if there is a  $k$ -state perfect phylogeny for  $M$ , and if there is one, to construct one. The special case of  $k = 2$  (the binary perfect phylogeny problem) has been extensively studied and is motivated by the “infinite sites” model from population genetics. The cases of  $k > 2$  arise from non-binary integer data, and are further motivated by the related “infinite alleles” model from population genetics, rather than the infinite sites model. In the last decade, SNP data (which is binary) has been the dominant data type in population genomics, but increasingly *non-binary* integer population genomic data is becoming available and informative.

If neither  $k$  nor  $n$  nor  $m$  is fixed, so  $k$  can grow with  $n$ , then the  $k$ -state perfect phylogeny problem is NP-complete [3][25]. In contrast, if  $k$  is any *fixed* integer, independent of  $n$  and  $m$ , then the  $k$ -state Perfect Phylogeny Problem can be solved in time that is polynomial in  $n$  and  $m$ . This was first shown for  $k = 2$  in [6] (and shown to be solvable in linear time in [13]), for  $k = 3$  in [5], for  $k = 3$  or 4 in [18], and for any fixed  $k$  in [1]. The later result was improved in [19], and a related method for the *near-perfect* phylogeny problem was developed in [9]. An excellent survey of most of these results appears in [8].

In this paper we consider the following three problems that extend the utility of the basic  $k$ -state Perfect Phylogeny model.

**The Missing-Data (MD) Problem:** If the input matrix  $M$  contains some cells with no values (and each cell with a value takes a value from  $Z(k) = \{1..k\}$ ), can integers from  $Z(k)$  be assigned to the cells with missing values so that the resulting matrix has a  $k$ -state perfect phylogeny?

Even for  $k = 2$  this problem is NP-complete, although a directed version of it, when  $k = 2$ , can be solved in polynomial time [23]. The requirement that missing values be selected from  $Z(k)$  is both biologically meaningful and computationally challenging. The MD problem has a very simple solution if integers up to  $n$  can be assigned [24].

**The Character-Removal (CR) Problem:** If there is no missing data in  $M$ , and  $M$  does not have a  $k$ -state perfect phylogeny, what is the *minimum* number of characters to remove so that the resulting matrix does have a  $k$ -state perfect phylogeny?

It is well known [24][7] that for binary data ( $k = 2$ ), the CR problem reduces to the node-cover problem, but this is not true for  $k > 2$ , where the CR problem was previously unaddressed.

**The Missing-Data Character-Removal (MDCR) Problem:** If the input matrix  $M$  contains some cells with no values, and the answer to the MD problem is ‘no’, how should the missing values be set in order to minimize the solution to the resulting CR problem?

Problem MD is motivated by the reality of missing entries in biological datasets. For molecular genomic applications, missing data tends to be in the 1% to 5% range, and for phylogenetic applications, missing data in the 30% range is not uncommon. The CR and MDCR problems are motivated by the common practice in phylogenetics of removing characters when the existing data does not fit the perfect phylogeny model. This is most often done when the data is binary, but the problem and practice also arise for non-binary data [8][20]. The expectation in phylogenetics is that while there may be some characters that must be removed to make a perfect phylogeny, and hence some data are lost, if a perfect phylogeny can be constructed using a large percentage of the original characters then the resulting tree will give valid evolutionary information about the taxa. There are similar scenarios that motivate removal of sites in molecular data that arises in population genomics. Of course, in order to get the most informative

tree, we want to remove as few characters or sites as possible, motivating the Character-Removal problem.

In [14] we showed how to effectively solve, using *Integer Linear Programming*, the MD and the MDCR problems (and several other related problems) on *binary* data (i.e.,  $k = 2$ ) of size and complexity that is characteristic of many current applications in biology.

In this paper, we examine the MD, CR and MDCR problems for  $k > 2$ . We have developed Integer Linear Programming (ILP) solutions to these problems for the special cases of 3, 4 and 5 permitted states per character, but will only detail here the case of 3 states. We report on extensive empirical testing of these solutions. Then we develop a general theory to solve the MD problem for an *arbitrary* number of permitted states, using chordal graph theory and results on minimal triangulation of non-chordal graphs. This establishes new necessary and sufficient conditions for the existence of a perfect phylogeny with (or without) missing data. We implement the general theory using integer linear programming, although other optimization methods are possible. We extensively explore the empirical behavior of the general solution, showing that the methods are very practical for data of size and complexity that is characteristic of many current applications in phylogenetics. Some of the empirical results for the MD problem with an arbitrary number of permitted states are very surprising, suggesting the existence of additional combinatorial structure in multi-state perfect phylogenies.

When there is no missing data, these methods also establish alternative algorithms for the Perfect Phylogeny Problem. All of the software (other than the ILP solver) needed to replicate the results stated in this paper is available through the author's homepage: <http://wwwcsif.cs.ucdavis.edu/~gusfield/>.

## 2 The MD, CR and MDCR Problems for Three States

We have developed specialized methods for the cases of  $k = 3, 4, 5$ . For lack of space, we will only discuss the case of  $k = 3$ . For the MD problem, the general method for any  $k$  is more efficient on large problem instances than the specialized methods for  $k = 3, 4, 5$ .

First we state a needed definition and fact. In a binary matrix, two columns are called “incompatible” if they contain all of the binary pairs 0,0; 0,1; 1,0; and 1,1. Otherwise they are called “compatible”. In our solution [14] to the MD and MDCR problems for *binary* data, we define a binary variable  $Y(i, j)$  for every cell  $(i, j)$  that has a missing value in  $M$ , and a binary variable  $I(p, q)$  for every pair of characters  $p, q$ . In [14] we developed compact linear inequalities that set  $I(p, q)$  to 1 if and only if the  $Y$  variables are set in a way that makes characters  $p$  and  $q$  incompatible. These inequalities are also needed in our ILP formulations for the MD and MDCR problems with  $k > 2$ ; in this paper we assume their existence, but refer the reader to [14] for details.

We next restate the main result from the paper by A. Dress and M. Steel [5] that establishes that the 3-state perfect phylogeny problem can be solved in

polynomial time. For this exposition, create another matrix  $\overline{M}$  derived from  $M$ , with three characters  $C_c(1), C_c(2), C_c(3)$ , for each character  $c$  in  $M$ . All the taxa that have state  $i$  for  $c$  in  $M$  are given state 1 for character  $C_c(i)$  in  $\overline{M}$ , and the other taxa are given state 0 for  $C_c(i)$ . So, the original input matrix  $M$  is recoded as a binary matrix  $\overline{M}$  with three expanded characters for each character in  $M$ . Each expanded character defines a split of the taxa. The main structural result in [5], interpreted in terms of  $\overline{M}$  is:

**Theorem 1.** [5] Given matrix  $M$  with  $k = 3$ , there is a perfect phylogeny for  $M$ , if and only if there is a set of characters  $S$  of  $\overline{M}$  which are pairwise compatible, and where for each character  $c$  in  $M$ ,  $S$  contains at least two of the characters  $C_c(1), C_c(2), C_c(3)$ .

In [5], a polynomial-time algorithm is given to find an appropriate set  $S$ , if one exists, but we will not need that here. In [5] we note that Theorem 1 can be used to reduce the three-state perfect phylogeny problem to the 2-SAT problem.

To explain our ILP solution to the MD problem for  $k = 3$ , we first describe how to solve the 3-state perfect phylogeny problem (without missing data) using integer linear programming. For each character  $c$  in  $M$ , let  $S(c, 1), S(c, 2), S(c, 3)$  be three binary variables associated with the characters  $C_c(1), C_c(2), C_c(3)$  in  $\overline{M}$ , and create the inequality

$$(*) \quad S(c, 1) + S(c, 2) + S(c, 3) \geq 2.$$

Variable  $S(c, z)$  (for  $z$  from  $\{1, 2, 3\}$ ) is set to 1 to indicate that character  $C_c(z)$  is selected in set  $S$ . Binary variable  $I(C_c(z); C_{c'}(z'))$  is set to 1 if and only if characters  $C_c(z)$  and  $C_{c'}(z')$  are incompatible in  $\overline{M}$ . To implement the requirement that every pair of characters in  $S$  must be compatible, create the inequality:  $S(c, z) + S(c', z') + I(C_c(z); C_{c'}(z')) \leq 2$ . Then by Theorem 1, there is a 3-state perfect phylogeny for  $M$  if and only if there is a feasible solution to the ILP created by the above inequalities. Note that when there is no missing data, the value of  $I(C_c(z); C_{c'}(z'))$  is determined from  $\overline{M}$  and fixed, but when there is missing data, its value may be affected by the values given to missing data.

To solve the MD problem for 3-state data, we extend the above ILP for the 3-state perfect phylogeny problem as follows: If cell  $(i, j)$  of  $M$  is missing a value, we add the inequality  $Y(i, j, 1) + Y(i, j, 2) + Y(i, j, 3) = 1$ , where  $Y(i, j, 1), Y(i, j, 2), Y(i, j, 3)$  are binary variables that select how the missing value in cell  $(i, j)$  should be set. We also add in the inequalities (introduced in Section 1) that set each binary variable  $I(C_c(z); C_{c'}(z'))$  to 1 if and only if the  $Y$  variables are set in a way that makes the resulting columns  $C_c(z)$  and  $C_{c'}(z')$  in  $\overline{M}$  incompatible.

We modify the above formulation for the MD problem to create an ILP that solves the MDCR problem. We remove the  $(*)$  inequalities, and create a binary variable  $R(c)$  for each character  $c$  in  $M$ . Then for each  $c$  in  $M$ , we add the inequalities  $S(c, 1) + S(c, 2) + S(c, 3) - 2R(c) \geq 0$ , which ensures that  $R(c)$  will be set to 1 only if at least two of the variables  $S(c, 1), S(c, 2), S(c, 3)$  are set to 1. Finally, use the objective function of maximizing  $\sum_c R(c)$ . Problem CR is solved in the special case that there are no missing values in  $M$ .

### 3 The MD Problem for Arbitrary $k$

In this section we develop the conceptually deepest method, leading to the most surprising empirical results of this work. We consider a formulation to the MD problem that is correct for any allowed number of states,  $k$ . Note that although there is no restriction on what  $k$  can be in the formulation,  $k$  is known and fixed for any particular problem instance, and imputed values for missing entries must be from  $Z(k)$ . We show that an approach which at first might seem impractical, but in fact works extremely well in practice on data of size and complexity that are consistent with many (but not all) current applications in phylogenetics.

Our approach to the MD problem for arbitrary  $k$  is to use a classic, but not widely exploited, result about the Perfect Phylogeny Problem, together with newer combinatorial results on chordal graphs, clique-trees and minimal triangulations of non-chordal graphs. We assume throughout that the rows of  $M$  are distinct. We first introduce the classic theorem due to Buneman [4,24].

Given an  $n$  by  $m$  input matrix  $M$  define the PI (partition intersection) graph  $G(M)$  for  $M$  as follows:  $G(M)$  has one node for every character-state pair  $(c, i)$  that occurs in  $M$ ; there is an edge between the nodes for character-state pairs  $(c, i)$  and  $(c', i')$  if and only if there is a taxon in  $M$  with state  $i$  for character  $c$  and with state  $i'$  for character  $c'$ . Thus  $G(M)$  is formed by the superposition of  $n$  cliques (one for each taxon), each of size  $m$ , and  $G(M)$  is an  $m$ -partite graph with  $m$  classes, one class for each character. Each class is sometimes referred to as a *color*. A pair of nodes  $(c, i)$ ,  $(c', i')$  where  $c \neq c'$ , defines a *legal potential edge* (or ‘legal edge’ for short) if that edge is not already in  $G(M)$ . A pair of nodes  $(c, i)$ ,  $(c, i')$ , for some *single* character  $c$ , defines an *illegal potential edge* (or ‘illegal edge’ for short), and is not in  $G(M)$ . A pair of nodes in the same class of  $G(M)$  are called a *mono-chromatic pair*.

A cycle  $C$  in a graph  $G$  is called *chordless* if  $G$  does not contain any edge  $e$  between two nodes in  $C$  unless  $e$  is an edge of  $C$ . A graph is *chordal* if and only if it contains no chordless cycles of size four or more. A *triangulation* or *chordal fill-in*  $H(G)$  of a graph  $G$  is a chordal super-graph of  $G$ , on the same nodes as  $G$ . The edges in  $H(G) - G$  are the *added edges*. Given  $M$ , a *legal triangulation* of  $G(M)$  is a triangulation where all of the added edges are legal. Note that  $G(M)$  might itself be chordal, but if not chordal, there might not be any legal triangulation of  $G(M)$ .

**Buneman’s Theorem** [4,24].  $M$  has a perfect phylogeny if and only if  $G(M)$  has a legal triangulation  $H(G(M))$ .

We will also need the following definition and facts.

**Definition.** Let  $G$  be a graph and  $T$  be a tree containing one node for each maximal clique (a clique where no additional nodes can be added) in  $G$ . It is useful to label each node  $v$  in  $T$  with the nodes of  $G$  that form the maximal clique in  $G$  associated with node  $v$  in  $T$ . Then  $T$  is defined to be a *clique-tree* for  $G$  if and only if for each node  $w$  in  $G$ , the nodes in  $T$  which contain the label  $w$  form a *connected* subtree of  $T$ .

**Theorem [11,4,12,20].** A graph  $G$  is chordal if and only if there is a clique-tree  $T$  for  $G$ .

When  $G(M)$  is the PI graph for an input  $M$ , and a legal triangulation  $H(G(M))$  exists for  $G(M)$ , each node in  $G(M)$  represents a character-state pair, and each taxon in  $M$  induces a maximal clique in  $G(M)$  and in  $H(G(M))$ . It is then easy to see that any clique tree for  $H(G(M))$  is a perfect phylogeny for  $M$ . It is known [20] that a clique tree for a chordal graph can be found in time linear in the size of the chordal graph. Hence, when  $G(M)$  has a legal triangulation, a perfect phylogeny for  $M$  can be found in linear time from  $H(G(M))$ .

### 3.1 The PI-Graph Approach to the MD Problem

If  $M$  contains missing data, let  $G(M)$  now be created by only using cells in  $M$  where the value is *not* missing. If a taxon has  $q$  out of  $m$  cells with known values, then that taxon induces a clique in  $G(M)$  of size  $q$  rather than a clique of size  $m$ ;  $G(M)$  is again an  $m$ -partite graph formed by the superposition of  $n$  cliques. The key point, which is easy to establish, is that Buneman's Theorem continues to hold for  $G(M)$  created in this way, i.e., for input  $M$  with missing values, even if  $G(M)$  now has fewer nodes. Indeed, even when  $M$  has no missing values, unless  $G(M)$  is chordal without adding any edges, a legal triangulation can be thought of as adding new data (in fact, new taxa) to  $M$  which will be represented at internal nodes of the perfect phylogeny. When  $M$  has missing data, then a legal triangulation essentially adds new data to the taxa in  $M$  that are missing entries, along with adding new taxa to  $M$ . In more detail, let  $M$  be input where a taxon  $q$  has some missing entries, and suppose there is a legal triangulation  $H(G(M))$  for  $G(M)$ ; let  $T$  be a clique-tree for  $H(G(M))$ . Every node in  $T$  corresponds to a maximal clique in  $H(G(M))$ , and the original clique in  $G(M)$  induced by taxon  $q$  must be completely contained in one or more maximal cliques in  $H(G(M))$ . Also each taxon in  $M$  that has no missing data induces a maximal clique in  $H(G(M))$ . Therefore there is one node in  $T$  for each taxon with no missing data in  $M$ , and at least one node  $v$  in  $T$  whose label contains all the nodes in the clique in  $G(M)$  associated with taxon  $q$ . We then assign taxon  $q$  to node  $v$  of  $T$ , and if  $v$  is not a leaf, we add a new edge connecting  $v$  to a new leaf labeled  $q$ . The resulting tree  $T$  is now a perfect phylogeny for  $M$ .

We next use  $T$  to impute values for any missing data in  $M$ . Consider again taxon  $q$  as above. Recall that node  $v$  in  $T$  is labeled with nodes of  $G(M)$ , and hence is labeled with character/state pairs from  $M$  with at most one state for any character. Assign all the character/states that label  $v$  to taxon  $q$ . For any character  $c$  which does not have an associated label in  $v$ , do a breath-first search from  $v$  until reaching some node  $v'$  in  $T$  which is labeled by character/state pair  $(c, i)$ , for some state  $i$ . Then label each node on the path from  $v$  to  $v'$  in  $T$  with  $(c, i)$ , assigning state  $i$  for character  $c$  to taxon  $q$ . Successively repeat this until all taxa have values for all characters. The result is a filled matrix  $M$  which has a perfect phylogeny  $T$ .

## Finding a Legal Triangulation

The PI-graph approach to the  $k$ -state Perfect Phylogeny problem is conceptually perfect to handle missing data. However, it is not clear how to efficiently find a legal triangulation in practice, or determine that there is none, so and it may seem that this would not be a productive approach. We next show, by exploiting more contemporary results about triangulation in general graphs, that this approach can be effectively implemented in practice on data of realistic size for many current application in biology. We first state some definitions and facts.

**Definition.** A triangulation of a non-chordal graph  $G$  is called *minimal* if no subset of the added edges defines a triangulation of  $G$ .

There is a large and continuing literature on the structure of minimal triangulations and on algorithms for finding them [16]. Clearly, for an input graph  $M$ , if there is a legal triangulation of  $G(M)$  (and hence a perfect phylogeny for  $M$ ), then there is a legal triangulation which is minimal, and we can apply appropriate results from the literature on minimal triangulations.

**Definition.** An  $a,b$  *separator* in a graph  $G$  is a set of nodes whose removal separates node  $a$  from node  $b$ . A *minimal  $a,b$  separator* is an  $a,b$  separator such that no subset of it is an  $a,b$  separator. A separator is called a *minimal separator* if it is a minimal  $a,b$  separator for some pair of nodes  $a,b$ . A minimal separator  $K$  is said to *cross* a minimal separator  $K'$  if  $K$  is a separator for some pair of nodes in  $K'$ .

It is easy to establish [22] that crossing is a symmetric relation for minimal separators; testing whether two minimal separators cross can be done in time proportional to the number of edges in  $G$ . Clearly, a minimal separator which is a clique cannot cross any other minimal separator, because crossing is symmetric and no separator can separate two nodes that are connected by an edge.

**Definition.** Two minimal separators that do not cross each other are said to be *parallel*.

**Definition.** A minimal separator  $K$  is *completed* by adding in all the missing edges to make  $K$  a clique.

It is also easy to establish [22] that when a minimal separator  $K$  is completed, every minimal separator that  $K$  crossed (before  $K$  was completed) now ceases to be a minimal separator.

**Definition.** A set  $Q$  of pairwise parallel minimal separators in  $G$  is said to be *maximal* if every other minimal separator in  $G$  crosses some separator in  $Q$ .

The capstone theorem in the study of minimal triangulations, due to A. Parra and P. Scheffler, is

**Minimal Triangulation Theorem [21,22]:** Every minimal triangulation of a non-chordal graph  $G$  can be created by finding a maximal set  $Q$  of pairwise parallel minimal separators in  $G$  and completing each separator in  $Q$ . Conversely, the completion of each separator in any maximal set of pairwise parallel minimal separators in  $G$  creates a minimal triangulation of  $G$ .

Now we can relate these general definitions and results about minimal triangulations to the multi-state Perfect Phylogeny Problem. Given input  $M$ , a minimal separator  $K$  in the partition intersection graph  $G(M)$  is defined to be *legal* if, for each character  $c$  in  $M$ ,  $K$  contains at most one node  $(c, i)$  associated with character  $c$ . That is, only legal edges will be added if  $K$  is completed. A minimal separator that is not legal is called *illegal*. With the above, we can establish the following new characterization of the existence of a multi-state Perfect Phylogeny, even when there is missing data.

**Theorem 2 (MSP).** *There is a perfect phylogeny for input  $M$  (even if  $M$  has missing data) if and only if 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  $Q$ .*

**Proof.** If there is a perfect phylogeny for  $M$  then there is a legal triangulation of  $G(M)$  (by Buneman's theorem and its extension to the case of missing data), and so there is a minimal triangulation  $H(G(M))$  that is legal. By the Minimal Triangulation theorem, there is a maximal set  $Q$  of pairwise parallel minimal separators of  $G(M)$  whose completion results in the graph  $H(G(M))$ . Set  $Q$  cannot contain an illegal minimal separator  $K$ , for if it did, the completion of  $K$  would add an illegal edge, contradicting the fact that the triangulation is legal. Therefore  $Q$  only contains legal minimal separators, and the fact that it is maximal (with respect to all separators, legal or not) means that every illegal minimal separator must be crossed by at least one separator in  $Q$ .

Conversely, suppose 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  $Q$ . If  $Q$  is not maximal (because there are additional *legal* minimal separators that are not crossed by any member of  $Q$ ),  $Q$  can be extended to become maximal by greedily adding in legal minimal separators until every remaining legal minimal separator crosses some separator in the extended  $Q$ . Then by the Minimal Triangulation theorem, the completion of each separator in  $Q$  results in a chordal graph. Since only legal edges were used in this triangulation, Buneman's theorem establishes that  $M$  has a perfect phylogeny.  $\square$

Theorem MSP leads to three corollaries which, when they apply, each solve the MD problem without a complex search for  $Q$  and without the need to solve an optimization problem.

**Corollary 1 (MSP1).** *If  $G(M)$  has an illegal minimal separator (or monochromatic pair of nodes) that is not crossed by any legal minimal separator, then  $M$  has no perfect phylogeny.*

**Corollary 2 (MSP2).** *If  $G(M)$  has no illegal minimal separators, then  $M$  has a perfect phylogeny.*

**Corollary 3 (MSP3).** *If no pair of legal minimal separators in  $G(M)$  cross, and every illegal minimal separator (or mono-chromatic pair of nodes) is crossed by at least one legal minimal separator, then  $M$  has a perfect phylogeny.*

Each of these corollaries applies in some problem instances in our empirical tests. Strikingly, in our tests, Corollary MSP1 applies in *almost every* problem instance that has no perfect phylogeny, and Corollaries MSP2 and MSP3 apply frequently.

Theorem MSP can be extended in a way that leads to a simpler characterization of the existence of a perfect phylogeny, with practical consequences in some cases.

**Theorem 3 (MSPN).** *There is a perfect phylogeny for input  $M$  (even if  $M$  has missing data) if and only if there is a set  $Q$  of pairwise parallel legal minimal separators in  $G(M)$  such that every mono-chromatic pair of nodes in  $G(M)$  is separated by some separator in  $Q$ .*

**Proof.** We prove the ‘if’ direction first. Let  $i, j$  be a pair of states of a character (color)  $c$  in  $M$ , and let  $u_i, u_j$  be the mono-chromatic pair of nodes in  $G(M)$  representing states  $i, j$  of  $c$ . Every  $(u_i, u_j)$  separator crosses every illegal minimal separator that contains  $(u_i, u_j)$ . By definition, an illegal minimal separator must contain a monochromatic pair of nodes. So the existence of a set of pairwise-parallel legal minimal separators that separate every monochromatic pair of nodes in  $G(M)$  demonstrates that there is a maximal set of pairwise-parallel minimal separators all of which are legal, and hence, by Theorem MSP there is a perfect phylogeny for  $M$ .

The ‘only if’ direction is more subtle. If  $M$  has a perfect phylogeny, then by the Buneman theorem there is a legal triangulation of the partition-intersection graph  $G(M)$ . As argued earlier, any clique-tree  $T$  derived from  $G(M)$  is a perfect phylogeny for  $M$ . For clarity, we will use the term “nodes” for  $G(M)$  and “vertices” for  $T$ . By convexity, for any pair of states  $i, j$  of a character  $c$  in  $M$  there must be an edge  $e(i, j)$  in  $T$  that separates all vertices in  $T$  labeled with state  $i$  for  $c$  from all vertices labeled with state  $j$  for  $c$ . Let  $u_i, u_j$  be the nodes of  $G(M)$  associated with states  $i, j$  of  $c$ . To prove this direction of the theorem, we first want to identify a legal, minimal separator in  $G(M)$  that separates  $u_i$  and  $u_j$ . To do this, we the fact that  $T$  is a clique-tree, and two facts about clique-trees.

Each vertex  $v$  in  $T$  is labeled with a subset of nodes of  $G(M)$  that form a maximal clique in  $G(M)$ . Let  $S$  and  $S'$  be the two maximal cliques in  $G(M)$  associated with two end-vertices of an edge  $e = (v, v')$  in  $T$ . The removal of  $e$  from  $T$  creates two connected subtrees  $T_v$  and  $T_{v'}$ ; let  $N_v$  and  $N_{v'}$  be the two sets of nodes in  $G_M$  that label the vertices in these subtrees. We now state two central facts about chordal graphs (in general) and their clique-trees, specialized to  $G(M)$  and  $T$ : a) the nodes in  $S \cap S'$  form a minimal separator,  $K$ , in  $G(M)$ , which is an  $a, b$  minimal separator for every pair of nodes  $a \in (T_v - K)$  and  $b \in (T_{v'} - K)$ ; b) the set of all such minimal separators, obtained from the edges in  $T$ , are pairwise parallel. See [20][22] for the general version of the first fact, and [22] for the general version of the second fact. As a side note, these facts can be used to speed up the  $k$ -state perfect phylogeny algorithms from [11][19] in practice, but we will not detail that in this paper.

Now we apply these facts to edge  $e(i, j)$ . By convexity, node  $u_i$  cannot be part of labels of vertices in both subtrees of  $T - e(i, j)$ , and the same holds for  $u_j$ .

Therefore, neither  $u_i$  nor  $u_j$  can be in the minimal separator  $K(i, j)$  of  $G(M)$  associated with  $e(i, j)$ , and hence  $K(i, j)$  separates  $u_i$  and  $u_j$ . Further,  $K(i, j)$  is legal because the nodes in  $K(i, j)$  are the intersection of two maximal cliques in  $G(M)$ , and no clique in  $G(M)$  has two nodes representing states of the same character (for then  $G(M)$  would have an illegal edge). Hence, we have proved that if  $M$  has a perfect phylogeny, it has a perfect phylogeny,  $T$ , where the set of edges of  $T$  define a set of legal, pairwise-parallel minimal separators of  $G(M)$  which separate every mono-chromatic pair of nodes.  $\square$

We can also establish a different characterization as follows. Suppose  $K$  is a minimal separator in  $G(M)$ , and  $x$  is a sequence in  $M$ . Recall that there is a clique in  $G(M)$  associated with  $x$ . Note that if one of the nodes in that clique is in the connected component  $C$  of  $G(M) - K$ , then all nodes of that clique must be in  $C \cup K$ . This follows because there can be no edge between nodes in different connected components of  $G(M) - K$ . Therefore, we say that  $K$  separates two taxa  $x$  and  $x'$  in  $M$  if and only if  $K$  separates a node in the clique for  $x$  from a node in the clique for  $x'$ .

**Theorem 4 (MSPNN).** *There is a perfect phylogeny for input  $M$  (even if  $M$  has missing data) if and only if there is a set  $Q$  of pairwise parallel legal minimal separators in  $G(M)$  such that every pair of taxa in  $M$  is separated by some separator in  $Q$ .*

### 3.2 An ILP Formulation of Problem MD for Arbitrary $k$

From theorems MSP, MSPN and MSPNN, it is now conceptually simple to formulate an integer linear program for the Perfect Phylogeny Problem and for problem MD. This ILP tries to select an appropriate subset  $Q$  of legal minimal separators in  $G(M)$ . Given input  $M$ , the approach based on Theorem MSP creates the ILP as follows: find all the legal and illegal minimal separators in  $G(M)$ ; for each legal minimal separator  $K$ , determine which minimal separators cross  $K$ ; let  $\mathcal{K}$  be the set of legal minimal separators which each cross some illegal minimal separator; create a binary variable for each minimal separator in  $\mathcal{K}$ ; a variable will be set to 1 to indicate that the associated legal minimal separator will be selected for  $Q$ ; create inequalities that forbid the selection into  $Q$  of two legal minimal separators in  $\mathcal{K}$  that cross each other; and create inequalities that require that each illegal minimal separator be crossed by at least one legal minimal separator in  $\mathcal{K}$ . Then, the MD problem has a solution (values for the missing data can be found so that the resulting data has a perfect phylogeny) if and only if these inequalities have a feasible solution. Further, if there is a feasible solution (identifying a set  $Q$ ), and  $Q$  is extended to become maximal (as in the proof of Theorem MSP), then completing the minimal separators in  $Q$  creates a chordal graph from which a clique-tree and a perfect phylogeny  $T$  for  $M$  can then be constructed; imputed values for any missing data in  $M$  can be obtained from  $T$ .

Similarly, from Theorems MSPN and MSPNN, we can formulate an ILP solution for the MD problem *without* knowing the illegal minimal separators of  $G(M)$ ; it is sufficient to know only the legal minimal separators of  $G(M)$ .

Although we have formulated and tested the problem of finding  $Q$  as an ILP, alternative (perhaps more direct) methods are possible and may be more practical, especially when the number of minimal separators found in  $G(M)$  is small.

### 3.3 Finding the Minimal Separators

All the minimal separators in  $G(M)$  can be found using the algorithm in [2], which has a worst-case running time proportional to the number of edges in  $G(M)$  times the total number of nodes in all the minimal separators. In worst case, the number of nodes in  $G(M)$  is  $km$  and the number of edges is bounded by  $\min(n, k^2) \times \binom{m}{2}$ . Note that  $k \leq n$ . However, we only need to find the minimal separators if the number of edges is less than  $km(m - 1)$ . This follows from the fact that a pair of characters  $i, j$  cannot have a perfect phylogeny (or be in a perfect phylogeny) unless the subgraph of  $G(M)$  induced by the set of state-nodes of characters  $i$  and  $j$  is acyclic [10, 24], meaning that the induced subgraph can have at most  $2k - 1$  edges. This required bound continues to apply if edges are added to  $G(M)$ . When we build  $G(M)$  from  $M$ , we check that the number of edges induced by each pair of characters is less than  $km(m - 1)$  (we call this the *graph-density test*) and then search for the minimal separators only on graphs that pass the test. Moreover, there is a faster alternative approach to finding the legal minimal separators, detailed next, in the case that there are no missing entries.

**Theorem 5.** *No minimal separator in  $Q$  can have  $m$  nodes, and so we only need to find the legal minimal separators in  $G(M)$  of size  $m - 1$  or less. When  $M$  has no missing data, the number of legal minimal separators of size  $m - 1$  or less in  $G(M)$  is bounded by the number of proper clusters in  $M$  (the main object in the algorithms of [1, 19]) and all these legal minimal separators can be found in  $O(2^k nm^2)$  time.*

Following Theorems MSPN and MSPNN, Theorem 5 can be used to more efficiently construct an ILP for the  $k$ -state perfect phylogeny problem, and this also puts a polynomial bound, for any fixed  $k$ , on the size of the ILP needed to solve the problem. That ILP will have  $O(2^k m)$  variables and  $O(2^{2k} m^2 + \min(n^2, mk^2))$  inequalities. Note that Theorems MSPN and MSPNN hold for problem MD, but Theorem 5 has only been established for the case of *no* missing data. Whether this can be generalized to the case of missing data is an open question. Another open question is to extend the PI-graph approach to the CR and MDCR problems.

## 4 Empirical Results

For biologically informative simulation results, we want data obtained from simulators that model appropriate biological processes. To generate biologically informative data that is guaranteed to have a perfect phylogeny, we use the

well-known coalescent-based program *ms* [17], with no recombination, to generate *binary* matrices that are guaranteed to have 2-state perfect phylogenies. The following theorem explains how we use those matrices to obtain data guaranteed to have a  $k$ -state perfect phylogeny, for any chosen  $k$ .

**Theorem 6.** *Suppose  $M$  is an  $n$  by  $m$  binary matrix that has a 2-state perfect phylogeny, and that  $m = m_k \times (k - 1)$ . Let  $M'$  be an  $n$  by  $m'$  matrix obtained by partitioning each row of  $M$  into  $m_k$  groups of  $k - 1$  consecutive columns each, and converting to decimal the  $(k - 1)$ -length binary number in each group. Then each column of  $M'$  has at most  $k$  distinct numbers, and if, in each column, we map those  $k$  numbers to  $Z(k)$  (in any arbitrary way), the resulting matrix is guaranteed to have a  $k$ -state perfect phylogeny.*

To generate  $k$ -state data which might not have a  $k$ -state perfect phylogeny we set the *ms* recombination parameter  $r$  to a value greater than zero so that the binary data is not guaranteed to have a binary perfect phylogeny; we again group  $k - 1$  consecutive columns, creating binary numbers of length  $k - 1$ . In each such group, we scan the rows from top to bottom to find the first  $k$  distinct binary numbers, and map these to the states 0 to  $k - 1$ ; any binary number outside that set of  $k$  binary numbers is randomly mapped to one of the  $k$  states. When  $r$  is small, this problem instance might have a  $k$ -state perfect phylogeny but is not guaranteed to. As  $r$  grows, the chance that this data has a perfect phylogeny falls. We used the ILP solver Cplex 11, running on a 2.5 GHz. iMac with 3 Gig. of memory.

#### 4.1 Empirical Results for $k = 3$

The ILP formulations developed for problems MD, CR, and MDCR for the special case of  $k = 3$  are conceptually simple, and so the main result of this section is the empirical observation that those ILP formulations generally solve very quickly for biologically meaningful ranges of data. We have done extensive testing with a wide range of parameters, with results that are consistent with the illustrative results shown in Table 1. The times reported are for solving the ILPs; these are the times of interest. The times to generate the ILPs ranged from under one second to a small number of seconds, but are not reported because the Perl programs that generate the ILPs are highly unoptimized.

#### 4.2 Empirical Results for Solving Problem MD for Arbitrary $k$

While the approach to problem MD based on PI graphs is conceptually correct, one might expect (and we did initially expect) that it would not be practical. There were two potential bottlenecks, the time needed to find all the minimal separators in  $G(M)$ , and the time needed to solve the resulting ILPs. The main result of this section is the surprising empirical observation that this approach is very practical in data sizes that cover many current, large phylogenetic applications, although not genomic size applications. We detail illustrative empirical

**Table 1.** Average ILP execution times (in seconds unless noted otherwise) for problems MD, MDCR and CR (problem MDCR reduces to problem CR when the percent deletion is 0) with  $k = 3$ , on relatively large problem instances. Each average is taken over 100 instances (except for the last entry in rows b and c) where a 3-state perfect phylogeny exists. The time for the last entries in rows b and c are averaged over ten instances. The header shows the percentage of cells whose values were removed before solving the problems. Row a) is for problem MD on data of size 50 by 25, and row b) is for problem MD on data of size 100 by 50. Times for instances of problem MD where the data does not have a 3-state perfect phylogeny are similar, but smaller. Row c) is for problem MDCR on data of size 50 by 25. Surprisingly, the times are very similar to the times for problem MD, except for the case of 35% deletion where the variance in the times for the MDCR solution becomes large. In the ten executions of MDCR taking an average of 43 minutes, seven took under two seconds, one took two minutes, one took around two hours, and one took over five hours. Problem MDCR did not solve quickly enough on data of size 100 by 50 for in-depth exploration, except for zero deletions, where it solved in an average of 0.05 seconds. On data of size 30 by 100, and 1% deleted values, the MDCR problem solved in an average of 3.3 seconds; with 5% deleted values it solved in an average of 13 seconds.

	0%	1%	5%	10%	20%	35%
a	0.01	0.06	0.18	0.32	0.6	32
b	0.024	0.6	1.8	3.04	19.6	1hr.20mins.
c	0.018	0.06	0.2	0.38	0.81	43 mins.

results in Table 2. The programs implementing the PI-graph approach are written in Perl, except for the program to find all of the minimal separators and their crossing relations, which is written in C for greater efficiency.

We separate the tests with  $ms$  recombination parameter,  $r$ , set to zero (where a perfect phylogeny is guaranteed), from tests with  $r > 0$  (where a perfect phylogeny is not guaranteed, although one might exist). The choice of  $r$  is critical to test how the PI-graph approach performs on data that does not have a perfect phylogeny, and to see how often such data acquires a perfect phylogeny as the amount of missing data increases. If  $r$  is set too high, most instances fail the graph-density test, establishing trivially that there is no perfect phylogeny without needing to try to find a legal triangulation of  $G(M)$ . If  $r$  is set too low, then there will be a perfect phylogeny. Through experimentation, we have found values of  $r$  that provide a good balance; generally  $r$  must decrease as the size of the problem increases.

The most striking empirical results, and the key observed reasons for the efficiency of the PI-graph approach are: 1) In data where there is a perfect phylogeny we observe vastly fewer minimal separators than are possible in worst-case; the minimal separators are found surprisingly quickly in practice; most of the observed legal minimal separators do not cross any illegal minimal separator, and so do not enter into the ILPs; many of the problem instances are solved by the use of Corollaries MSP2 and MSP3; in cases where an ILP must be solved, the ILPs are tiny and most solve in the Cplex pre-solver; all the ILPs, whether solved

**Table 2.** Illustrative empirical results for the PI-graph approach. The results in each row are based on 100 instances (except for the case of  $n = 100$ , where 25 instances were used). When  $r = 0$ , each instance is guaranteed to have a perfect phylogeny. When  $r > 0$ , an instance might have a perfect phylogeny, but we are interested in the instances which do not (the number of instances in any row that have no perfect phylogeny equals  $d + c1 + \inf$ ). Details for the column labels:  $n, m$  = dimension of  $M$ ;  $k$  = number of allowed states;  $r$  =  $ms$  recombination parameter; % miss = expected percentage of missing entries; #  $v, e$  = average number of nodes and edges in  $G(M)$ ;  $L, I_{seps}$  = average number of legal and illegal minimal separators in  $G(M)$ ; sep time = average time to find all the minimal separators and the needed crossing relations and build the ILP if needed; %  $d, c1, c2, c3$  = percentage of problem instances failing the graph-density test, or solved using Corollaries MSP1, MSP2 and MSP3 respectively; % ilps = percentage of instances that must be solved as ILPs; # var, con = average number of variables and constraints in the generated ILPs;  $\inf$  = number of the ILPs that are infeasible, establishing that the problem instance has no perfect phylogeny; % pre = percentage of the generated ILPs that were solved in the Cplex pre-solver. Note that the number of ILP variables is the number of legal minimal separators that cross at least one illegal minimal separator. Comparing that number to the number of legal minimal separators shows that most legal minimal separators do not cross any illegal minimal separator. All ILPs solved in 0.00 Cplex-reported seconds.

$n, m$	$k$	$r$	% miss	# $v, e$	# $L, I_{seps}$	sep time	% $d, c1, c2, c3$	% ilps	# var	con	# $\inf$	% pre
20, 20	4	0	0	70.9, 990	16, 0.47	0.07 s	0, 0, 72, 27	1	3, 2	0	100	
20, 20	4	0	5	69.8, 952	16.4, 0.75	0.071 s	0, 0, 64, 33	3	3.6, 2.6	0	100	
20, 20	4	0	20	66.8, 840	17.7, 1.6	0.066 s	0, 0, 41, 40	19	5.3, 4.3	0	100	
20, 20	4	0	35	62.3, 696	20.6, 5.4	0.066 s	0, 0, 10, 15	75	8.4, 10.3	0	88	
40, 40	10	0	0	292, 8554	49, 6	0.850 s	0, 0, 23, 33	44	9, 5	0	91	
40, 40	10	0	5	287, 8221	51, 8	0.855 s	0, 0, 13, 30	57	9, 15	0	95	
40, 40	10	0	20	272, 7141	57, 27	1.03 s	0, 0, 0, 11	89	19, 36	0	91	
40, 40	10	0	35	255, 5874	69, 11	2.3 s	0, 0, 0, 0	100	36, 136	0	40	
40, 40	20	0	0	447, 12562	56, 23	1.44 s	0, 0, 3, 21	76	13, 41	0	72	
40, 40	20	0	5	439, 11954	60, 31	1.58 s	0, 0, 2, 19	79	15, 51	0	71	
40, 40	20	0	20	410, 9983	69, 90	2.58 s	0, 0, 0, 1	99	28, 111	0	63	
60, 60	15	0	0	601, 27072	83, 31	6.1 s	0, 0, 1, 24	75	13, 43	0	89	
60, 60	15	0	10	581, 24904	89, 56	7.4 s	0, 0, 0, 9	91	22, 67	0	89	
60, 60	15	0	20	559, 22553	94, 107	9.9 s	0, 0, 0, 1	99	37, 118	0	85	
80, 80	10	0	0	613, 38290	96, 19	12 s	0, 0, 4, 44	52	11, 31	0	96	
80, 80	10	0	5	606, 37093	99, 24	12.6 s	0, 0, 2, 39	59	13, 37	0	97	
80, 80	10	0	10	597, 35841	102, 37	14 s	0, 0, 0, 21	79	18, 48	0	95	
80, 80	10	0	20	584, 33863	109, 70	18 s	0, 0, 0, 7	93	32, 83	0	87	
80, 80	20	0	0	1026, 62540	118, 85	32 s	0, 0, 0, 12	88	21, 106	0	83	
80, 80	20	0	1	1023, 62081	118, 91	33 s	0, 0, 0, 12	88	22, 113	0	84	
100, 100	10	0	0	791, 62733	122, 33	33 s	0, 0, 0, 36	64	13, 51	0	94	
100, 100	10	0	5	783, 60924	123, 42	34 s	0, 0, 0, 36	64	17, 60	0	94	
20, 20	10	1.5	0	124, 1669	21, 5	0.099 s	5, 26, 43, 11	12	6,8	0	100	
20, 20	10	1.5	10	119, 1493	23, 6	0.096 s	3, 27, 27, 23	20	6,8	1	95	
20, 20	10	1.5	35	103, 1015	29, 16	0.095 s	1, 12, 3, 3	81	12, 21	1	75	
40, 40	10	1.25	0	289, 8738	48, 18	1.03 s	37, 28, 4, 13	18	8, 13	1	83	
40, 40	10	1.25	20	267, 7217	55, 97	2.4 s	15, 46, 1, 4	34	20,56	1	85	
40, 40	10	1.25	35	250, 5899	67, 334	7.3 s	3, 55, 0, 0	42	36,145	1	41	
80, 80	10	0.75	0	618, 39621	93, 31	13 s	52, 26, 3, 4	15	10, 26	0	93	
80, 80	10	0.75	5	610, 38359	97, 41	15 s	50, 28, 1, 3	18	13, 31	0	94	
80, 80	10	0.75	20	584, 34353	105, 117	29 s	35, 42, 1, 2	20	36,86	0	90	
80, 80	20	0.5	0	1023, 62911	112, 117	49 s	2, 48, 0, 5	45	20, 104	0	84	
80, 80	20	0.5	5	1006, 60449	114, 159	55 s	2, 49, 0, 3	46	27, 128	0	83	
80, 80	20	0.5	20	954, 52563	107, 321	82 s	0, 54, 0, 0	46	60, 264	0	65	

in the pre-solver or not, are solved by Cplex in 0.00 Cplex-reported seconds. 2) In instances which do not have a perfect phylogeny (and do not immediately fail the graph-density test), the solution to the MD problem was almost always detected by the use of Corollary MSP1 without even constructing an ILP. The few ILPs that were constructed for instances without a perfect phylogeny also solved in 0.00 Cplex-reported seconds. Theoretical explanations of these observations will likely be probabilistic and be related to the way the datasets were generated.

## Acknowledgments

I thank Rob Gysel for programming the algorithm in [2] which finds all the minimal separators in a graph and the crossing pairs involving legal minimal separators. I thank the reviewers for careful reading and helpful comments. This research was partially supported by NSF grants SEI-BIO 0513910, CCF-0515378, and IIS-0803564.

## 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 J. on Computing* 23, 1216–1224 (1994)
2. Berry, A., Bordat, J.-P., Cogis, O.: Generating All the Minimal Separators of a Graph. In: Widmayer, P., Neyer, G., Eidenbenz, S. (eds.) *WG 1999. LNCS*, vol. 1665, pp. 167–172. Springer, Heidelberg (1999)
3. Bodlaender, H., Fellows, M., Warnow, T.: Two strikes against perfect phylogeny. In: *Proc. of the 19'th Inter. colloquium on Automata, Languages and Programming*, pp. 273–283 (1992)
4. Buneman, P.: A characterization of rigid circuit graphs. *Discrete Math.* 9, 205–212 (1974)
5. Dress, A., Steel, M.: Convex tree realizations of partitions. *Applied Math. Letters* 5, 3–6 (1993)
6. Estabrook, G., Johnson, C., McMorris, F.: An idealized concept of the true cladistic character. *Math. Bioscience* 23, 263–272 (1975)
7. Felsenstein, J.: *Inferring Phylogenies*. Sinauer, Sunderland (2004)
8. Fernandez-Baca, D.: The perfect phylogeny problem. In: Du, D.Z., Cheng, X. (eds.) *Steiner Trees in Industries*. Kluwer Academic Publishers, Dordrecht (2000)
9. Fernandez-Baca, D., Lagergren, J.: A polynomial-time algorithm for near-perfect phylogeny. In: Meyer auf der Heide, F., Monien, B. (eds.) *ICALP 1996. LNCS*, vol. 1099, pp. 670–680. Springer, Heidelberg (1996)
10. Fitch, W.: Towards finding the tree of maximum parsimony. In: Estabrook, G.F. (ed.) *Proceedings of the eighth international conference on numerical taxonomy*, pp. 189–230. W.H. Freeman, New York (1975)
11. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combinatorial Theory, B* 16, 47–56 (1974)
12. Golumbic, M.C.: *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York (1980)
13. Gusfield, D.: Efficient algorithms for inferring evolutionary history. *Networks* 21, 19–28 (1991)

14. 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)
15. Gusfield, D., Wu, Y.: The three-state perfect phylogeny problem reduces to 2-SAT (to appear)
16. Heggernes, P.: Minimal triangulations of graphs: A survey. *Discrete Mathematics* 306, 297–317 (2006)
17. Hudson, R.: Generating samples under the Wright-Fisher neutral model of genetic variation. *Bioinformatics* 18(2), 337–338 (2002)
18. Kannan, S., Warnow, T.: Inferring evolutionary history from DNA sequences. *SIAM J. on Computing* 23, 713–737 (1994)
19. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies when the number of character states is fixed. *SIAM J. on Computing* 26, 1749–1763 (1997)
20. McKee, T.A., McMorris, F.R.: Topics in Intersection Graph Theory. Siam Monographs on Discrete Mathematics (1999)
21. Parra, A., Scheffler, P.: How to use the minimal separators of a graph for its chordal triangulation. In: Fülop, Z., Gecseg, F. (eds.) ICALP 1995. LNCS, vol. 944, pp. 123–134. Springer, Heidelberg (1995)
22. Parra, A., Scheffler, P.: Characterizations and algorithmic applications of chordal graph embeddings. *Discrete Applied Mathematics* 79, 171–188 (1997)
23. Pe'er, I., Pupko, T., Shamir, R., Sharan, R.: Incomplete directed perfect phylogeny. *SIAM J. on Computing* 33, 590–607 (2004)
24. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
25. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. *J. of Classification* 9, 91–116 (1992)

# COE: A General Approach for Efficient Genome-Wide Two-Locus Epistasis Test in Disease Association Study

Xiang Zhang<sup>1</sup>, Feng Pan<sup>1</sup>, Yuying Xie<sup>2</sup>, Fei Zou<sup>3</sup>, and Wei Wang<sup>1</sup>

<sup>1</sup> Department of Computer Science, University of North Carolina at Chapel Hill  
`{xiang, panfeng, weiwang}@cs.unc.edu`

<sup>2</sup> Department of Genetics, University of North Carolina at Chapel Hill  
`xyy@email.unc.edu`

<sup>3</sup> Department of Biostatistics, University of North Carolina at Chapel Hill  
`fzou@bios.unc.edu`

**Abstract.** The availability of high density single nucleotide polymorphisms (SNPs) data has made genome-wide association study computationally challenging. Two-locus epistasis (gene-gene interaction) detection has attracted great research interest as a promising method for genetic analysis of complex diseases. In this paper, we propose a general approach, COE, for efficient large scale gene-gene interaction analysis, which supports a wide range of tests. In particular, we show that many commonly used statistics are convex functions. From the observed values of the events in two-locus association test, we can develop an upper bound of the test value. Such an upper bound only depends on single-locus test and the genotype of the SNP-pair. We thus group and index SNP-pairs by their genotypes. This indexing structure can benefit the computation of all convex statistics. Utilizing the upper bound and the indexing structure, we can prune most of the SNP-pairs without compromising the optimality of the result. Our approach is especially efficient for large permutation test. Extensive experiments demonstrate that our approach provides orders of magnitude performance improvement over the brute force approach.

## 1 Introduction

High throughput genotyping technologies produce vast amounts of genetic polymorphism data which empowers genome-wide association study, and at the same time, makes it a computationally challenging task [16][20][24]. As the most abundant source of genetic variations, the number of single nucleotide polymorphisms (SNPs) in public datasets is up to millions[1][3]. Through analyzing genetic variation across a population consisting of disease (case) and healthy (control) individuals, the goal of disease association study is to find the genetic factors underlying the disease phenotypes. Growing evidence suggests that many diseases are likely caused by the joint effect of multiple genes [8][29]. The interaction between genes is also referred to as epistasis [10]. In an epistatic interaction, each gene may only have weak association with the disease. But when combined, they have strong effect on the disease. A large amount of research has been devoted to find epistatic interactions between genes [4][11][17], among which the *two-locus association mapping* has attracted most attention. The goal is to find

SNP-pairs having strong association with the phenotype. Important findings are appearing in the literature from studying the association between phenotypes and SNP-pairs [26][27][33].

Two critical issues need to be addressed in epistasis detection – one from the *statistical* side, and one from the *computational* side. The statistic issue is to develop statistical tests that have strong power in capturing epistatic interactions. Commonly used statistics in disease association study include: chi-square test, G-test, information-theoretic association measurements, and trend test [4][21][31]. Different tests are good at detecting different epistatic interactions, and there is no single winner. Another thorny challenge in epistasis detection is the computational burden posed by the huge amount of SNPs genotyped in the whole genome. The enormous search space often makes the complete genome-wide epistasis detection intractable.

The computational issue is further compounded by the well-known multiple test problem, which can be described as the potential increase in Type I error when tests are performed multiple times. Let  $\alpha$  be the significant level for each independent test. If  $n$  independent comparisons are performed, the family-wise error  $\alpha'$  is given by  $\alpha' = 1 - (1 - \alpha)^n$ . For example, if  $\alpha = 0.05$  and  $n = 20$ , then  $\alpha' = 1 - 0.95^{20} = 0.64$ . We have probability 0.64 to get at least one spurious result. Permutation test is a standard procedure for family-wise error rate controlling. By repeating the test many times with randomly permuted phenotype, a critical threshold can be established to assess the statistical significance of the findings. Ideally, permutation test should be performed in the genome-wide scale. In practice, however, permutation test is usually reserved for a small number of candidate SNPs. This is because large permutation test usually entails prohibitively long computation time. For example, if the number of SNPs is 10,000, and the number of permutations is 1,000. The number of SNP-pairs need to be tested in a two-locus epistasis detection is about  $5 \times 10^{10}$ . In this paper, we focus on addressing the computational challenges of two-locus epistatic detection when large permutation test is needed. In the following discussion, we briefly review the related work from a computational point of view.

Exhaustive algorithms [19][22] have been developed for small datasets consisting of tens to hundreds of SNPs. Since they explicitly enumerate all possible SNP combinations, they are not well adapted to genome-wide association studies. Genetic algorithm [7] has been proposed. However, this heuristic approach does not guarantee to find the optimal solution. A two-step approach [14][30] is commonly used to reduce the computational burden. The idea is to first select a subset of important SNPs according to some criteria, which is also known as SNP tagging [9][15][28]. Then in the second step, an exhaustive search is performed to find the interactions among the selected SNPs. This approach is incomplete since it ignores the interactions among the SNPs that individually have weak association with the phenotype. A case study on colon cancer demonstrates that the two-step approach may miss important interacting SNPs.

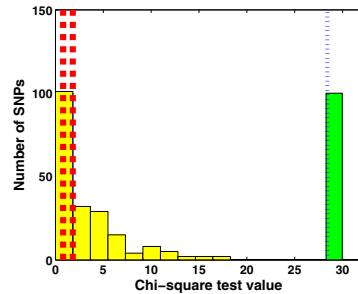
## 1.1 A Case Study of Colon Cancer

We perform two-locus chi-square test on genome-wide mouse SNP data. The dataset is extracted from [32] and [2]. There are 14 cases out of 32 individuals. The number of SNPs is 132,896. The top-100 SNP-pairs having highest chi-square test values are

recorded. We show that a two-step approach will fail to identify most of the significant pairs. We compute the single-locus chi-square test value for every SNP in these pairs. In Figure 1 the yellow bars show the histogram of their single-locus test values. More than half of the SNPs have very low test values. However, when combined with other SNPs, the test values dramatically increases. The green bar in the figure represents the histogram of the two-locus test values of the pairs. Since the two-step approach ignores the interactions between SNPs that individually have weak association with the phenotype, a majority of the top-100 SNP-pairs will not be identified.

Among the SNP pairs identified above, the interaction between the SNP located at 86,627,952 base pair on Chromosome 2 and the SNP located at 94,546,781 base pair on Chromosome 6 was reported previously [25]. They correspond to two candidate genes: Ptpn1, located on Chromosome 2 from 90,269,911 to 90,319,327, and Lrig1, located on Chromosome 6 from 95,067,900 to 95,079,646. Numerous studies have emphasized the crucial importance of Ptpn1 to colon cancer susceptibility. It is well known that immune system may play a protective role on colon-cancer, indicating the potential importance of Ptpn1 to cancer susceptibility [18]. Ptpn1 knock-out mice display an impaired immune system with decreased B cell number, abnormal B cell differentiation and shortened life-span. Previous evidence also suggests that Lrig1 acts as a feedback negative regulator of signaling by receptor tyrosine kinases through a mechanism that involves enhancement of receptor ubiquitination and accelerated intracellular degradation. Receptor tyrosine kinases including EGFR/ERBB1 are believed to be a main player on colon cancer-genesis, and Egfr expression has correlated with poor prognosis of colon cancer. Therefore, Lrig1 is a good candidate for colon cancer susceptibility [13]. Each of the two genes individually shows very weak association signal. Their single-locus chi-square test values are 1.81 and 0.79 respectively, depicted by the two red dotted lines to the left in Figure 1. However, this pair of genes jointly show much stronger association. The two-locus test value is 28.4, depicted by the blue dotted line to the right in Figure 1. This implies a strong epistatic interaction between the two genes. Using a two-step approach, however, these two SNPs will not be selected for interaction study since they both have very low single-locus test values.

Some recent work [34][35] has taken the initial steps to develop complete algorithms for genome-wide two-locus epistasis detection: FastANOVA [34] for two-locus ANOVA (analysis of variance) test on quantitative traits and FastChi [35] for two-locus chi-square test on case-control phenotypes. Both methods rework the formula of ANOVA test and Chi-square test to estimate an upper bound of the test value for SNP pairs. These upper bounds are used to identify candidate SNP pairs that may have strong epistatic effect. Repetitive computation in a permutation test is also identified and performed once whose results are stored for use by all permutations. These two strategies lead to substantial speedup, especially for large permutation test, without compromising the accuracy of the test. These approaches guarantee to find the optimal solutions.



However, a common drawback of these methods is that they are designed for specific tests, i.e., chi-square test and ANOVA test. The upper bounds used in these methods do not work for other statistical tests, which are also routinely used by researchers. In addition, new statistics for epistasis detection are continually emerging in the literature [5][12][36]. Therefore, it is desirable to develop a general model that supports a variety of statistical tests.

In this paper, we propose a general approach, COE<sup>1</sup>, to scale-up the process of genome-wide two-locus epistasis detection. Our method guarantees to find the optimal solution. A significant improvement over previous methods is that our approach can be applied to a wide range of commonly used statistical tests. We show that a key property of these statistics is that they are all convex functions of the observed values of certain events in two-locus tests. This allows us to apply the convex optimization techniques [6]. Specifically, by examining the contingency tables, we can derive constraints on these observed values. Utilizing these constraints, an upper bound can be derived for the two-locus test value. Similar to the approaches in [34][35], this upper bound only depends on single-locus test and the genotype of the SNP-pairs. It avoids redundant computation in permutation test by grouping and indexing the SNP-pairs by their genotypes. An important difference, however, is that the upper bound presented in this paper is general and much tighter than those in previous methods such as FastChi. It supports all tests using convex statistics and can prune the search space more efficiently. As a result, our method is orders of magnitude faster than the brute force approach, in which all SNP-pairs need to be evaluated for their test values, and is an order of magnitude faster than the pruning strategies used in previous methods such as FastChi. In this paper, we focus on the case where SNPs are binary variables which can be encoded by {0, 1}. The principle introduced here is also applicable to heterozygous case where SNPs are encoded using {0, 1, 2}.

## 2 Problem Formalization

Let  $\{X_1, X_2, \dots, X_N\}$  be the set of all biallelic SNPs for  $M$  individuals, and  $Y$  be the binary phenotype of interest (e.g., disease or non-disease). We adopt the convention of using 0 to represent majority allele and 1 to represent minority allele, and use 0 for non-disease and 1 for disease. We use  $\mathcal{T}$  to denote the statistical test. Specifically, we represent the test value of SNP  $X_i$  and phenotype  $Y$  as  $\mathcal{T}(X_i, Y)$ , and represent the test value of SNP-pair  $(X_i X_j)$  and  $Y$  as  $\mathcal{T}(X_i X_j, Y)$ . A contingency table, which records the observed values of all events, is the basis for many statistical tests. Table II shows contingency tables for the single-locus test  $\mathcal{T}(X_i, Y)$ , genotype relationship between SNPs  $X_i$  and  $X_j$ , and two-locus test  $\mathcal{T}(X_i X_j, Y)$ .

The goal of permutation test is to find a critical threshold value. A two-locus epistasis detection with permutation test is typically conducted as follows [21][34][35]. A permutation  $Y_k$  of  $Y$  represents a random reshuffling of the phenotype  $Y$ . In each permutation, the phenotype values are randomly reassigned to individuals with no replacement. Let  $Y' = \{Y_1, Y_2, \dots, Y_K\}$  be the set of  $K$  permutations of  $Y$ . For each permutation  $Y_k \in Y'$ , let  $\mathcal{T}_{Y_k}$  represent the maximum test value among all SNP-pairs, i.e.,

---

<sup>1</sup> COE stands for Convex Optimization-based Epistasis detection algorithm.

**Table 1.** Contingency Tables

(a) $X_i$ and $Y$			(b) $X_i$ and $X_j$					
	$X_i = 0$	$X_i = 1$	Total		$X_i = 0$	$X_i = 1$	Total	
$Y = 0$	event $A$	event $B$		$X_j = 0$	event $S$	event $T$		
$Y = 1$	event $C$	event $D$		$X_j = 1$	event $P$	event $Q$		
Total			$M$	Total			$M$	

(c) $X_i X_j$ and $Y$							
	$X_i = 0$		$X_i = 1$		Total		
	$X_j = 0$	$X_j = 1$	$X_j = 0$	$X_j = 1$	Total		
$Y = 0$	event $a_1$	event $a_2$	event $b_1$	event $b_2$			
$Y = 1$	event $c_1$	event $c_2$	event $d_1$	event $d_2$			
Total							$M$

$\mathcal{T}_{Y_k} = \max\{\mathcal{T}(X_i X_j, Y_k) | 1 \leq i < j \leq N\}$ . The distribution of  $\{\mathcal{T}_{Y_k} | Y_k \in Y'\}$  is used as the null distribution. Given a Type I error threshold  $\alpha$ , the *critical value*  $\mathcal{T}_\alpha$  is the  $\alpha K$ -th largest value in  $\{\mathcal{T}_{Y_k} | Y_k \in Y'\}$ . After determining the critical value  $\mathcal{T}_\alpha$ , a SNP-pair  $(X_i X_j)$  is considered significant if its test value with the original phenotype  $Y$  exceeds the critical value, i.e.,  $\mathcal{T}(X_i X_j, Y) \geq \mathcal{T}_\alpha$ .

Determining the critical value is computationally more demanding than finding significant SNP-pairs, since the test procedure needs to be repeated for every permutation in order to find the maximum values. These two problems can be formalized as follows.

**Determining Critical Value:** For a given Type I error threshold  $\alpha$ , find the critical value  $\mathcal{T}_\alpha$ , which is the  $\alpha K$ -th largest value in  $\{\mathcal{T}_{Y_k} | Y_k \in Y'\}$ .

**Finding Significant SNP-pairs:** For a given critical value  $\mathcal{T}_\alpha$ , find the significant SNP-pairs  $(X_i X_j)$  such that  $\mathcal{T}(X_i X_j, Y) \geq F_\alpha$ .

In the remainder of the paper, we first show the convexity of common statistics. Then we discuss how to establish an upper bound of two-locus test and use it in the algorithm to efficiently solve the two problems.

### 3 Convexity of Common Test Statistics

In this section, we show that many commonly used statistics are convex functions. Since there are many statistics in the literature, it is impossible to exhaustively enumerate all of them. We focus on four widely used statistics: chi-square test, G-test, entropy-based statistic, and Cochran-Armitage trend test.

Let  $A, B, C, D, S, T, P, Q, a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2$  represent the events as shown in Table 1. Let  $E_{event}$  and  $O_{event}$  denote the expected value and observed value of an event. Suppose that  $\mathbb{E}_0 = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$ ,  $\mathbb{E}_1 = \{a_1, a_2, c_1, c_2\}$ , and  $\mathbb{E}_2 = \{b_1, b_2, d_1, d_2\}$ . The two-locus chi-square tests can be calculated as follows:

$$\chi^2(X_i X_j, Y) = \underbrace{\sum_{event \in \mathbb{E}_1} \frac{(O_{event} - E_{event})^2}{E_{event}}}_{\chi_1^2(X_i X_j, Y)} + \underbrace{\sum_{event \in \mathbb{E}_2} \frac{(O_{event} - E_{event})^2}{E_{event}}}_{\chi_2^2(X_i X_j, Y)}. \quad (1)$$

Note that we intentionally break the calculation into two components: one for the events in  $\mathbb{E}_1$ , denoted as  $\chi^2_1(X_iX_jY)$ , and one for the events in  $\mathbb{E}_2$ , denoted as  $\chi^2_2(X_iX_jY)$ . The reason for separating these two components is that each of these two components is a convex function (See Lemma 1).

The G-test, also known as a likelihood ratio test for goodness of fit, is an alternative to the chi-square test. The formula for two-locus G-test is

$$G(X_iX_j, Y) = 2 \sum_{event \in \mathbb{E}_1} O_{event} \cdot \ln\left(\frac{O_{event}}{E_{event}}\right) + 2 \sum_{event \in \mathbb{E}_2} O_{event} \cdot \ln\left(\frac{O_{event}}{E_{event}}\right). \quad (2)$$

Information-theoretic measurements have been proposed for association study [12, 36]. We examine the mutual information measure, which is the basic form of many other measurements. The mutual information between SNP-pair ( $X_iX_j$ ) and phenotype  $Y$  is  $I(Y; X_iX_j) = H(Y) + H(X_iX_j) - H(X_iX_jY)$ , in which the joint entropy  $-H(X_iX_jY)$  is calculated as

$$-H(X_iX_jY) = \sum_{event \in \mathbb{E}_1} \frac{O_{event}}{M} \cdot \log \frac{O_{event}}{M} + \sum_{event \in \mathbb{E}_2} \frac{O_{event}}{M} \cdot \log \frac{O_{event}}{M}. \quad (3)$$

Let  $\mathcal{T}(X_iX_j, Y)$  represent any one of  $\chi^2(X_iX_j, Y)$ ,  $G(X_iX_j, Y)$ , and  $-H(X_iX_jY)$ . Let  $\mathcal{T}_1(X_iX_jY)$  denote the component for events in  $\mathbb{E}_1$ , and  $\mathcal{T}_2(X_iX_jY)$  denote the component for events in  $\mathbb{E}_2$ . The following lemma shows the convexity of  $\mathcal{T}_1(X_iX_jY)$  and  $\mathcal{T}_2(X_iX_jY)$ .

**Lemma 1.** *Given the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ ,  $\mathcal{T}_1(X_iX_jY)$  is a convex function of  $O_{c_2}$ , and  $\mathcal{T}_2(X_iX_jY)$  is a convex function of  $O_{d_2}$ .*

*Proof.* See Appendix.

The Cochran-Armitage test for trend is another widely used statistic in genetic association study. Let  $Z = (O_{c_1} - pO_S)(s_1 - \bar{s}) + (O_{c_2} - pO_P)(s_2 - \bar{s}) + (O_{d_1} - pO_T)(s_3 - \bar{s}) + (O_{d_2} - pO_Q)(s_4 - \bar{s})$ . The Cochran-Armitage two-locus test can be calculated as

$$z^2 = Z^2 / [p(1-p)(O_S(s_1 - \bar{s})^2 + O_P(s_2 - \bar{s})^2 + O_T(s_3 - \bar{s})^2 + O_Q(s_4 - \bar{s})^2)],$$

where  $p$  is the percentage of cases in the case-control population,  $s_i$  ( $i \in \{1, 2, 3, 4\}$ ) are user specified scores for the four possible genotype combinations of ( $X_iX_j$ ): {00, 01, 10, 11}, and  $\bar{s} = (O_S s_1 + O_P s_2 + O_T s_3 + O_Q s_4)/M$  is the weighted average score. The following theorem shows the convexity of the trend test.

**Lemma 2.** *Given the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ , the Cochran-Armitage test for trend  $z^2$  is a convex function of  $(O_{c_2}, O_{d_2})$ .*

*Proof.* See Appendix.

Suppose that the range of  $O_{c_2}$  is  $[l_{c_2}, u_{c_2}]$ , and the range of  $O_{d_2}$  is  $[l_{d_2}, u_{d_2}]$ . For any convex function, its maximum value is attained at one of the vertices of its convex domain [6]. Thus, from Lemmas 1 and 2, we have the following theorem.

**Theorem 1.** Given the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ , for chi-square test, G-test, and entropy-based test, the maximum value of  $\mathcal{F}_1(X_i X_j Y)$  is attained when  $O_{c_2} = l_{c_2}$  or  $O_{c_2} = u_{c_2}$ . The maximum value of  $\mathcal{F}_2(X_i X_j Y)$  is attained when  $O_{d_2} = l_{d_2}$  or  $O_{d_2} = u_{d_2}$ . The maximum value of Cochran-Armitage test  $z^2$  is attained when  $(O_{c_2}, O_{d_2})$  takes one of the four values in  $\{(l_{c_2}, l_{d_2}), (l_{c_2}, u_{d_2}), (u_{c_2}, l_{d_2}), (u_{c_2}, u_{d_2})\}$ .

Therefore, we can develop an upper bound of the two-locus test if we identify the range of  $O_{c_2}$  and  $O_{d_2}$ . For example, suppose that the value of vector  $(O_A, O_B, O_C, O_D, O_P, O_Q)$  is  $(6, 10, 10, 6, 7, 6)$ . In Figure 2, we plot function  $\chi_1^2(X_i X_j, Y)$ . The blue stars represent the values of  $\chi_1^2(X_i X_j, Y)$  when  $O_{c_2}$  takes different values. Clearly,  $\chi_1^2(X_i X_j, Y)$  is a convex function of  $O_{c_2}$ , and its upper bound is determined by the two end points of the range of  $O_{c_2}$ . Since  $O_{c_2}$  is always less than  $O_C$ , in this example, the default range of  $O_{c_2}$  is  $[0, O_C] = [0, 10]$ . Typically, the actual range of  $O_{c_2}$  is tighter, as indicated by the red dotted lines, which leads to a tighter upper bound of the test value. In the next section, by examining the contingency tables, we derive a set of constraints that determine the range of  $O_{c_2}$  and  $O_{d_2}$ .

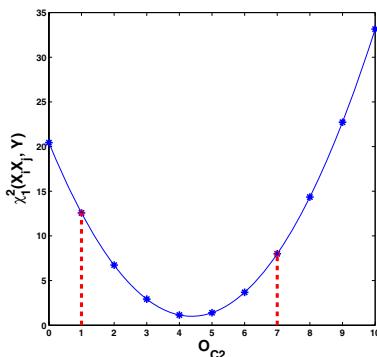


Fig. 2. Convexity Example

## 4 Constraints on Observed Values

From the contingency tables shown in Table I, we can develop a set of equations, as shown in Figure 3 at the left side of the arrow sign. Although there are 8 equations, the rank of the linear equation system is 6. We choose 6 linear equations to form a full rank system. The matrix multiplication form of these 6 equations is shown in Figure 3 at the right side of the arrow sign. The reason for choosing the 6 equations is two-fold. First, these 6 equations can be used to derive the range of  $O_{c_2}$  and  $O_{d_2}$ . Second, the values of  $O_A, O_B, O_C, O_D$  are determined by the single-locus contingency table in Table II(a).

$$\left\{ \begin{array}{l} O_{a_1} + O_{a_2} = O_A \\ O_{b_1} + O_{b_2} = O_B \\ O_{c_1} + O_{c_2} = O_C \\ O_{d_1} + O_{d_2} = O_D \\ O_{a_1} + O_{c_1} = O_S \\ O_{a_2} + O_{c_2} = O_P \\ O_{b_1} + O_{d_1} = O_T \\ O_{b_2} + O_{d_2} = O_Q \end{array} \right. \implies \left( \begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right) \left( \begin{array}{c} O_{a_1} \\ O_{a_2} \\ O_{b_1} \\ O_{b_2} \\ O_{c_1} \\ O_{c_2} \\ O_{d_1} \\ O_{d_2} \end{array} \right) = \left( \begin{array}{c} O_A \\ O_B \\ O_C \\ O_D \\ O_S \\ O_P \\ O_T \\ O_Q \end{array} \right)$$

Fig. 3. Linear equation system derived from contingency tables

$$\begin{pmatrix} O_{a_1} \\ O_{a_2} \\ O_{c_1} \\ O_{c_2} \end{pmatrix} = \begin{pmatrix} O_A - O_P \\ O_P \\ O_C \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} O_{c_2}, \text{ and } \begin{pmatrix} O_{b_1} \\ O_{b_2} \\ O_{d_1} \\ O_{d_2} \end{pmatrix} = \begin{pmatrix} O_B - O_Q \\ O_Q \\ O_D \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} O_{d_2}.$$

**Fig. 4.** Relations between observed values in the contingency table of two-locus test

The remaining two values,  $O_P$  and  $O_Q$ , only depend on the SNP-pair's genotype. It enables us to index the SNP-pairs by their  $(O_P, O_Q)$  values to effectively apply the upper bound. This will become clear when we present the algorithm in Section 5.

From these 6 equations, we obtain the relationships between the observed values shown in Figure 4. Since all observed values in the contingency table must be greater or equal to 0, the ranges of  $O_{c_2}$  and  $O_{d_2}$  are stated in Theorem 2.

**Theorem 2.** *Given the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ , the ranges of  $O_{c_2}$  and  $O_{d_2}$  are*

$$\begin{cases} \max\{0, O_P - O_A\} \leq O_{c_2} \leq \min\{O_P, O_C\}; \\ \max\{0, O_Q - O_B\} \leq O_{d_2} \leq \min\{O_Q, O_D\}. \end{cases}$$

Given  $O_A, O_B, O_C, O_D, O_P, O_Q$ , the values of  $O_{a_1}, O_{a_2}, O_{c_1}$  are determined by  $O_{c_2}$ , the values of  $O_{b_1}, O_{b_2}, O_{d_1}$  are determined by  $O_{d_2}$ . So all values in the contingency table for two-locus test in Table I(c) depend only on  $O_{c_2}$  and  $O_{d_2}$ . The maximum value,  $ub(\mathcal{T}(X_i X_j, Y))$ , is attained when  $O_{c_2}$  and  $O_{d_2}$  take the boundary values shown in Theorems 1 and 2. Continuing with the example in Figure 2, the value of  $(O_A, O_B, O_C, O_D, O_P, O_Q)$  is  $(6, 10, 10, 6, 7, 6)$ . From Theorem 2, the range of  $O_{c_2}$  is  $[1, 7]$ , as indicated by the red lines. The upper bound of  $\chi^2_1(X_i X_j, Y)$  is reached when  $O_{c_2} = 1$ .

Note that the upper bound value only depends on  $O_A, O_B, O_C, O_D, O_P, O_Q$ . This property allows us to group and index SNP-pairs by their genotypes so that the upper bound can effectively estimated and applied to prune the search space.

## 5 Applying the Upper Bound

Theorems 1 and 2 show that the upper bound value of the two-locus test  $\mathcal{T}(X_i X_j, Y)$  (for any one of the four tests discussed in Section 3) is determined by the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ . As shown in Table I, these values only depend on the contingency table for the single-locus test  $\mathcal{T}(X_i, Y)$  and the contingency table for the SNP-pair  $(X_i X_j)$ 's genotype. This allows us to group the SNP-pairs and index them by their genotypes. The idea of building such indexing structure has also been explored in [34][35]. For self-containment, in this section, we first discuss how to apply the upper bound to find the significant SNP-pairs. Then we show that a similar idea can be used to find the critical values  $\mathcal{T}_\alpha$  using permutation test.

<sup>2</sup> For entropy-based statistic, so far we have focused on the joint entropy  $-H(X_i X_j | Y)$ . Note that, given the values of  $O_A, O_B, O_C, O_D, O_P, O_Q$ , the upper bound for the mutual information  $I(X_i X_j, Y)$  can also be easily derived.

For every  $X_i$  ( $1 \leq i \leq N$ ), let  $AP(X_i) = \{(X_i X_j) | i + 1 \leq j \leq N\}$  be the SNP-pairs with  $X_i$  being the SNP of lower index value. We can index the SNP-pairs in  $AP(X_i)$  by their  $(O_P, O_Q)$  values in a 2D array, referred to as  $Array(X_i)$ . Note that  $O_P$  is the number of 1's in  $X_j$  when  $X_i$  takes value 0.  $O_Q$  is the number of 1's in  $X_j$  when  $X_i$  takes value 1.

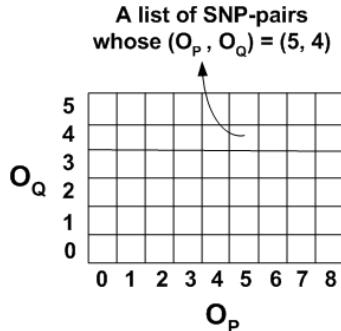


Fig. 5. Indexing SNP-pairs

5

It is obvious that for any SNP-pair  $(X_i X_j) \in AP(X_i)$ , if the upper bound value of the two-locus test is less than the critical value, i.e.,  $ub(\mathcal{T}(X_i X_j, Y)) < \mathcal{T}_\alpha$ , then this SNP-pair cannot be significant since its actual test value will also be less than the threshold. Only the SNP-pairs whose upper bound values are greater than the threshold need to be evaluated for their test values. We refer to such SNP-pairs as *candidates*.

Recall that from Theorems 1 and 2 the upper bound of two-locus test value is a constant for given  $O_A, O_B, O_C, O_D, O_P, O_Q$ . Given SNP  $X_i$  and phenotype  $Y$ , the values of  $O_A, O_B, O_C, O_D$  are fixed. For SNP-pairs  $(X_i X_j) \in AP(X_i)$ , once we index them by their  $(O_P, O_Q)$  values as shown in Figure 5, we can identify the candidate SNP-pairs by accessing the indexing structure: For each entry of the indexing structure, we calculate the upper bound value. If the upper bound value is greater than or equal to the critical value  $\mathcal{T}_\alpha$ , then all SNP-pairs indexed by this entry are candidates and subject to two-locus tests. The SNP-pairs whose upper bound values are less than the critical value are pruned without any additional test.

Suppose that there are  $m$  1's and  $(M - m)$  0's in SNP  $X_i$ . The maximum size of the indexing structure  $Array(X_i)$  is  $m(M - m)$ . Usually, the number of individuals  $M$  is much smaller than the number of SNPs  $N$ . Therefore, the number of entries in the indexing structure is also much smaller than  $N$ . Thus there must be a group of SNP-pairs indexed by the same entry. Since all SNP-pairs indexed by the same entry have the same upper bound value, the indexing structure enables us to calculate the upper bound value for this group of SNP-pairs together.

So far, we have discussed how to use the indexing structure and the upper bound to prune the search space to find significant SNP-pairs for a given critical value  $\mathcal{T}_\alpha$ . The problem of finding this critical value  $\mathcal{T}_\alpha$  is much more time consuming than finding the significant SNP-pairs since it involves large scale permutation test. The indexing structure  $Array(X_i)$  can be easily incorporated in the algorithm for permutation

For example, suppose that there are 13 individuals in the dataset. SNP  $X_i$  consists of 8 0's and 5 1's. Thus for the SNP-pairs in  $AP(X_i)$ , the possible values of  $O_P$  are  $\{0, 1, 2, \dots, 8\}$ . The possible values of  $O_Q$  are  $\{0, 1, 2, \dots, 5\}$ . Figure 5 shows the  $6 \times 9$  array,  $Array(X_i)$ , whose entries represent the possible values of  $(O_P, O_Q)$  for the SNP-pairs  $(X_i X_j)$  in  $AP(X_i)$ . Each entry of the array is a pointer to the SNP-pairs  $(X_i X_j)$  having the corresponding  $(O_P, O_Q)$  values. For example, all SNP-pairs in  $AP(X_i)$  whose  $(O_P, O_Q)$  value is  $(5, 4)$  are indexed by the entry  $(5, 4)$  in Figure 5.

test. The key property is that the indexing structure  $\text{Array}(X_i)$  is independent of the phenotype. Once  $\text{Array}(X_i)$  is built, it can be reused in all permutations. Therefore, building the indexing structure  $\text{Array}(X_i)$  is only a one time cost. The permutation procedure is similar to that of finding significant SNP-pairs. The only difference is that the threshold used to prune the search space is a dynamically updated critical value found by the algorithm so far. The overall procedure of our algorithm COE is similar to that in [34][35]. An important difference is that COE utilizes the convexity of statistical tests and is applicable to all four statistics. We omit the pseudo code of the algorithm in the main body of the paper. Please refer to the Appendix for further details.

*Property 1.* The indexing structure  $\text{Array}(X_i)$  can be applied in computing the upper bound value for all four statistical tests, i.e., chi-square test, G-test, mutual information, and trend test.

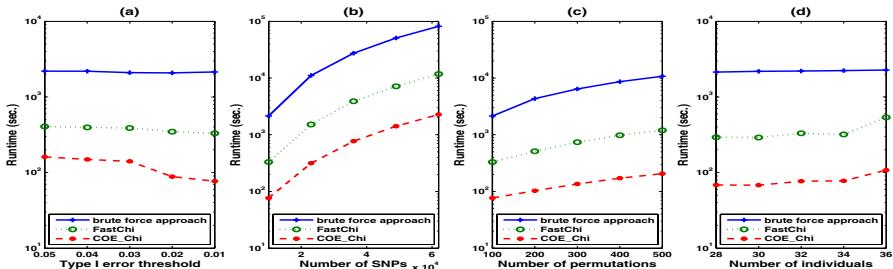
The correctness of Property I relies on the fact that the upper bound is always a function of  $O_A, O_B, O_C, O_D, O_P, O_Q$ , regardless of the choice of test. All SNP-pairs having the same  $(O_P, O_Q)$  value will always share a common upper bound. This property shows that there is no need to rebuild the indexing structure if the users want to switch between different tests. It only needs to be built once and retrieved for later use.

The time complexity of COE for permutation test is  $O(N^2M + KNM^2 + CM)$ , where  $N$  is the number of SNPs,  $M$  is the number of individuals,  $K$  is the number of permutations, and  $C$  is the number of candidates reported by the algorithm. Experimental results show that  $C$  is only a very small portion of all SNP-pairs. A brute force approach has time complexity  $O(KN^2M)$ . Note that  $N$  is the dominant factor, since  $M \ll N$ . The space complexity of COE is linear to the size of the dataset. The derivation of the complexity is similar to that in [34][35] and can be found in the Appendix.

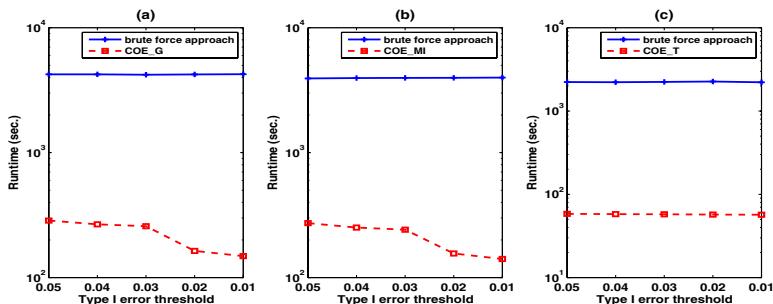
## 6 Experimental Results

In this section, we present extensive experimental results on evaluating the performance of the COE algorithm. COE is implemented in C++. We use COE\_Chi, COE\_G, COE\_MI, COE\_T to represent the COE implementation for the chi-square test, G-test, mutual information, and trend test respectively. The experiments are performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

**Dataset and Experimental Settings:** The SNP dataset is extracted from a set of combined SNPs from the 140k Broad/MIT mouse dataset [32] and 10k GNF [2] mouse dataset. This merged dataset has 156,525 SNPs for 71 mouse strains. The missing values in the dataset are imputed using NPUTE [23]. The phenotypes used in the experiments are simulated binary variables which contain half cases and half controls. This is common in practice, where the numbers of cases and controls tend to be balanced. If not otherwise specified, the default settings of the experiments are as follows: #individuals = 32, #SNPs=10,000, #permutations=100. There are 62,876 unique SNPs for these 32 strains.



**Fig. 6.** Performance comparison of the brute force approach, FastChi, and COE\_Chia



**Fig. 7.** Performance comparison of the brute force approach, COE\_G, COE\_MI, and COE\_T

## 6.1 Performance Comparison

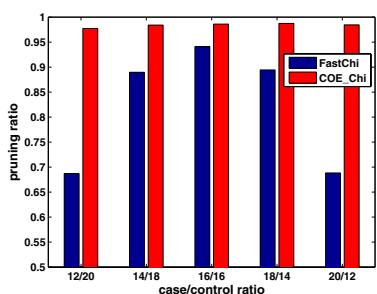
Figure 6 shows the runtime comparison of the brute force two-locus chi-square test, the FastChi algorithm [35], and the COE implementation of chi-square test, COE\_Chia, in permutation test under various settings. Note that the runtime reported in this section are based on the complete executions of all methods including the one time cost for building the indexing structures. Figure 6(a) shows the comparison when the Type I error threshold varies. The y-axis is in logarithm scale. COE\_Chia improves the efficiency of two-locus epistasis detection by one order of magnitude over FastChi (which was specifically designed for two-locus chi-square test), and two orders of magnitude over the brute force approach. Figure 6(b), (c), and (d) demonstrate similar performance improvements of COE\_Chia over the other two approaches when varying number of SNPs, number of permutations, and number of individuals respectively. This is consistent with the pruning effect of the upper bounds which will be presented later.

Figure 7(a) shows the runtime comparison between the brute force two-locus G-test and COE\_G when varying the type I error threshold. The runtime of COE\_G dramatically reduces as the type I error threshold decreases. COE\_G is one to two orders magnitudes faster than the brute force approach. Similar performance improvement can also be observed for COE\_MI and COE\_T in Figures 7(b) and 7(c). Note that for these three tests, we also have similar results when varying other settings. Due to space limitation, we omit these results here.

**Table 2.** Pruning effects of FastChi and COE on four different statistics

		FastChi	COE.Chi	COE.G	COE.MI	COE.T
$\alpha$	0.05	87.59%	95.70%	95.84%	95.80%	99.90%
	0.04	87.98%	96.11%	96.23%	96.23%	99.92%
	0.03	88.12%	96.32%	96.40%	96.43%	99.93%
	0.02	89.43%	98.18%	98.31%	98.28%	99.96%
	0.01	90.03%	98.59%	98.65%	98.62%	99.98%
# SNPs	10k	90.03%	98.59%	98.65%	98.62%	99.98%
	23k	91.52%	99.08%	99.50%	99.13%	99.99%
	36k	91.39%	99.03%	99.43%	99.09%	99.99%
	49k	91.39%	99.04%	99.43%	99.09%	99.99%
	62k	91.22%	99.04%	99.43%	99.09%	99.99%
# Perm.	100	90.03%	98.59%	98.65%	98.62%	99.98%
	200	91.79%	99.03%	99.42%	99.08%	99.99%
	300	91.90%	99.04%	99.43%	99.09%	99.99%
	400	91.91%	99.04%	99.43%	99.09%	99.99%
	500	91.99%	99.04%	99.43%	99.09%	99.99%
# Indiv.	28	91.05%	98.77%	99.83%	99.06%	99.99%
	30	91.23%	98.83%	98.94%	99.06%	99.98%
	32	90.03%	98.59%	99.65%	98.62%	99.98%
	34	91.54%	98.80%	99.74%	98.84%	99.97%
	36	89.08%	97.94%	95.74%	93.55%	99.94%

## 6.2 Pruning Power of the Upper Bound

**Fig. 8.** FastChi v.s. COE.Chi

of COE.Chi demonstrates the strength of convex optimization in finding the maximum values. In addition, the upper bound derived by applying convex optimization is not only more effective, but also more robust for unbalanced datasets.

Figure 8 shows the pruning effectiveness of FastChi and COE.Chi when the ratio of case/control varies. It is clear that the pruning power of FastChi is weakened when the case/control ratio becomes unbalanced. Therefore, FastChi is not very effective for unbalanced case-control datasets. In contrast, COE.Chi maintains a steady pruning percentage under different case/control ratios. Thus it remains effective for the unbalanced datasets. Similar behaviors of COE are also observed in the other three statistical tests.

Table 2 shows the percentage of SNP-pairs pruned under different experimental settings for the four statistical tests. We also include the pruning ratio of FastChi in the table for comparison. From the table, most of the SNP-pairs are pruned by COE. Note that COE.Chi has more pruning power than FastChi. The upper bound used in FastChi is derived by loosening the observed values for the events in two-locus test without using the convexity property. The tighter upper bound

## 7 Discussion

Genome-wide epistasis detection is computationally demanding due to the large number of SNPs. As a golden standard for proper family-wise error controlling, the permutation test dramatically increases the computation burden. In this paper, we present a general approach COE that support genome-wide disease association study with a wide range of statistics composing of convex terms. We use four commonly used statistics as prototypes: chi-square test, G-test, entropy-based test, and Cochran-Armitage trend test. COE guarantees optimal solution and performs two orders of magnitude faster than brute force approaches.

The performance gain is attributed to two main contributions of COE. The first is a tight upper bound estimated using convex optimization. It has much higher pruning power than any upper bounds used in previous methods such as FastChi. As a result, COE\_Chi is an order of magnitude faster than FastChi. Moreover, COE serves as a general platform for two-locus epistasis detection, which eliminates the need of designing specific pruning methods for different statistical tests. Recall that any observed value in a two-locus test is a function of  $O_{c_2}$  and  $O_{d_2}$  for given  $O_A, O_B, O_C, O_D, O_P, O_Q$ . Let  $x = O_{c_2}$  and  $y = O_{d_2}$ . A wide spectrum of functions of  $x$  and  $y$  are convex [6], which include all linear and affine functions on  $x$  and/or  $y$ , exponential terms  $e^{ax}$  ( $a \in \mathbb{R}$ ), powers  $x^a$  ( $a \geq 1$  or  $a \leq 0$ ), negative logarithm  $-\log x$ , maximum  $\max\{x, y\}$ . In addition, many operations preserve convexity. For example, if  $f(x, y)$  is a convex function, and  $g(x, y)$  is an affine mapping, then  $f(g(x, y))$  is also a convex function. Please refer to [6] for further details.

The second source of performance improvement is from indexing SNP-pairs by their genotypes. Applying this indexing structure, we can compute a common upper bound value for each group. The indexing structure is independent of the phenotype permutations and the choice of statistical test . We can eliminate redundant computation in permutation test and provide the flexibility of supporting multiple statistical tests on the fly.

In this paper, we focus on binary SNPs and case-control phenotypes. The principle is also applicable to the heterozygous case, where SNPs are encoded using  $\{0, 1, 2\}$ , and to evaluate quantitative phenotypes, where phenotypes are continuous variables. We will investigate these two cases in our future work.

## References

1. [http://www.fnih.org/GAIN2/home\\_new.shtml](http://www.fnih.org/GAIN2/home_new.shtml)
2. <http://www.gnf.org/>
3. <http://www.jax.org/>
4. Balding, D.J.: A tutorial on statistical methods for population association studies. *Nature Reviews Genetics* 7(10), 781–791 (2006)
5. Bohringer, S., Hardt, C., Mitterski, B., Steland, A., Epplen, J.T.: Multilocus statistics to uncover epistasis and heterogeneity in complex diseases: revisiting a set of multiple sclerosis data. *European Journal of Human Genetics* 11, 573–584 (2003)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

7. Carlborg, O., Andersson, L., Kinghorn, B.: The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. *Genetics* 155, 2003–2010 (2000)
8. Carlson, C.S., Eberle, M.A., Kruglyak, L., Nickerson, D.A.: Mapping complex disease loci in whole-genome association studies. *Nature* 429, 446–452 (2004)
9. Chi, P.B., et al.: Comparison of snp tagging methods using empirical data: association study of 713 snps on chromosome 12q14.3-12q24.21 for asthma and total serum ige in an african caribbean population. *Genet. Epidemiol.* 30(7), 609–619 (2006)
10. Cordell, H.J.: Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Human Molecular Genetics* 11(20), 2463–2468 (2002)
11. Doerge, R.W.: Multifactorial genetics: Mapping and analysis of quantitative trait loci in experimental populations. *Nature Reviews Genetics* 3, 43–52 (2002)
12. Dong, C., et al.: Exploration of gene–gene interaction effects using entropy-based methods. *European Journal of Human Genetics* 16, 229–235 (2008)
13. Erlichman, C., Sargent, D.J.: New treatment options for colorectal cancer. *N. Engl. J. Med.* 351, 391–392 (2004)
14. Evans, D.M., Marchini, J., Morris, A.P., Cardon, L.R.: Two-stage two-locus models in genome-wide association. *PLoS Genet.* 2, e157 (2006)
15. Halperin, E., Kimmel, G., Shamir, R.: Tag snp selection in genotype data for maximizing snp prediction accuracy. In: Proc. ISMB (2005)
16. Herbert, A., et al.: A common genetic variant is associated with adult and childhood obesity. *Science* 312, 279–284 (2006)
17. Hoh, J., Ott, J.: Mathematical multi-locus approaches to localizing complex human trait genes. *Nature Reviews Genetics* 4, 701–709 (2003)
18. Kirman, I., Huang, E.H., Whelan, R.L.: B cell response to tumor antigens is associated with depletion of b progenitors in murine colocalcarcinoma. *Surgery* 135, 313–318 (2004)
19. Nelson, M.R., Kardia, S.L., Ferrell, R.E., Sing, C.F.: A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation. *Genome Research* 11, 458–470 (2001)
20. Ozaki, K., et al.: Functional snps in the lymphotoxin-alpha gene that are associated with susceptibility to myocardial infarction. *Nat. Genet.* 32, 650–654 (2002)
21. Pagano, M., Gauvreau, K.: Principles of Biostatistics. Duxbury Press, Pacific Grove (2000)
22. Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *American Journal of Human Genetics* 69, 138–147 (2001)
23. Roberts, A., McMillan, L., Wang, W., Parker, J., Rusyn, I., Threadgill, D.: Inferring missing genotypes in large snp panels using fast nearest-neighbor searches over sliding windows. In: Proc. ISMB (2007)
24. Roses, A.: The genome era begins. *Nat. Genet.* 33(suppl. 2), 217 (2003)
25. Ruivenkamp, C.A., Csikos, T., Klous, A.M., van Wezel, T., Demant, P.: Five new mouse susceptibility to colon cancer loci, scc11-scc15. *Oncogene* 22, 7258–7260 (2003)
26. Saxena, R., et al.: Genome-wide association analysis identifies loci for type 2 diabetes and triglyceride levels. *Science* 316, 1331–1336 (2007)
27. Scuteri, A., et al.: Genome-wide association scan shows genetic variants in the fto gene are associated with obesity-related traits. *PLoS Genet.* 3(7) (2007)
28. Sebastiani, P., Lazarus, R., Weiss, S.T., Kunkel, L.M., Kohane, I.S., Ramoni, M.F.: Minimal haplotype tagging. *Proc. Natl. Acad. Sci. USA* 100(17), 9900–9905 (2003)
29. Segré, D., DeLuna, A., Church, G.M., Kishony, R.: Modular epistasis in yeast metabolism. *Nat. Genet.* 37, 77–83 (2005)
30. Storey, J., Akey, J., Kruglyak, L.: Multiple locus linkage analysis of genomewide expression in yeast. *PLoS Biology* 8, e267 (2005)

31. Thomas, D.C.: Statistical methods in genetic epidemiology. Oxford University Press, Oxford (2004)
32. Wade, C.M., Daly, M.J.: Genetic variation in laboratory mice. Nat. Genet. 37(7), 1175–1180 (2005)
33. Weedon, M.N., et al.: A common variant of hmga2 is associated with adult and childhood height in the general population. Nat. Genet. 39, 1245–1250 (2007)
34. Zhang, X., Zou, F., Wang, W.: Fastanova: an efficient algorithm for genome-wide association study. In: KDD (2008)
35. Zhang, X., Zou, F., Wang, W.: FastChi: an efficient algorithm for analyzing gene-gene interactions. In: PSB (2009)
36. Zhao, J., Boerwinkle, E., Xiong, M.: An entropy-based statistic for genomewide association studies. Am. J. Hum. Genet. 77, 27–40 (2005)

## Appendix

### Proof of Lemma 1 and Lemma 2

*Proof.* We first show that  $\chi_1^2(X_iX_j, Y)$  is a convex function of  $O_{c_2}$ . Recall that

$$\chi_1^2(X_iX_j, Y) = \sum_{\text{event} \in \{a_1, a_2, c_1, c_2\}} \frac{(O_{\text{event}} - E_{\text{event}})^2}{E_{\text{event}}}.$$

For fixed  $O_A, O_B, O_C, O_D, O_P, O_Q$ , we know that the expected values of the four events are constants:

$$\begin{cases} E_{a_1} = \frac{O_S(O_A + O_B)}{M} = \frac{(O_A + O_C - O_P)(O_A + O_B)}{M} \\ E_{a_2} = \frac{O_P(O_A + O_B)}{M} \\ E_{c_1} = \frac{O_S(O_C + O_D)}{M} = \frac{(O_A + O_C - O_P)(O_C + O_D)}{M} \\ E_{c_2} = \frac{O_P(O_C + O_D)}{M} \end{cases}$$

From the relations between the observed values of the events in two-locus test (as shown in Figure 4), we have that  $O_{a_1}, O_{a_2}, O_{c_1}$  are linear functions of  $O_{c_2}$ <sup>3</sup>. So  $\chi_1^2(X_iX_j, Y)$  is a positive quadratic function of  $O_{c_2}$ . Thus  $\chi_1^2(X_iX_j, Y)$  is a convex function of  $O_{c_2}$ .

Next, we show that

$$G_1(X_iX_j, Y) = \sum_{\text{event} \in \{a_1, a_2, c_1, c_2\}} O_{\text{event}} \cdot \ln \frac{O_{\text{event}}}{E_{\text{event}}}$$

is a convex function of  $O_{c_2}$ . From previous result, for fixed  $O_A, O_B, O_C, O_D, O_P, O_Q$ , the expected values of the four events  $\{a_1, a_2, c_1, c_2\}$  are constants, and  $O_{a_1}, O_{a_2}, O_{c_1}$  are linear functions of  $O_{c_2}$ . Thus  $G_1(X_iX_j, Y)$  is a function of  $O_{c_2}$ . To prove the convexity of  $G_1(X_iX_j, Y)$ , it suffices to show that the second derivative  $\nabla^2 G_1(X_iX_j, Y)$

<sup>3</sup> Note that, although these relations are presented after shown the convexity of the statistics, it is easy to see that the derived relations are independent of whether the statistics are convex.

$= \frac{\partial^2 G_1(X_i X_j, Y)}{\partial O_{c_2}^2}$  is nonnegative. We show this is the case for the component of event  $a_2$ :

$$\nabla^2(O_{a_2} \cdot \ln \frac{O_{a_2}}{E_{a_2}}) = \nabla^2((O_P - O_{c_2}) \cdot \ln \frac{O_P - O_{c_2}}{E_{a_2}}) = \frac{1}{O_P - O_{c_2}} \geq 0.$$

Similarly, we can prove that the second derivative of other components are nonnegative. Therefore,  $G_1(X_i X_j, Y)$  is a convex function of  $O_{c_2}$ .

Following the similar idea, i.e., by showing the second derivative of  $-H(X_i X_j Y)$  is nonnegative, we can prove that  $-H_1(X_i X_j Y)$  is a convex function of  $O_{c_2}$ .

Thus we have shown the  $\mathcal{T}_1(X_i X_j Y)$  is a convex function of  $O_{c_2}$ . The convexity  $\mathcal{T}_2(X_i X_j Y)$  can be proven in a similar way.

We now prove that the Cochran-Armitage trend test is a convex function of  $(O_{c_2}, O_{d_2})$ . Observe that the  $O_{c_1}$  is a linear function of  $O_{c_2}$ , and  $O_{d_1}$  is a linear function of  $O_{d_2}$ . The values of  $p, s_i$  ( $i \in \{1, 2, 3, 4\}$ ), and  $\bar{s}$  are fixed. Thus the trend statistic  $z^2$  is a quadratic function of the two variables  $(O_{c_2}, O_{d_2})$ . This completes the proof.  $\square$

### Pseudo code of COE for permutation test

---

#### Algorithm 1. COE for permutation test

---

**Input:** SNPs  $X' = \{X_1, X_2, \dots, X_N\}$ , phenotype permutations  $Y' = \{Y_1, Y_2, \dots, Y_K\}$ , and the Type I error  $\alpha$ .  
**Output:** the critical value  $\mathcal{T}_\alpha$ .

```

1  $Tlist \leftarrow \alpha K$  dummy phenotype permutations with test value 0 ;
2  $\mathcal{T}_\alpha = 0$ ;
3 for every  $X_i \in X'$ , do
4   index  $(X_i X_j) \in AP(X_i)$  by  $Array(X_i)$ ;
5   for every  $Y_k \in Y'$ , do
6     access  $Array(X_i)$  to find the candidate SNP-pairs and store them in
       $Cand(X_i, Y_k)$ ;
7     for every  $(X_i X_j) \in Cand(X_i, Y_k)$  do
8       if  $\mathcal{T}(X_i X_j, Y_k) \geq \mathcal{T}_\alpha$  then
9         update  $Tlist$ ;
10         $\mathcal{T}_\alpha$  = the smallest test value in  $Tlist$ ;
11      end
12    end
13  end
14 end
15 return  $\mathcal{T}_\alpha$ .

```

---

Algorithm 1 describes our COE algorithm for finding critical values in two-locus epistasis detection using permutation test. The algorithm for finding significant SNP-pairs is similar. The overall process is similar to that in [34][35]. The goal is to find the critical value  $\mathcal{T}_\alpha$ , which is the  $\alpha K$ -th largest value in  $\{\mathcal{T}_{Y_k} | Y_k \in Y'\}$ . We use  $Tlist$  to keep the  $\alpha K$  phenotype permutations having the largest test values found by the algorithm so far. Initially,  $Tlist$  contains  $\alpha K$  dummy permutations with test values 0.

The smallest test value in  $Tlist$ , initially 0, is used as the threshold to prune the SNP-pairs. For each  $X_i$ , the algorithm first builds the indexing structure  $Array(X_i)$  for the SNP-pairs  $(X_i X_j) \in AP(X_i)$ . Then it accesses  $Array(X_i)$  to find the set of candidates  $Cand(X_i, Y_k)$  for every phenotype permutation. Two-locus tests are performed on these candidates to get their test values. If a candidate's test value is greater than the current threshold, then  $Tlist$  is updated: If the candidate's phenotype  $Y_k$  is not in the  $Tlist$ , then the phenotype in  $Tlist$  having the smallest test value is replaced by  $Y_k$ . If the candidate's phenotype  $Y_k$  is already in  $Tlist$ , we only need to update its corresponding test value to be the maximum value found for the phenotype so far. The threshold is also updated to be the smallest test value in  $Tlist$ .

**Time complexity:** For each  $X_i$ , the algorithm needs to index  $(X_i X_j)$  in  $AP(X_i)$ . The complexity to build the indexing structure for all SNPs is  $O(N(N - 1)M/2)$ . The worst case for accessing all  $Array(X_i)$  for all permutations is  $O(KNM^2)$ . Let  $C = \sum_{i,k} |Cand(X_i, Y_k)|$  represent the total number of candidates. The overall time complexity of our algorithm is  $O(N(N - 1)M/2) + O(KNM^2) + O(CM) = O(N^2M + KNM^2 + CM)$ .

**Space complexity:** The total number of variables in the dataset, including the SNPs and the phenotype permutations, is  $N + K$ . The maximum space of the indexing structure  $Array(X_i)$  is  $O(M^2 + N)$ . For each SNP  $X_i$ , our algorithm only needs to access one indexing structure,  $Array(X_i)$ , for all permutations. Once the evaluation process for  $X_i$  is done for all permutations,  $Array(X_i)$  can be cleared from the memory. Therefore, the space complexity of COE is  $O((N + K)M) + O(M^2 + N) = O((N + K + M)M + N)$ . Since  $M \ll N$ , the space complexity is linear to the dataset size.

# Overlapping Pools for High Throughput Targeted Resequencing

Snehit Prabhu\* and Itsik Pe'er\*

Department of Computer Science  
Columbia University, New York  
`{snehitp,itsik}@cs.columbia.edu`

**Abstract.** Resequencing genomic DNA from pools of individuals is an effective strategy to detect new variants in targeted regions and compare them between cases and controls. There are numerous ways to assign individuals to the pools on which they are to be sequenced. The naïve, disjoint pooling scheme (many individuals to one pool) in predominant use today, offers insight into allele frequencies, but does not offer the identity of an allele carrier. We present a framework for overlapping pool design, where each individual sample is resequenced in several pools (many individuals to many pools). Upon discovering a variant, the set of pools where this variant is observed reveals the identity of its carrier. We formalize the mathematical framework for such pool designs, and list the requirements from such designs. Next, we build on the theory of error-correcting codes to design arrangements that overcome pitfalls of pooled sequencing. Specifically, three practical concerns of low coverage sequencing are investigated: (1) False positives due to errors introduced during amplification and sequencing; (2) False negatives due to undersampling particular alleles aggravated by non-uniform coverage; and consequently (3) Ambiguous identification of individual carriers in the presence of errors. We show that in practical parameters of resequencing studies, our designs guarantee high probability of unambiguous singleton carrier identification, while maintaining the features of naïve pools in terms of sensitivity, specificity, and the ability to estimate allele frequencies. We demonstrate the ability of our designs by extracting rare variations on pooled short read data of 12 individuals from the 1000 Genome Pilot 3 project.

---

\* S.P. was supported in part by NSF CCF 0829882 ; I.P. was supported in part by NIH 5 U54 CA121852.

# Deep Sequencing of a Genetically Heterogeneous Sample: Local Haplotype Reconstruction and Read Error Correction

Osvaldo Zagordi<sup>1</sup>, Lukas Geyrhofer<sup>1</sup>, Volker Roth<sup>2</sup>, and Niko Beerenwinkel<sup>1</sup>

<sup>1</sup> Department of Biosystems Science and Engineering, ETH Zurich, Basel,  
Switzerland

<sup>2</sup> Department of Computer Science, University of Basel, Switzerland

**Abstract.** We present a computational method for analyzing deep sequencing data obtained from a genetically diverse sample. The set of reads obtained from a deep sequencing experiment represents a statistical sample of the underlying population. We develop a generative probabilistic model for assigning observed reads to unobserved haplotypes in the presence of sequencing errors. This clustering problem is solved in a Bayesian fashion using the Dirichlet process mixture to define a prior distribution on the unknown number of haplotypes in the mixture. We devise a Gibbs sampler for sampling from the joint posterior distribution of haplotype sequences, assignment of reads to haplotypes, and error rate of the sequencing process to obtain estimates of the local haplotype structure of the population. The method is evaluated on simulated data and on experimental deep sequencing data obtained from HIV samples.

## 1 Introduction

The recent introduction of a new generation of high-throughput DNA sequencing technologies, known as deep sequencing, has opened up new experimental approaches to whole-genome sequencing, metagenomics, transcriptomics, epigenetics, and gene regulation analysis. As compared to traditional Sanger sequencing, deep sequencing can produce many more DNA bases in a single run, but the sequence reads are typically shorter and more error-prone. Thus, many statistical and computational challenges arise in analyzing and interpreting deep sequencing data [1,2,3].

Here we present a computational method for analyzing deep sequencing data obtained from a genetically diverse sample and for quantifying the genetic variation in the mixture. This problem is frequent as most tissues are built from different cell types and they consist of cells in different epigenetic states. Genetic diversity is of particular interest in cancer cells of a tumor and in pathogen populations, because it often drives disease progression. For example, HIV exists in each single infected patient as a quasispecies, a population of evolutionary related, diverse viral strains. Viral genetic diversity is associated with diseases progression, it complicates vaccine design, and it limits the efficacy of antiretroviral therapy [4,5,6].

When a mixed sample is sequenced by means of the traditional Sanger method, the output is a consensus sequence of the population. However, this approach misses all SNPs with a frequency below 20% and the phasing of multiple SNPs cannot be resolved. In order to address these limitations by Sanger sequencing, one needs to sequence individual clones from the population, which is impractical for most applications because of the higher work effort and costs.

Deep sequencing has the potential to resolve the genetic variation in a sample at unprecedented detail by directly sequencing the mixture of clones [7,8]. With this technique one can draw a very large number of reads from the population of interest. We will use this statistical sample in order to make inference about the structure of the underlying population. Working in sequence windows that are covered completely by subsets of the aligned reads, our goal is to locally infer haplotypes and their frequencies in the population. In order to deal with the inherent error rate of deep sequencing, we develop a probabilistic model for assigning reads to haplotypes and for reconstructing the most likely haplotype composition.

The local haplotype structure is an important source of information about the population that cannot be obtained easily with traditional methods. Since reads that are grouped together are likely to originate from the same haplotype, we can correct sequencing errors by removing the remaining within-cluster differences. Thus, we separate signals of biological variation from technical noise. Furthermore, the number of reads per cluster provides an estimate of the haplotype frequencies. Our method allows for quantifying the allele frequencies at any single sequence position and also at DNA segments of length on the order of the read length. Hence, the prevalence of mutational patterns in regions of several hundreds base pairs can be assessed. For example, the 454/Roche deep sequencing platform produces 250bp long reads and a recent upgrade has pushed this limit to 400 bp. Finally, the local haplotype structure imposes constraints on the global haplotype structure and can therefore inform global haplotype reconstruction methods [9,10,11].

We develop a generative probabilistic model for clustering reads based on the Dirichlet process mixture (DPM). In a Bayesian fashion, the DPM defines a prior distribution that captures the uncertainty in the number of haplotypes, which will eventually be determined by the data. The model parameters include the DNA sequences of the haplotypes, the assignment of reads to haplotypes, and the error rate of the sequencing process. We devise a Gibbs sampler for sampling from the posterior distribution of the parameters given the observed reads.

Because we are dealing with mixed samples, the number of haplotypes in the underlying population is inherently unknown. For example, intra-host HIV populations can be almost clonal during primary infection and are often very diverse at late infection stages. The unknown model complexity is estimated by the DPM, which is governed by a single hyperparameter that controls the probability of creating new clusters. In a related approach, Xing et al. [12] have applied the DPM to define a prior on the number of ancestors in a population of diploids. However, haplotype reconstruction from SNPs in diploid organisms is different from the estimation problem addressed here, because the generation of genotypes from

haplotypes is considered error free, and every genotype consists of exactly two haplotypes. Moreover, the number of observations is typically much smaller.

Eriksson et al. [10] have used a  $k$ -means algorithm for read clustering in combination with a statistical test to determine the number  $k$  of haplotypes. The main limitation of this approach is that a hard estimate of the error rate has to be assumed for the hypothesis test. By contrast, the conceptual advantage of the DPM approach developed here is to define a probability distribution based on the basic features of the sequencing process that expresses our uncertainty with respect to the true error rate.

In the Methods section, we formally define the model and derive equations for the Gibbs sampler. In the Results section, we present several computational results to validate and test our method. We use both simulated data and real data obtained from HIV samples using the 454/Roche FLX technology. Our results indicate that to a large extent the local haplotype structure of a mixed HIV sample can be recovered from deep sequencing data. We investigate the performance of our method as a function of coverage, pairwise distances between haplotypes, and relative frequencies of haplotypes. The described method is implemented in the software package *ShoRAH*, which is freely available at <http://www.cbg.ethz.ch/software/shorah>.

## 2 Methods

The main difficulty in inferring the genetic diversity from a set of reads is to distinguish biological variation from technical noise. If a read shows a mutation with respect to the consensus sequence, two possibilities must be considered: (1) a sequence with this particular mutation is actually present in the sample, or (2) a technical error occurred in the sequencing process. Our approach to error correction relies on the assumption that, in sequence space, reads tend to cluster around the true haplotypes, with a distribution depending on the error process, while haplotypes are separated by their true evolutionary distance.

Reads are aligned to a reference sequence in order to identify those that overlap with a given window. In this work, we do not face the problem of mapping reads to a very long reference genome. In fact, since we are always interested in a specific region (thousands of bases rather than billions), the computational time to perform pairwise alignments is not prohibitive. In order to build a multiple alignment from the set of pairwise alignments, we simply add gaps from pairwise alignments to all other alignments overlapping it. In other words, for every position where at least one read has an inserted base, a gap is inserted into all other reads overlapping that position. This approach certainly overestimates the number of insertions, and entrusts the removing of technical ones to the following error correction step. Other, more tailored approaches can be pursued, and are object of separate research [13].

### Dirichlet process mixture

There exist many efficient general purpose clustering algorithms, all of which face the problem of choosing the right number of clusters, which is in general

not an easy task. This model selection problem is key to read clustering, because a sample can be anything from clonal to very diverse, so that in principle there is no *a priori* knowledge on its phylogenetic structure. For this reason, we use a statistical Bayesian method, the Dirichlet process mixture (DPM), to capture the uncertainty in the number of clusters and their structure. When a set of observations comes from an unknown distribution it can be useful to model it as a mixture of simpler distributions. The DPM introduces a prior on mixing proportions that leads to few dominating classes and is controlled by a single hyperparameter,  $\alpha$ . This prior is expressed in the following equation for the probability that  $c_i = c$ , i.e. that observation  $i$  is assigned to class  $c$ :

$$p(c_i = c | c_j : j \neq i) = \begin{cases} \frac{n_{\setminus i,c}}{n-1+\alpha} & \text{if class } c \text{ is already populated,} \\ \frac{\alpha}{n-1+\alpha} & \text{if a new class is instantiated,} \end{cases} \quad (1)$$

where  $n_{\setminus i,c}$  denotes the number of observations in class  $c$  except observation  $i$  itself, and  $n$  the total number of observations.

The above equation states that observation  $i$  is assigned to class  $c$  with a probability proportional to the number of observations currently in that class, or, with probability proportional to  $\alpha$ , instantiates a new class populated initially by observation  $i$ . The hyperparameter  $\alpha$  controls the probability of creating new classes. In our application, the classes, or clusters, will be the haplotypes and the observations will be the reads.

Equation (1) constitutes a prior on the assignment of observations to classes that has to be embedded in a full probabilistic model. In order to do so, we model the generation of reads from different haplotypes in a single experiment, accounting for errors in the sequencing process.

Each read  $r_i$  comes from  $h_k$ , one of the  $K$  haplotypes, in which case the assignment  $c_i$  takes the value  $c_i = k$ . The  $j$ -th base  $r_{i,j}$  of read  $i$  is sampled from the corresponding  $j$ -th base  $h_{k,j}$  of the haplotype according to a parameter  $\theta$ , the probability that the base is drawn without error. The model assumes that bases at different positions are independent and subject to the same error rate.

Under this model, the probability of a set of reads  $\mathbf{r} = \{r_1, \dots, r_n\}$ , given their assignments  $\mathbf{c} = \{c_1, \dots, c_n\}$  to the haplotypes  $\mathbf{h} = \{h_1, \dots, h_K\}$  is given by

$$p(\mathbf{r} | \mathbf{c}, \mathbf{h}) = \prod_{k=1}^K \theta^{m_k} \left( \frac{1-\theta}{|B|-1} \right)^{m'_k}, \quad (2)$$

with

$$m_k = \sum_{i=1}^n \sum_{j=1}^J \mathbb{I}(r_{i,j} = h_{k,j}) \mathbb{I}(c_i = k), \quad (3a)$$

$$m'_k = \sum_{i=1}^n \sum_{j=1}^J \mathbb{I}(r_{i,j} \neq h_{k,j}) \mathbb{I}(c_i = k), \quad (3b)$$

where  $J$  is the window length (i.e. the length of the reads and of the haplotypes),  $\mathbb{I}$  is the indicator function, and  $B$  the alphabet of the bases.

A starting point for the prior on the haplotypes might be a uniform distribution of all possible sequences of length  $J$ . From a computational point of view, it is more convenient to assume that all haplotypes originate from a reference genome through a mutation process similar to the error process that generates the reads from the haplotypes, but with a different parameter  $\gamma$ . This assumption aims at restricting the volume of sequence space that needs to be sampled in order to obtain reliable estimates of the posterior distribution, thus considerably speeding up our sampling procedure. The reference genome can be, for example, the consensus sequence of all reads, or any other genome sequence of the organism under consideration. We regard it as a prior because, even when its estimate is computed from the data, it remains fixed during the sampling.

### Gibbs sampler

Several algorithms have been proposed to sample the posterior distribution of a DPM [14]. We show here how to implement a Markov chain Monte Carlo algorithm to perform Gibbs sampling of the posterior DPM distribution under our model. The Gibbs sampler is a method for sampling from a distribution by drawing a value for each random variable from the conditional distribution of that variable given the values of all the others, and iterating over all variables. In our model we sample the following variables: the assignment variables  $c_i$ , the haplotype bases  $h_{k,j}$ , the error parameter  $\theta$ , the mutation parameter  $\gamma$ .

The conditional probabilities of the assignment variables  $c_i$  must take into account the Dirichlet prior,

$$p(c_i = k | c_{\setminus i}, \mathbf{r}, \mathbf{h}, \theta, \gamma) = \begin{cases} b \frac{n_{\setminus i, k}}{n-1+\alpha} p(r_i | h_k, \theta) & \text{if } c \text{ is populated} \\ b \frac{\alpha}{n-1+\alpha} \sum_{h'} p(r_i | h') p(h') & \text{otherwise} \end{cases}. \quad (4)$$

The factor  $b$  ensures the normalization of the probabilities. Further specializing, these equations become

$$p(c_i = k | c_{\setminus i}, \mathbf{r}, \mathbf{h}, \theta, \gamma) = \begin{cases} b \frac{n_{\setminus i, k}}{n-1+\alpha} \theta^{m_{i,k}} \left( \frac{1-\theta}{|B|-1} \right)^{m'_{i,k}} & \\ b \frac{\alpha}{n-1+\alpha} p(r_i | h_0) & \end{cases}, \quad (5)$$

where  $m_{i,k}$  and  $m'_{i,k}$  are defined similarly to (3) as

$$m_{i,k} = \sum_j \mathbb{I}(r_{i,j} = h_{k,j}) \mathbb{I}(c_i = k) \quad (6a)$$

$$m'_{i,k} = \sum_j \mathbb{I}(r_{i,j} \neq h_{k,j}) \mathbb{I}(c_i = k). \quad (6b)$$

The quantity  $p(r_i | h_0)$  is the likelihood of the read to come from any haplotype generated by the reference genome  $h_0$ , and results from our choice of a prior for the haplotype  $p(h')$ . With our assumption of all haplotypes originating from a reference genome  $h_0$ , the probability that a read originated by any possible haplotype becomes

$$\begin{aligned} \sum_{h'} p(r_i | h') p(h') &= \sum_{h'} p(r_i | h') p(h' | h_0) = p(r_i | h_0) = \\ &= \left[ \theta\gamma + (1-\gamma) \frac{1-\theta}{|B|-1} \right]^{m_{i,0}} \left[ \frac{1}{(|B|-1)^2} (\theta + \gamma + |B|(1-\gamma\theta) - 2) \right]^{m'_{i,0}}. \end{aligned} \quad (7)$$

The advantage of considering a prior different from the uniform distribution over all possible haplotypes becomes evident because under that assumption,  $\sum_{h'} p(r_i | h') p(h') = |B|^{-J}$ , which even for moderate read lengths would lead to an extremely slow mixing in the sampling.

The haplotype probability  $p(h_k | \{r_i : c_i = k\})$ , i.e. the probability of haplotype  $k$  given all the reads assigned to it, can be written base-wise, because we consider sequence positions independently:

$$p(h_k | \{r_i : c_i = k\}) = \prod_j p(h_{k,j} | \{r_{i,j} : c_i = k\}). \quad (8)$$

By means of Bayes' theorem we can rewrite this probability as

$$\begin{aligned} p(h_{k,j} | \{r_{i,j} : c_i = k\}) &= \frac{1}{Z} p(\{r_{i,j} : c_i = k\} | h_{k,j}) \\ &= \frac{1}{Z} \theta^{m_{j,k}} \left( \frac{1-\theta}{|B|-1} \right)^{m'_{j,k}}, \end{aligned} \quad (9)$$

with

$$m_{j,k} = \sum_i \mathbb{I}(r_{i,j} = h_{k,j}) \mathbb{I}(c_i = k) \quad (10a)$$

$$m'_{j,k} = \sum_i \mathbb{I}(r_{i,j} \neq h_{k,j}) \mathbb{I}(c_i = k). \quad (10b)$$

The normalization constant  $Z$  must be chosen such that the sum over all possible alleles is one. So we have

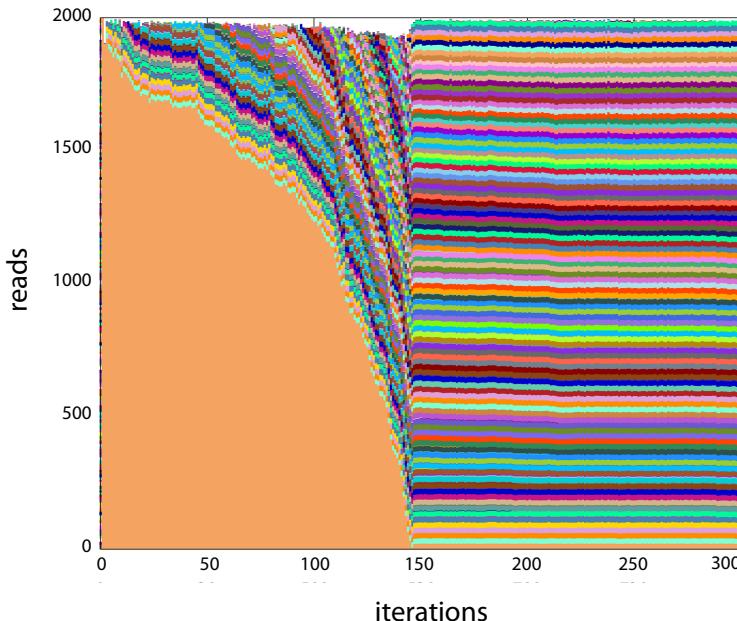
$$p(h_{k,j} | \{r_{i,j} : c_i = k\}) = \frac{\theta^{m_{j,k}} \left( \frac{1-\theta}{|B|-1} \right)^{m'_{j,k}}}{\sum_{l \in B} \theta^{m_{j,k}(l)} \left( \frac{1-\theta}{|B|-1} \right)^{m'_{j,k}(l)}}, \quad (11)$$

where  $m_{j,k}(l)$  and  $m'_{j,k}(l)$  are the same quantities defined in equation (10) with  $h_{k,j} = l$ . Finally  $\theta$  and  $\gamma$  are estimated as

$$\theta = \frac{\sum_{i,k} m_{i,k} \mathbb{I}(c_i = k)}{nJ} \quad (12a)$$

$$\gamma = \frac{\sum_k m_{k,0}}{KJ}. \quad (12b)$$

To summarize, reads are assigned to existing haplotypes or they instantiate a new class according to Equation (5). Then, haplotypes are sampled according



**Fig. 1.** Assignment of reads in the first iterations of the Gibbs sampling. Reads from 100 different haplotypes (mutual distance  $\simeq 2\%$ ) are initially assigned randomly to 200 different classes. In the first iterations almost all of them form a single giant cluster. Then they start to populate new clusters, stabilizing to 100 after about 150 iterations. All singletons are shown in white on top of the plot.

to the reads that are associated to them following Equation (9), and finally, parameters  $\theta$  and  $\gamma$  are estimated according to Equations (12). This process is iterated. The starting point for the algorithm is a random assignment of the reads to a number of haplotypes (the default number is  $n/10$ ). After a burn-in phase, we record the assignment of reads to haplotypes. If a read has been assigned for more than 80% of the iterations to a single haplotype, we consider it to originate from that haplotype. In Figure 1, the assignments of reads during the first iterations of a run of the Gibbs sampler are depicted.

In order to correct read errors, a set of successive windows along the sequence is considered. The clustering is performed in every window, considering the reads overlapping it for more than a given fraction of its length (the default value is 80%). Then the program shifts all the windows and repeats the clustering. The shifts are chosen such that every base is analyzed three times. The final error correction follows a majority rule.

### 3 Results

We test the performance of the algorithm on simulated data and on experimental data obtained from HIV samples. For the simulations, we assume different

haplotype structures and generate reads from these populations using the program ReadSim, which mimics the error process of 454/Roche sequencing machines [15][16]. Unless otherwise indicated, ReadSim has been run with the default parameters controlling the error process. The value of  $\alpha$  in the DPM has been chosen such that during the sampling the proposal of new classes happens frequently (between one and ten times per iteration).

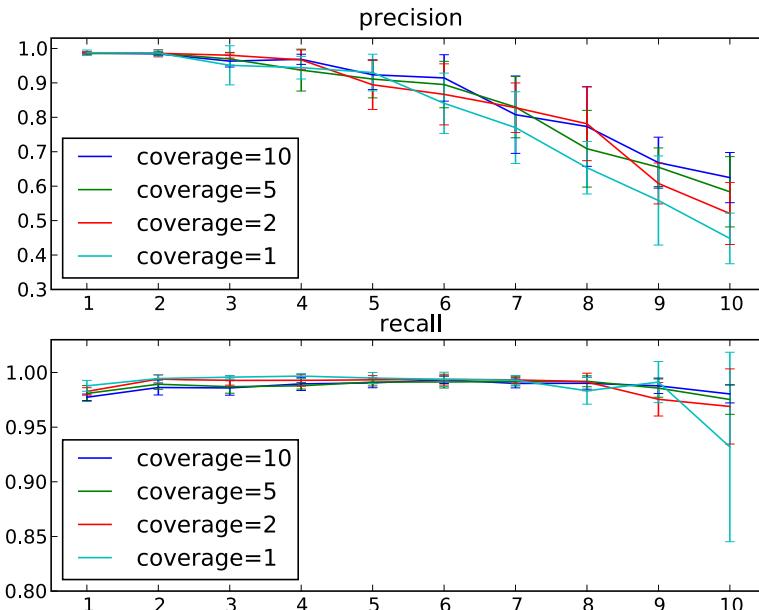
We first consider a mixture of ten haplotypes at pairwise distances between 4 and 6% with frequencies corresponding (approximately) to the first ten terms of the geometric series  $1/2 + 1/4 + 1/8 + \dots = 1$ . We report recall and precision of error correction for this mixture at different sequencing depths. Recall refers to the ability to detect errors, whereas precision measures the fraction of calls for

**Table 1.** Table summarizing the classification scheme used in the precision recall analysis.

		True classification	
		error	no error
Predicted	error, corrected	$TP_c$	$FP$
	error, mis-corrected	$TP_m$	
	no error	$FN$	$TN$

$$\text{Recall} = \frac{TP_c}{TP_c + TP_m + FN}$$

$$\text{Precision} = \frac{TP_c}{TP_c + TP_m + FP}$$



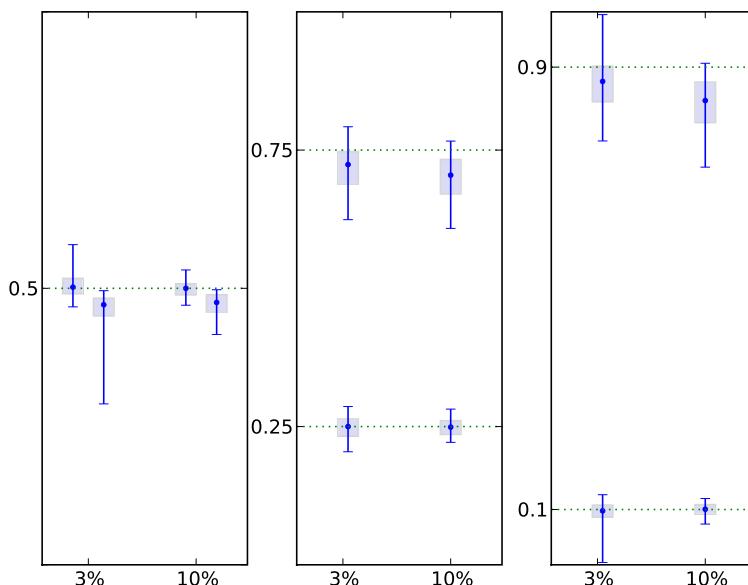
**Fig. 2.** Precision and recall for a mixture of haplotypes with exponentially decreasing proportions (frequency of  $i$ -th haplotype is  $1/2^i$ ). The lines show four different values of the expected coverage relative to the least frequent haplotype.

correction that have actually resulted in the correct base. The exact definitions of both terms are given in Table II.

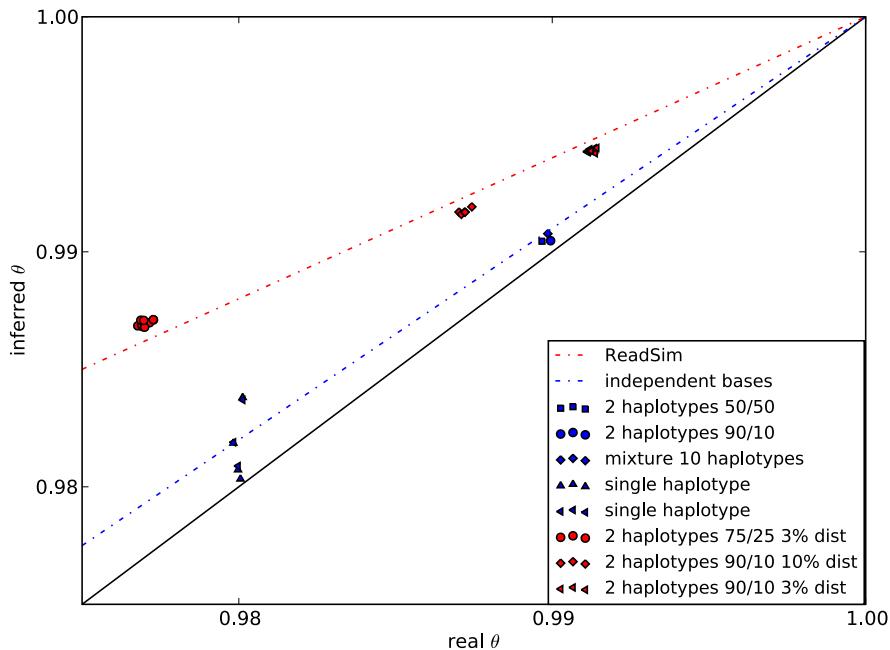
Figure 2 illustrates the high sensitivity of our method to detect errors by recall values above 0.97 for virtually all coverages and all haplotypes. By contrast, the precision of error correction depends stronger on coverage and on haplotype frequency. While precision is very high for the most frequent haplotypes, it will eventually drop below 90% for frequencies less than  $1/2^5 \approx 3.1\%$ , especially for low coverage. In this regime, reads from less frequent haplotypes tend to be assigned to haplotype clusters with higher frequency.

Another test consisted in assessing the performance in quantifying a mixture of two haplotypes, considering mixing ratios of 50:50, 75:25, and 90:10. The two haplotypes are chosen at distances of 3 and 10%, which, in the context of HIV, corresponds to strains from the same subtype and from different subtypes, respectively. A total of 4000 reads of average length 250 bp were drawn from sequences of length 1200 bp.

Figure 3 shows the fraction of reads that were assigned to the two most frequent haplotypes during sampling. For both distances, we find good agreement between the original mixing proportions and the estimated fractions, indicating



**Fig. 3.** Fraction of reads assigned to the two most frequent haplotypes in the sampling for three different mixing proportions (left, center and right) and two distances between the haplotypes, recorded for all windows of length 200. Dots, boxes, and error bars represent median, 25-75% quantile and 5-95% quantile, respectively. Distances between haplotypes are reported on the *x*-axis, mixing proportions are represented by horizontal dotted lines.

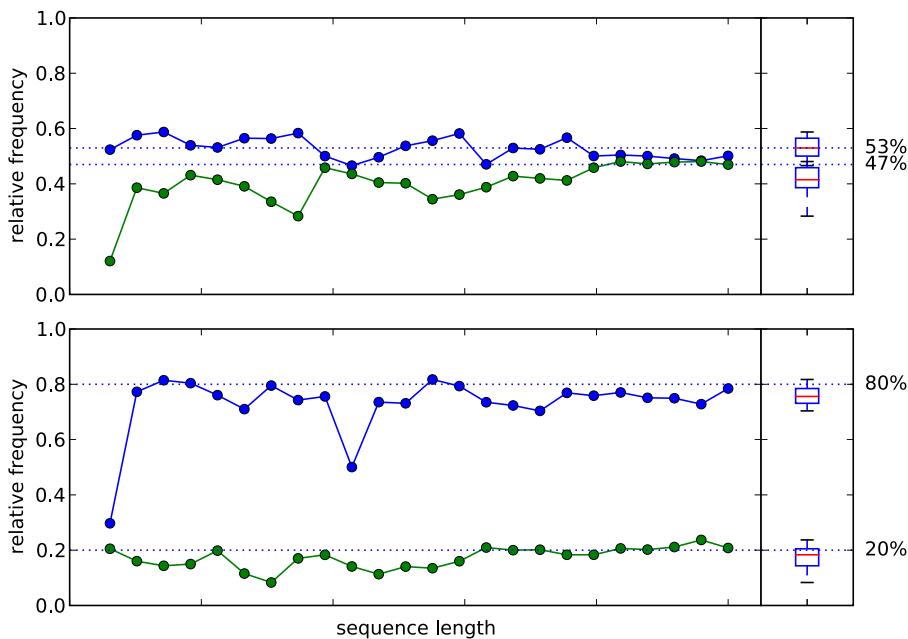


**Fig. 4.** Scatter plot for the “true” values of  $\theta$  derived from the alignment ( $x$ -axis) and inferred with the Dirichlet process ( $y$ -axis). Different symbols represent different experimental settings. The correlation between true and inferred values of  $\theta$  is preserved across various experimental settings.

that we can recover local haplotype frequencies. A smaller deviation in the estimated fraction is observed in the 10% distance case than in the 3% case, because with higher diversity, it is easier to separate the two haplotypes, and chances are lower to find a region on the sequence that shows no difference at all between the two haplotypes. In these conserved regions, the population is locally clonal and all reads are assigned to a single haplotype.

Because our goal is to separate technical noise from biological signal, it is important to assess the performance in estimating the sequencing error,  $1-\theta$ . The Gibbs sampler produces a sample of the posterior distribution of this parameter. We report in Figure 4 a scatter plot of  $\theta$  and its estimate for several experimental settings. On the  $x$ -axis the “true” value of  $\theta$  is shown as derived from aligning the simulated reads. On the  $y$ -axis, the posterior distribution is plotted.

Because  $\theta$  summarizes all features of the error process, it assigns an overall credibility to the sequencing process. To evaluate this summary, we generated reads using both ReadSim accounting for more likely indels in homopolymeric regions and our simplified error model, which assumes independence over sites. Figure 4 confirms that it is more difficult to quantify correctly the error rate when ReadSim is used. Nevertheless, for both models, the true value of  $\theta$  and the inferred distribution are highly correlated. The reason why we tend to underestimate the error rate is that a few reads are assigned to very sparsely populated clusters.



**Fig. 5.** Fraction of reads assigned to the two most frequent haplotypes in the sampling of a mixture of reads from two HIV samples. On the left, the profile of the two highest haplotype frequencies along the sequence is shown. On the right, a statistical summary of these frequencies is displayed: the horizontal solid line and the box represent the median and 25–75% quantile, respectively.

Finally, the procedure has been tested on real deep sequencing data obtained with the 454/Roche FLX pyrosequencing platform. A 1.5 kbp region of the HIV *pol* gene has been sequenced separately in four clonal samples. Two of these samples are of subtype A and two are of subtype B. After discarding reads that did not map (i.e. with more than 10% mismatches when aligned to the reference), we ran our algorithm on two sets of reads with different mixing proportions: one set of 2383 reads and a mixing ratio of 80:20, and one with 2717 reads and a ratio of 53:47.

This validation scheme captures the real distribution of reads along the haplotypes, of read errors, and of read lengths, while controlling for mixing proportions. This setup allows for a hard assessment of the performance. In Figure 5, we report the proportion of the two most frequent haplotypes reconstructed for a set of successive windows. The assignment of reads to haplotypes reflects the original proportions for the most part of the sequence.

## 4 Discussion

We have developed a specific probabilistic clustering method based on the Dirichlet process mixture for correcting technical errors in deep sequencing reads and

for highlighting the biological variation in a genetically heterogeneous sample. Unlike traditional Sanger sequencing, our analysis is not restricted to single sites, and we can estimate the co-occurrence of mutations over regions of length on the order of the read length. We have focused here on reads of average length 250 bp as generated by the 454/Roche FLX technology. With improved reagents this platform now produces reads with average length 400 bp. Therefore, our local haplotype reconstruction method can detect heterogeneity even at the level of an entire gene of interest, namely the 297 bp HIV protease gene, which is an important target of antiretroviral therapy.

Using Gibbs sampling, our approach allows for estimating the posterior distribution of the haplotype sequences and their frequencies in the population and of the technical error rate of the sequencing process. The simulations presented in this paper show that it is possible to infer the local structure of the population for different levels and patterns of diversity.

By clustering the reads we aim at eliminating technical noise while preserving the biological signal, i.e., the underlying population of haplotypes. Traditional Sanger sequencing can detect genetic variants only if their frequency exceeds a threshold of 20%. Due to the much higher coverage of deep sequencing we are able to identify variants at a much lower frequency. Obviously, a haplotype can only be identified if at least one read comes from it. This means that, in an ideal setting, it would be possible to detect variants that have a coverage of one. A deep sequencing experiment can easily produce 10,000 reads in a single run, presently of length 400 bp. For example, if we focus on a sequence of 2 kb, this results in an average coverage of 2,000 and without taking errors into account, it would be possible to detect haplotypes at 0.05% frequency. This threshold will certainly be higher in practice, because this idealized assumption is not fulfilled. Figure ② gives an estimate, albeit partial, to this threshold. Our computational experiments show that with a coverage of 10,000 we can expect to detect variants with a frequency lower than 1%, about two orders of magnitude lower than for Sanger sequencing.

The ability to resolve the population structure depends not only on coverage and on the haplotype frequencies, but also on the genetic distances between haplotypes. If we assume that the differences between two haplotypes are distributed randomly along the sequence and if we ignore sequencing errors for the moment, then the probability that the two haplotypes at distance  $d$  will be distinguished in a window of length  $L$  is  $1 - (1 - d)^L$ . For  $L = 250$  this quantity is greater than 90% already at a distance  $d = 1\%$ , and greater than 99.9% when  $d = 3\%$ . For  $L = 400$ , this quantity is greater than 90% already when  $d = 0.6\%$ , and greater than 99.9% if  $d = 2\%$ . Thus, we can, for example, identify with high confidence a coinfection of two HIV strains, not only if the two strains belong to different subtypes (about 10% expected distance), but also if they share the same subtype (about 3 to 5% expected distance).

On the other hand, other deep sequencing technologies produce much shorter reads of about 36 bp length, typically at higher coverage. With the same assumptions as above, in order to have a 90% probability of separating two haplotypes in

such a small window, their distance must be greater than 6%. This number will decrease to 4% once the announced increase in read length to 50 bp is realized. However, due to the high coverage, this technology will be able to detect even rarer variants at individual sites. A comparison of deep sequencing platforms for reconstructing mixed samples will also depend on the different error patterns of the machines and on the use of paired ends.

The generative probabilistic model for deep sequencing data from mixed samples will allow for future extensions of the methodology, including different types of sequencing errors, paired end data, partially overlapping reads, and global haplotypes. We also envision various applications, because genetic diversity is important in many biological systems, for example, in bacterial communities and in cancer cells of a tumor [17].

## Acknowledgments

We are grateful to Martin Däumer, Rolf Klein, and Bernhard Thiele (Institute of Immunology and Genetics, Kaiserslautern, Germany) for providing the HIV deep sequencing data.

## References

1. Mardis, E.R.: The impact of next-generation sequencing technology on genetics. *Trends Genet.* 24(3), 133–141 (2008)
2. Pop, M., Salzberg, S.L.: Bioinformatics challenges of new sequencing technology. *Trends Genet.* 24(3), 142–149 (2008)
3. Chi, K.R.: The year of sequencing. *Nat. Methods* 5(1), 11–14 (2008)
4. Nowak, M.A., Anderson, R.M., McLean, A.R., Wolfs, T.F., Goudsmit, J., May, R.M.: Antigenic diversity thresholds and the development of AIDS. *Science* 254(5034), 963–969 (1991)
5. Walker, B.D., Burton, D.R.: Toward an AIDS vaccine. *Science* 320(5877), 760–764 (2008)
6. Perrin, L., Telenti, A.: HIV treatment failure: testing for HIV resistance in clinical practice. *Science* 280(5371), 1871–1873 (1998)
7. Hoffmann, C., Minkah, N., Leipzig, J., Wang, G., Arens, M.Q., Tebas, P., Bushman, F.D.: DNA bar coding and pyrosequencing to identify rare HIV drug resistance mutations. *Nucleic Acids Res.* 35(13), e91 (2007)
8. Wang, C., Mitsuya, Y., Gharizadeh, B., Ronaghi, M., Shafer, R.W.: Characterization of mutation spectra with ultra-deep pyrosequencing: application to HIV-1 drug resistance. *Genome Res.* 17(8), 1195–1201 (2007)
9. Wildenberg, A., Skiena, S., Sumazin, P.: Deconvolving sequence variation in mixed DNA populations. *J. Comput. Biol.* 10(3-4), 635–652 (2003)
10. Eriksson, N., Pachter, L., Mitsuya, Y., Rhee, S.Y., Wang, C., Gharizadeh, B., Ronaghi, M., Shafer, R.W., Beerenwinkel, N.: Viral population estimation using pyrosequencing. *PLoS Computational Biology* 4(4), e1000074 (2008)
11. Westbrooks, K., Astrovskaia, I., Campo, D., Khudyakov, Y., Berman, P., Zelikovsky, A.: HCV quasispecies assembly using network flows. In: Măndoiu, I., Sunderraman, R., Zelikovsky, A. (eds.) *ISBRA 2008. LNCS (LNBI)*, vol. 4983, pp. 159–170. Springer, Heidelberg (2008)

12. Xing, E.P., Jordan, M.I., Sharan, R.: Bayesian haplotype inference via the Dirichlet process. *J. Comput. Biol.* 14(3), 267–284 (2007)
13. Saeed, F., Khokhar, A., Zagordi, O., Beerenwinkel, N.: Multiple sequence alignment system for pyrosequencing reads. In: Bioinformatics and Computational Biology (BICoB) conference 2009, LNCS (in press, 2009)
14. Neal, R.: Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics* 9(2), 249–265 (2000)
15. Schmid, R., Schuster, S., Steel, M., Huson, D.: Readsim- a simulator for sanger and 454 sequencing (unpublished) (2006)
16. Richter, D.C., Ott, F., Auch, A.F., Schmid, R., Huson, D.H., Field, D.: Metasim—a sequencing simulator for genomics and metagenomics. *PLoS ONE* 3(10), e3373 (2008)
17. Campbell, P.J., Pleasance, E.D., Stephens, P.J., Dicks, E., Rance, R., Goodhead, I., Follows, G.A., Green, A.R., Futreal, P.A., Stratton, M.R.: Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing. *Proc. Natl. Acad. Sci. USA* 105(35), 13081–13086 (2008)

# Lifting Prediction to Alignment of RNA Pseudoknots

Mathias Möhl\*, Sebastian Will\*, and Rolf Backofen\*\*

<sup>1</sup> Programming Systems Lab, Saarland University, Saarbrücken, Germany  
mmohl@ps.uni-sb.de

<sup>2</sup> Bioinformatics, Institute of Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany  
{will,backofen}@informatik.uni-freiburg.de

**Abstract.** Prediction and alignment of RNA pseudoknot structures are NP-hard. Nevertheless, several efficient prediction algorithms by dynamic programming have been proposed for restricted classes of pseudoknots. We present a general scheme that yields an efficient alignment algorithm for arbitrary such classes. Moreover, we show that such an alignment algorithm benefits from the class restriction in the same way as the corresponding structure prediction algorithm does. We look at five of these classes in greater detail. The time and space complexity of the alignment algorithm is increased by only a linear factor over the respective prediction algorithm. For four of the classes, no efficient alignment algorithms were known. For the fifth, most general class, we improve the previously best complexity of  $O(n^5m^5)$  time to  $O(nm^6)$ , where  $n$  and  $m$  denote sequence lengths. Finally, we apply our fastest algorithm with  $O(nm^4)$  time and  $O(nm^2)$  space to comparative de-novo pseudoknot prediction.

## 1 Introduction

In the last years, it has become clear that RNA molecules play very important roles in a cell, among them regulatory and catalytic ones [1], much beyond acting only as a messenger. A recent computational screen for structural RNA [2] has revealed that there are more than 30.000 putative non-coding RNAs (ncRNAs). For the functional annotation, classification and further investigation of these RNAs, it is essential to infer the secondary structure of these ncRNA, and to use this structure information for comparative analysis.

Nearly all major approaches for the computational analysis of ncRNAs have been restricted to nested secondary structure, neglecting pseudoknots. However, pseudoknots are far from being a rare event. As stated in [3], the pseudoknot motif is “among the most prevalent RNA structures”. Using exactly clustered stochastic simulations, Xayaphoummine et al. [4] have estimated that up to 30% of the base pairs in G+C-rich sequences are forming pseudoknots. For *E.coli*, it was estimated that  $15.5\% \pm 6.5\%$  of the base pairs are included in pseudoknots.

---

\* These authors contributed equally.

\*\* Corresponding author.

There are three major problems concerning the analysis of pseudoknotted RNAs, which depend on each other. First, there are only few known pseudoknots. Second, the prediction of pseudoknots is computationally very expensive. The full problem is known to be NP-hard [5], efficient algorithms exist only for restricted classes of pseudoknots. And third, the reliability of the existing prediction programs is not very good. The main reason for the low quality of the prediction programs is that there are too few known structures where the programs can be trained on, an approach that was successfully applied to nested RNA structures (see e.g. CONTRAfold [6]). Vice versa, the low prediction quality and the high computational costs hinder the research on pseudoknot structures.

The most promising way out of this dilemma is to use comparative approaches for predicting pseudoknotted secondary structures. Since the structure is more conserved than the sequence of RNA, this requires an alignment of both sequence *and* structure (called sequence-structure alignment in the following). This has been a very successful approach for nested secondary structures, where a complete variety of practical tools (e.g. LocARNA [7], MARNA [8], FOLDALIGN [9][10], Dynalign [11][12] to name some) exists. However, there are only few approaches for sequence-structure alignment for pseudoknotted approaches (lara [13]). The research on this topic is far behind the computational analysis of pseudoknot structure prediction, where several approaches in varying complexity for restricted classes have been introduced [5][14][15][16][17][18][19]. All these algorithms use the properties of the restricted class in a dynamic programming approach to efficiently solve the prediction problem.

In this paper, we consider the problem of sequence-structure alignment with known pseudoknot structures. This is the necessary basis for comparative analysis of pseudoknots. We introduce a general scheme that generates a pseudoknot alignment algorithms for a restricted class of pseudoknots, given the corresponding prediction approach. Our general scheme can be applied to pseudoknot classes for which a dynamic programming approach exists. Basically, we use the decomposition strategy of the structure prediction, and apply this strategy to solve the associated alignment problem for known structures efficiently. The additional complexity of the alignment algorithm compared to the corresponding prediction problem is only linear in the length of the sequence, both in space and time, which is surprisingly small. For comparison, in the case of the only known pseudoknot alignment methods for a restricted class of pseudoknots, namely for the class handled by Rivas&Eddy's  $O(n^6)$  algorithm [14], the corresponding alignment problem was solved by Evans [20] with a time complexity of  $O(n^{10})$  and space complexity of  $O(n^8)$ . Compared to that, our approach requires only  $O(n^7)$  time and  $O(n^5)$  space on this class, although it supports a more general scoring scheme. Furthermore, the run time of the algorithms generated by our general scheme scales with the complexity of the aligned structures. In particular, the worst case complexity applies only to the actual pseudoknots, whereas simpler parts of the structures are aligned much faster. A central technical insight of the work is how pseudoknot alignment can benefit from a variety of structural restrictions in the same way as structure prediction does. We discuss

five classes of pseudoknot structures in detail, namely [5]–[14]–[15]–[16]–[17]–[18]–[19]. For four of these classes, no exact alignment algorithms existed up to now.

Then, we used the fastest of our introduced alignment approaches that handles the structure class described by Reeder&Giegerich [19]. The resulting algorithm has a time complexity of  $O(n^5)$ , compared to  $O(n^4)$  for the prediction. We tested this approach on known pseudoknot classes from the Rfam-databases [21] and compared it with an implementation of the nested-nested sequence-structure alignment method of [22] in the tool MARNA [8]. We found that using the known pseudoknot structures improves the quality of alignments, especially for low pairwise sequence identity. Furthermore, we set up a first prototype pipeline for combining alignment and prediction of pseudoknot structures. Such a pipeline is the basis for automatic pseudoknot annotation of large amounts of candidate ncRNA as predicted by ncRNA screens (e.g. provided by the RNAz-tool [23]), where the secondary structure is unknown. We applied our prototype pipeline to a data set of 50 putative ncRNAs from a current *Ciona intestinalis* screen [24]. This revealed that using a comparative approach increases the reliability of pseudoknot structure annotation. Whereas the Reeder&Giegerich prediction method pknotsRG would find pseudoknots in all of the 50 examples, only 14 of them show sufficiently many compensatory base pair mutations after using alignment in addition to the pseudoknot structure prediction. Such compensatory base pair mutations enlarges the reliability of the prediction since they give a strong evidence for an evolutionary conservation of the structure. Compensatory base pair mutations are commonly used as a verification for real biologically relevant structures. This prototypical pipeline is only an initial step and some work remains to be done to setup an integrated pipeline. However, the prototype application shows that such an integrated pipeline is feasible. With the work presented here, we have done the first but essential step to set up such an approach.

*Related work.* Evans proposed a fixed parameter tractable algorithm that computes the longest arc preserving common subsequence of pseudoknots [25] and an algorithm to compute the maximum common ordered substructure of two RNA structures [20]. As mentioned above, the latter guarantees polynomial run time, but the polynomial is significantly larger than in our approach. In our own previous work, we developed a fixed parameter tractable algorithm to align arbitrary pseudoknots [26]. Despite addressing the same task, this algorithm differs much from our current approach and the run time depends on other properties of the input structures. Finally, there exists a heuristic approach based on integer linear programming that is usually fast in practice, but does not guarantee optimal solutions [13].

*Roadmap.* After giving basic concepts (Sec. 2), we present a general framework for decomposing RNA structures that can be instantiated with the methods used by various pseudoknot prediction algorithms (Sec. 3). Then we describe a general method to construct a corresponding sequence structure alignment algorithm for each of these instances (Sec. 4) and have a detailed look at the different instances (Sec. 5). Finally, we present experimental results obtained with our implementation (Sec. 6) and give some concluding remarks (Sec. 7).

## 2 Preliminaries

*Sequences and fragments.* An *arc-annotated sequence* is a pair  $(S, P)$ , where  $S$  is a string over the set of bases  $\{A, U, C, G\}$  and  $P$  is a set of arcs  $(l, r)$  with  $1 \leq l < r \leq |S|$  representing bonds between bases, such that each base is adjacent to at most one arc. We denote the  $i$ -th symbol of  $S$  by  $S[i]$ . For an arc  $p = (l, r)$ , we denote its left end  $l$  and right end  $r$  by  $p^L$  and  $p^R$ , respectively. An arc  $p \in P$  is *crossing* if there is an arc  $p' \in P$  such that  $p^L < p'^L < p^R < p'^R$  or  $p'^L < p^L < p'^R < p^R$ ; then  $p$  and  $p'$  form a *pseudoknot*. An arc-annotated sequence is *crossing* if it contains crossing arcs, otherwise it is *nested*.

An *alignment*  $A$  of two arc-annotated sequences  $(S_a, P_a)$  and  $(S_b, P_b)$  is a set  $A_1 \cup A_2$ , where  $A_1 \subseteq [1..|S_a|] \times [1..|S_b|]$  is a set of *match edges* such that for all  $(i, j), (i', j') \in A_1$  holds 1.)  $i > i'$  implies  $j > j'$  and 2.)  $i = i'$  if and only if  $j = j'$  and  $A_2$  is the set of *gap edges*  $\{(x, -) \mid x \in [1..|S_a|] \wedge \nexists y. (x, y) \in A_1\} \cup \{(-, y) \mid y \in [1..|S_b|] \wedge \nexists x. (x, y) \in A_1\}$ . A base that is adjacent to a gap edge is called *aligned to a gap*. Two bases  $S_a[i], S_b[j]$  are matched by  $A$  if  $(i, j) \in A$  and two arcs  $p_a \in P_a, p_b \in P_b$  are matched if  $(p_a^L, p_b^L) \in A$  and  $(p_a^R, p_b^R) \in A$ . Each alignment has an associated cost based on an edit distance with two classes of operations. The operations were first introduced by Jiang et al. [22] and are illustrated in Fig. 1. Base operations (mismatch and insertion/deletion) work solely on positions that are not incident to an arc. *Base mismatch* replaces a base with another base and has associated cost  $w_m$ . A *base insertion/deletion* removes or adds one base and costs  $w_d$ . The second class consists of operations that involve at least one position that is incident to an arc. An *arc mismatch* replaces one or both of the bases incident to an arc. It costs  $\frac{w_{am}}{2}$  if one base is replaced or  $w_{am}$  if both are replaced. An *arc breaking* removes one arc and leaves the incident bases unchanged. The associated cost is  $w_b$ . *Arc removing* deletes one arc and both incident bases and costs  $w_r$ . Finally, *arc altering* removes one of the two bases that are incident to an arc and costs  $w_a = \frac{w_b}{2} + \frac{w_r}{2}$  (cf. [22]).

Define for two arc annotated sequences  $(S_a, P_a)$  and  $(S_b, P_b)$  and  $k \in \{a, b\}$

$$\begin{aligned}\chi(i, j) &:= \text{if } S_a[i] \neq S_b[j] \text{ then 1 else 0} \\ \psi_k(i) &:= \text{if } \exists p \in P_k. p^L = i \text{ or } p^R = i \text{ then 1 else 0} \\ \text{gap}_k(i) &:= w_d + \psi_k(i) \left( \frac{w_r}{2} - w_d \right) \\ \text{basematch}(i, j) &:= \chi(i, j) w_m + (\psi_1(i) + \psi_2(j)) \frac{w_b}{2}.\end{aligned}$$

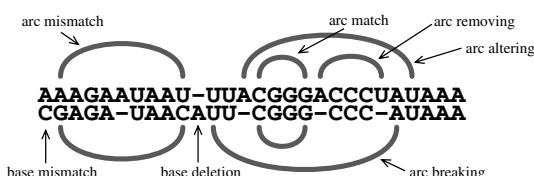


Fig. 1. Edit operations (cf. [22])

$\text{gap}_k(i)$  denotes the cost to align base  $S_k[i]$  to a gap,  $\text{basematch}(i, j)$  the cost to align  $S_a[i]$  to  $S_b[j]$  under the assumption that their possibly adjacent arcs are not matched.

### 3 Decomposing Sequences

In general, dynamic programming (DP) algorithms for RNA alignment, structure prediction or similar tasks, rely on a recursive decomposition of the RNA sequence into subsequences or combinations of subsequences (which we will call fragments in the following). For example, the standard nested secondary structure prediction [27,28] decomposes a subsequence into two consecutive subsequences, whereas the Rivas and Eddy algorithm [14] for predicting a restricted class of pseudoknots uses fragments that consist of two unconnected subsequences. Thus, the type of decompositions considered by each algorithm has major impact on both complexity and the class of structures handled by the algorithm.

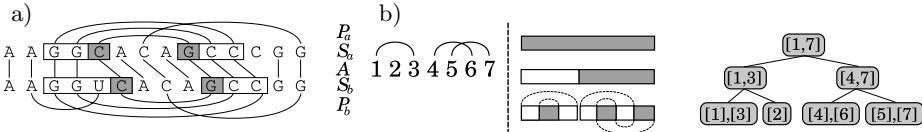
We will develop a general view on decomposition strategies for RNA structures with pseudoknots. As a first insight, we point out that the central difference between the various DP based structure prediction algorithms is their choice of one such strategy. This choice determines the characteristic trade-off between the class of handled pseudoknots and the resulting complexity of the algorithm. As our main contribution, we will introduce a general framework for sequence-structure alignment based on an arbitrary decomposition strategy. Compared to the structure prediction algorithm for this decomposition strategy, the alignment algorithm will have only linear time and space overhead. Thus the scheme provides the first sequence-structure alignment methods for many pseudoknot classes and covers the classes of all known DP based pseudoknot prediction algorithms.

A fragment  $F$  of an arc annotated sequence  $(S, P)$  is a  $k$ -tuple of intervals  $([l_1, r_1], \dots, [l_k, r_k])$  with  $1 \leq l_1 \leq r_1 + 1 \leq \dots \leq l_k \leq r_k + 1 \leq |S|$ . Note that this definition allows *empty intervals*  $[i+1, i]$ . The ranges between the intervals, i.e.  $[r_1 + 1, l_2 - 1], \dots, [r_{k-1} + 1, l_k - 1]$ , are called *gaps of  $F$* . We call  $k$  the *degree* of  $F$  and  $l_1, r_1, \dots, l_k, r_k$  its *boundaries*. The *set of positions covered by  $F$*  (denoted with  $\hat{F}$ ), is defined as the union of the intervals contained in  $F$ . The  $i$ -th interval  $[l_i, r_i]$  of  $F$  is denoted with  $F[i]$  and with  $F[i]^L$  and  $F[i]^R$  we denote its *left and right boundary  $l_i$  and  $r_i$* , respectively. For better readability, we abbreviate intervals of the form  $[i, i]$  as  $\langle i \rangle$ .

$F$  is called *arc-complete*, iff  $l \in \hat{F} \Leftrightarrow r \in \hat{F}$  for each  $(l, r) \in P$ .  $F$  is called *atomic* if  $F$  covers either exactly the two ends of an arc of  $P$  or a single position not adjacent to an arc. As an example, in Fig 2a, the boxed fragments have all a degree of 2,  $F_a^2$  and  $F_b^2$  are atomic and  $F_b$  as well as  $F_b^1$  are not arc-complete.

Let  $F$ ,  $F^1$  and  $F^2$  be fragments of the same sequence. The pair  $(F^1, F^2)$  is a *split* of  $F$  iff  $\hat{F} = \hat{F}^1 \uplus \hat{F}^2$ <sup>1</sup>. We call  $F^1$  and  $F^2$  the *children* and  $F$  the *parent of the split*. The split is called *arc-preserving*, if  $F$ ,  $F^1$  and  $F^2$  are arc complete.

<sup>1</sup> For simplicity, we introduce only binary splits. However, the introduced concepts are raised to n-ary splits straightforwardly.



**Fig. 2.** a) Alignment with some boxed fragments  $F_a, F_b$  that are split into their white and gray parts  $F_a^1, F_a^2$  (white boxes) and  $F_b^1, F_b^2$  (gray boxes), respectively. Gap edges of  $A$  are not shown. b) A structure and two ways to visualize a parse tree thereof. Note that in the parse tree, each leaf is atomic.

In Fig. 2a, the split  $(F_a^1, F_a^2)$  of  $F_a$  is arc-preserving – the split  $(F_b^1, F_b^2)$  of  $F_b$  is not arc-preserving due to the leftmost arc of  $P_b$ .

A *parse tree of a sequence*  $(S, P)$  is a binary tree where each node is an arc-complete fragment of  $(S, P)$  such that (a) the root is  $([1, |S|])$ , (b) each inner node is a fragment  $F$  and has two children  $F_1$  and  $F_2$ , such that  $(F_1, F_2)$  is an arc-preserving split of  $F$ , (c) each leaf is an atomic fragment. Fig. 2b shows two ways to visualize a parse tree.

A parse tree represents one fixed recursive decomposition of a sequence. The basic idea of our alignment algorithm scheme is to handle the sequences asymmetrically. The algorithm recursion follows a single fixed parse tree for the first sequence. At each split of the parse tree, it considers *all* compatible splits of the second sequence. In contrast to the splits of the parse tree, these compatible splits don't have to be arc-preserving. Formally, two splits are compatible, if they have the same split type which is defined as follows.

The *basic type of a split*  $(F^1, F^2)$  of a fragment  $F$  is defined by the following construction. The interval  $[\min(\hat{F}), \max(\hat{F})]$  decomposes into the intervals of  $F^1$ , the intervals of  $F^2$  and gaps of  $F$ . If we order these from left to right and replace the intervals of  $F^1$  by 1, the ones of  $F^2$  by 2 and the remaining ones by  $G$  (for gap), we obtain a string  $T$  over  $\{1, 2, G\}$  that we call the *basic type* of the split. Every split has exactly one basic type.

The type can be further refined by annotating constraints. In particular, we introduce the *size constraint* that restricts an interval to have at most size one in each instance of the type. It is indicated by marking the respective symbols 1 or 2 in the type with '. Size constraints will be used to describe the common case of splits that split off an atomic fragment. A type containing constraints is called a *constrained type*. Note that each size constraint reduces the number of splits of this type by one order of magnitude, because it reduces the degrees of freedom by one.

If  $(F^1, F^2)$  is of type  $T$ , we call it a *T-split*. As an example, in Fig. 2a the splits of  $F_a$  and  $F_b$  are of basic type 12G21 and of constrained type 12'G2'1.

The complexity of the alignment algorithm depends on the number of children and parent instances of the considered split types. On the number of parents, because for a given fragment in one sequence, we consider all fragments in the other sequence having the same type. On the number of children, because it determines the number of ways an aligned fragment can be split in sub-alignments. These numbers depend on the length  $m$  of the second sequence and are defined

as the *number of children* and the *number of parents*, respectively:

$$\#_C^m(T) = |\{(F^1, F^2) \mid (F^1, F^2) \text{ is a T-split of some } F \text{ and } \hat{F} \subseteq [1, m]\}|$$

$$\#_P^m(T) = |\{F \mid \exists(F^1, F^2) \text{ that is a T-split of } F \text{ and } \hat{F} \subseteq [1, m]\}|$$

**Lemma 1.** *For some sequence with length  $m$  and a split type  $T$ , let the degree of the parent and the two children be  $k_p, k_1$  and  $k_2$ , respectively. Furthermore, let  $c$  denote the degrees of freedom that are reduced by the constraints of  $T$  and  $c' \leq c$  denote the corresponding reduction for the parent instances. Then  $\#_C^m(T) \in O(m^{k_p+k_1+k_2-c})$  and  $\#_P^m(T) \in O(m^{2k_p-c'})$ .*

Due to space limitations, the proofs of all lemmata are available only online at <http://www.bioinf.uni-freiburg.de/Supplements/pkalign-recomb09>.

## 4 The Alignment Algorithm Scheme

### 4.1 The Variant for Basic Types

The algorithm takes two arc-annotated sequences  $(S_a, P_a)$ ,  $(S_b, P_b)$  and a parse tree for  $(S_a, P_a)$  as input<sup>2</sup>. For each fragment in the parse tree, the algorithm recursively computes alignments to all fragments of  $(S_b, P_b)$  that have the same basic type. In order to present the precise recursions, we need the following formal notion for alignments of fragments.

The restriction of an alignment  $A$  to fragments  $F_a, F_b$  is defined as  $A|_{F_a \times F_b} := \{(i, j) \in A \mid i \in \hat{F}_a \cup \{-\}, j \in \hat{F}_b \cup \{-\}\}$ .  $A$  aligns two fragments  $F_a$  and  $F_b$  of the same degree  $k$ , short align <sub>$A$</sub> ( $F_a, F_b$ ), if and only if for all  $(a_1, a_2) \in A$  and for all  $i \in 1 \dots k$  it holds that  $a_1 = -$  or  $a_2 = -$  or  $a_1 \in F_a[i] \Leftrightarrow a_2 \in F_b[i]$ . Note that for a given alignment  $A$ , a fragment of one sequence can be aligned to several fragments of the other; consider e.g. Fig. 2a, where  $A$  aligns  $F_a^1$  to  $F_b^1 = ([3, 5], [11, 12])$  and also to  $([3, 4][11, 12])$ .

The cost of  $A$  can be computed recursively as cost( $A$ ) with

$$\begin{aligned} \text{cost}(\{(i, -)\} \sqcup A') &= \text{gap}_a(i) + \text{cost}(A') \\ \text{cost}(\{(-, j)\} \sqcup A') &= \text{gap}_b(j) + \text{cost}(A') \\ \text{cost}(\{(l_1, l_2), (r_1, r_2)\} \sqcup A') &= (\chi(l_1, l_2) + \chi(r_1, r_2)) \frac{w_{am}}{2} + \text{cost}(A') \\ &\quad \text{if } (l_1, r_1) \in P_a, (l_2, r_2) \in P_b \\ \text{cost}(\{(i, j)\} \sqcup A') &= \text{basematch}(i, j) + \text{cost}(A') \\ &\quad \text{if third case is not applicable.} \end{aligned}$$

This computation relies only on the property of the scoring scheme that all costs except for matching an arc are local to a single base. If  $A$  aligns two fragments

<sup>2</sup> Such a parse tree can be constructed using standard parsing techniques. Furthermore, the structure prediction algorithms discussed later implicitly construct parse trees that can also be reused for this purpose.

$F_a$  and  $F_b$ , the cost is computed analogously as  $C_A(F_a, F_b) := \text{cost}(A|_{F_a \times F_b})$ . The optimal cost to align two fragments is defined as

$$C(F_a, F_b) := \min_{A \text{ with } \text{align}_A(F_a, F_b)} \{C_A(F_a, F_b)\}.$$

The optimal cost to align the entire sequences is  $C(F_a, F_b)$  for  $F_a = ([1, |S_a|])$  and  $F_b = ([1, |S_b|])$ . It can be computed by recursively applying the next lemma, where  $F_a^1$  and  $F_a^2$  are chosen according to the parse tree.

**Lemma 2 (Split lemma).** *Let  $F_a$  and  $F_b$  be fragments of  $(S_a, P_a)$  and  $(S_b, P_b)$ , respectively. Let  $(F_a^1, F_a^2)$  be an arc-preserving split of  $F_a$  of basic type  $T$ . Then*

$$C(F_a, F_b) = \min_{T\text{-split } (F_a^1, F_a^2) \text{ of } F_b} \{C(F_a^1, F_b^1) + C(F_a^2, F_b^2)\} \quad (1)$$

Note that if the split of  $F_b$  is not arc-preserving, the respective arcs are broken or removed, since there is no arc of  $F_a$  that they can be matched to. The cost for breaking or removing the two ends of the arcs is contained in  $C(F_a^1, F_b^1)$  and  $C(F_a^2, F_b^2)$ , respectively. The evaluation of the recursion is done efficiently by dynamic programming, i.e. all intermediate values  $C(F_a, F_b)$  are tabulated, such that each instance is computed only once. The recursive case, shown in Fig. 3a, is directly given by Eq. (1). At the leafs of the parse tree, the base cases, shown in Fig. 3b, are applied. The actual alignment can be constructed using the usual back-trace techniques.

a) Recursive case:

$$C(F_a, F_b) = \min_{T\text{-split } (F_a^1, F_a^2) \text{ of } F_b} \{C(F_a^1, F_b^1) + C(F_a^2, F_b^2)\},$$

where the parse tree splits  $F_a$  into  $(F_a^1, F_a^2)$  by a split of basic type  $T$

b) Base cases:

$$C(\langle i \rangle, [l, r]) = \min \begin{cases} C(\langle i \rangle, [l+1, r]) + \text{gap}_2(l) & \text{if } l \leq r \\ C(\langle i \rangle, [l, r-1]) + \text{gap}_2(r) & \text{if } l \leq r \\ \text{basematch}(i, l) & \text{if } l = r \\ \text{gap}_1(i) & \text{if } l > r \end{cases}$$

$$C(F_a = (\langle p^L \rangle, \langle p^R \rangle), ([l_1, r_1], [l_2, r_2])) =$$

$$\min \begin{cases} C(\langle p^L \rangle, [l_1, r_1]) + C(\langle p^R \rangle, [l_2, r_2]) \\ C(F_a, ([l_1+1, r_1], [l_2, r_2])) + \text{gap}_2(l_1) & \text{if } l_1 \leq r_1 \\ C(F_a, ([l_1, r_1-1], [l_2, r_2])) + \text{gap}_2(r_1) & \text{if } l_1 \leq r_1 \\ C(F_a, ([l_1, r_1], [l_2+1, r_2])) + \text{gap}_2(l_2) & \text{if } l_2 \leq r_2 \\ C(F_a, ([l_1, r_1], [l_2, r_2-1])) + \text{gap}_2(r_2) & \text{if } l_2 \leq r_2 \\ (\chi(p^L, l_1) + \chi(p^R, l_2)) \frac{w_{am}}{2} & \text{if } (l_1, l_2) = (r_1, r_2) \in P_b \end{cases}$$

**Fig. 3.** a) Recursive case for basic split type and b) base cases of the algorithm

*Complexity.* Let  $n$  and  $m$  be the length of the two sequences, respectively. First note that the parse tree has only  $O(n)$  nodes, since each split introduces at least one new boundary, of which there exist only  $O(n)$  many. Let  $T_p$  and  $T_c$  be types of splits in the parse tree, where  $\#_P^m(T_p)$  and  $\#_C^m(T_c)$  are maximal among the occurring split types, respectively. For a node  $F_a$  with split  $T_p$  the algorithm materializes the costs  $C(F_a, F_b)$  for  $\#_P^m(T_p)$  fragments  $F_b$ . Assuming the worst case for each node, this results in a space complexity of  $O(n \cdot \#_P^m(T_p))$ . The time complexity is dominated by the computation at  $T_c$ -splits. There, according to Lem. 2, the algorithm minimizes over  $\#_C^m(T_c)$  terms; each is computed in  $O(1)$ . This results in a worst case time complexity of  $O(n \cdot \#_C^m(T_c))$ .  $\#_P^m(T_p)$  and  $\#_C^m(T_c)$  are asymptotically bounded due to Lemma 1. For the case of non-constrained, basic types we instantiate to  $O(nm^{2k})$  space and  $O(nm^{3k})$  time complexity, where  $k$  is the maximal degree among the splits in the parse tree.

## 4.2 An Optimized Variant for Constrained Types

By the preceding complexity analysis, the time and space complexity directly depend on  $\#_P^m(T)$  and  $\#_C^m(T)$  for the basic types. Lemma 1 shows how constraints in types reduce these numbers, and thus bear the potential to reduce the complexity. However, we cannot simply use constraint types instead of basic types in the recursion of Fig. 3a. Let's assume that  $F_a^1$  is atomic and  $T$  is constrained correspondingly; furthermore,  $T_u$  denotes the unconstrained, basic split type corresponding to  $T$ . In the optimal alignment,  $F_a^1$  is not necessarily aligned to an atomic  $F_b^1$ . However, we know (by Lem. 2) that for any  $T_u$ -split  $(F_b^1, F_b^2)$  of  $F_b$ , at most one of the bases of  $F_b^1$  per interval of  $F_b^1$  is matched to  $F_a^1$  and the others are aligned to gaps. Using this observation, we can still split off a fragment  $F_b^1$  of  $F_b$  satisfying the constraint type  $T$  after ‘eating away’ the gaped bases, which we do by introducing ‘shrink’-cases.

The following lemma directly leads to the optimized recursion equation as given in Fig. 4.

**Lemma 3 (Split lemma for constrained types).** *Let  $F_a$  and  $F_b$  be fragments of  $(S_a, P_a)$  and  $(S_b, P_b)$ , respectively. Let  $(F_a^1, F_a^2)$  be an arc-preserving  $T$ -split of  $F_a$ , where  $T$  contains size constraints for at most one of the fragments and at least one boundary of each interval of the constrained fragment coincides with a boundary of  $F_a$ .<sup>3</sup> Let  $A$  be an optimal alignment of  $F_a$  and  $F_b$ . Then there is a  $T$ -split  $(F_b^1, F_b^2)$  of  $F_b$  such that either the constrained fragment of the split is matched to one or two gaps by  $A$  and the remaining fragment is aligned to  $F_a$  or there is a  $T$ -split  $(F_b^1, F_b^2)$  such that  $A$  aligns  $F_a^1$  to  $F_b^1$  and  $F_a^2$  to  $F_b^2$ .*

In extension of Lem. 2, Lem. 3 allows size constraints in the split type  $T$ . According to the lemma, the recursion of the optimized algorithm shown in Fig. 4 introduces additional shrink cases. These cover the minimization cases where one cannot split as in Lem. 2 by the (now possibly constrained) split type  $T$ .

<sup>3</sup> This condition can be generalized to constraints on more than one fragment as long as no two adjacent symbols are constrained. This is mostly relevant for  $n$ -ary splits.

$$C(F_a, F_b) = \min_{T\text{-split } (F_b^1, F_b^2) \text{ of } F_b} \min \begin{cases} C(F_a^1, F_b^1) + C(F_a^2, F_b^2) \\ C(F_a, F_b^2) + C(-, F_b^1) & \text{if } T \text{ contains some 1'} \\ C(F_a, F_b^1) + C(-, F_b^2) & \text{if } T \text{ contains some 2'} \end{cases}$$

**Fig. 4.** Optimized recursive case. This applies to the general case of a constrained type  $T$  satisfying the conditions in Lem. 3.  $C(-, F_b^i)$  denotes the cost of deleting  $F_b^i$ .

## 5 Instances of the Algorithm Scheme

In this section, we focus on the behaviour of our general algorithm scheme for different restricted classes of pseudoknots. We analyse the classes of pseudoknots produced by different structure prediction algorithms [5][14][15][16][17][18][19] and show that the alignment can benefit of the structural restrictions in exactly the same way as the prediction. In particular we show for each of the prediction algorithms how to construct a corresponding alignment algorithm with only a linear increase in complexity (see Table 1). Following Condon *et al.* [29], we name the classes of structures according to the authors of the respective prediction algorithms: R&E (Rivas and Eddy [14]), A&U (Akutsu [16] and Uemura [15]), L&P (Lyngsø and Pedersen [5]), D&P (Dirks and Pierce [18] and R&G (Reeder and Giegerich [19]). Also note that on nested structures the algorithm behaves like an algorithm by Jiang *et al.* [22].

*R&E structures.* The prediction algorithm by Rivas and Eddy [14] requires  $O(n^6)$  time and  $O(n^4)$  space. It is restricted to structures for which parse trees exist where each fragment has a degree of at most 2. Our algorithm aligns structures from this class in  $O(nm^6)$  time and  $O(nm^4)$  space. Compared to that, the best alignment algorithm for this class known so far (by Evans [20]) requires  $O(n^5m^5)$  time and  $O(n^4m^4)$  space.<sup>4</sup>

*A&U structures.* The algorithms of Akutsu [16], Uemura [15] and Deogun et al. [17] predict structures with simple pseudoknots in  $O(n^4)$  time and  $O(n^3)$  space. For the structures of this class, there always exist parse trees, where the splits are limited to the constrained types

$$12 \ 121 \ 12'G2'1 \ 1G2'12' \ 12'G1 \ 1G2'1 \ 1G12' \ 12'G2'.$$

By Lem. 1, there exist only  $O(m^4)$  splits  $(F_b^1, F_b^2)$  of these types (i.e.  $\#_C^m(T) \in O(m^4)$  for all but the first and last of the above types). Hence our algorithm runs in  $O(nm^4)$  time on these structures. Due to Lem. 1, the space complexity is at most  $O(nm^4)$  too; this can be improved to  $O(nm^3)$ , because for the allowed splits all unconstrained children fragments of degree 2, which dominate the space complexity, have the same leftmost boundary as their parent. In the computation of the costs  $C(F_a, F_b)$ , we can thus group all such fragments  $F_b$  with the same leftmost boundary. For each of these groups only  $O(m^3)$  fragments  $F_b$  exist and the space is reused for each group. A rigorous argument for the improvement is

<sup>4</sup> Evan's algorithm computes the longest arc-preserving common subsequence, which can be considered as a special case of our edit distance measure.

**Table 1.** Pseudoknot classes and complexity of their prediction and alignment.

class		R&E	A&U	L&P	D&P	R&G
prediction	time	$O(m^6)$	$O(m^4)$	$O(m^5)$	$O(m^5)$	$O(m^4)$
	space	$O(m^4)$	$O(m^3)$	$O(m^3)$	$O(m^4)$	$O(m^2)$
(literature)	time	$O(n^5 m^5)$	-	-	-	-
	space	$O(n^4 m^4)$	-	-	-	-
(new scheme)	time	$O(nm^6)$	$O(nm^4)$	$O(nm^5)$	$O(nm^5)$	$O(nm^4)$
	space	$O(nm^4)$	$O(nm^3)$	$O(nm^3)$	$O(nm^4)$	$O(nm^2)$

given by the next lemma, which is proved by simultaneous induction over the parse tree.

**Lemma 4 (Improved space complexity for A&U).** *Let  $F_a$  be a node of a parse tree for an A&U structure, where  $F_a$  has degree  $k$  and  $n'$  descendants. Then,  $O(n' m^3)$  space suffices to compute 1.) for  $k = 1$  or atomic  $F_a$  of degree 2, all  $O(m^2)$  costs  $C(F_a, F_b)$  and 2.) for  $k = 2$ , all  $O(m^3)$  costs  $C(F_a, F_b)$ , where all  $F_b$  share a fix leftmost boundary.*

This space improvement follows a general principle applicable in dynamic programming that makes use of invariants on the most complex items and groups their computation. This invariant is conveniently reflected in our representation by split types.

*L&P structures.* Lyngsø and Pedersen [5] predict certain pseudoknots in  $O(n^5)$  time and  $O(n^3)$  space. Their class of pseudoknots is restricted such that there must exist a parse, where the split of the root has basic type 12121 and such that the two fragments of this split are both nested. This implies that they can be further decomposed with splits of constrained types

$$\begin{array}{lllll} 12'G2'1G1 & 1G21G1 & 12G1G1 & 2G1G12 & \text{(for fragments with degree 3)} \\ 2'1G12' & 21G1 & 1G12 & 1G2 & \text{(for fragments with degree 2)} \\ 2'12' & 12 & & & \text{(for fragments with degree 1)} \end{array}$$

The degree 3 splits decompose only fragments that start with the first sequence position and end with the last sequence position; it suffices to consider only splits of  $F_b$  that share this property. This means, we see here a further example of a type constraint. This *maximality constraint* reduces the degrees of freedom for this type by 2. Indicating the constraints, we refine the first four types to  $\downarrow 12'G2'1G1 \downarrow \downarrow 1G21G1 \downarrow \downarrow 12G1G1 \downarrow \downarrow 2G1G12 \downarrow$ .

To see that no other splits for fragments of degree 3 are required, note that with the first three split types, the fragment can be decomposed until its first two intervals are no more connected by arcs and once this is the case, a split of the fourth type can be applied.  $\square$

<sup>5</sup> Lyngsø and Pedersen give another intuition based on the idea of considering cyclic sequences. This is reflected by the split types in the way that each split for fragments of degree 3 is analogous to the split for fragments of degree 2 below it, if the part before the first gap is cyclically moved to the end of the type.

The complexity analysis with Lem. II leads for each of these split types to at most  $O(m^5)$  instances, which implies  $O(nm^5)$  time complexity. For example, for split type  $T = \downarrow 12'G2'1G1\downarrow$ , we have  $k_p = k_1 = 3$ ,  $k_2 = 2$ ,  $c = 4$  and hence  $\#_C^m(T) \in O(m^{3+3+2-4}) = O(m^4)$  and for  $T' = \downarrow 2G1G12\downarrow$  we have  $k_p = 3$ ,  $k_1 = k_2 = 2$ ,  $c = 2$  and hence  $\#_C^m(T') \in O(m^{3+2+2-2}) = O(m^5)$ .

The space complexity can be reduced from  $O(nm^4)$  to  $O(nm^3)$  by the same general principle that we observed for the A&U structures. That is, first we can distinguish *simple fragments* that have degree 1 or are constrained and *complex fragments* that are unconstrained and have degree 2. Then, we identify an invariant for the occurring complex fragments and use it for space reduction by grouping. In the L&P split types, the second fragment is always simple. Hence, for each split there are only  $O(m^2)$  values to store for the second fragments.

The first fragment of each split type has either degree less than 2 or it contains the pattern  $1G1$ , where  $G$  is the last gap in the split type. This implies that the computation of each fragment of degree at least 2, recursively relies only on fragments that have the size of the last gap in common. The costs for the simple fragments can thus always be computed by grouping the complex fragments by this gap size and reusing the memory.

*D&P structures.* Dirks and Pierce [18] developed an algorithm to compute the partition function for RNA pseudoknots, which can also be modified to predict the MFE structure. The algorithm takes  $O(n^5)$  time and  $O(n^4)$  space. The corresponding parse trees of the predicted structures are limited to fragments of degree at most two and the splits are of types

$$12 \ 1212 \ 21G1 \ 12G1 \ 1G21 \ 1G12 \ 1'2G21'.$$

Again, according to Lem. II there exist at most  $O(m^5)$  splits  $(F_b^1, F_b^2)$  for each of these types and hence our alignment algorithm requires  $O(nm^5)$  time and  $O(nm^4)$  space on these structures.

*R&G structures.* The efficiency of the Reeder and Giegerich [19] structure prediction algorithm ( $O(n^4)$  time,  $O(n^2)$  space) is due to the restriction to canonical pseudoknots. A stem of base pairs is called *canonical* if it cannot be extended by another valid base pair. The canonical stem containing a given base pair is thus uniquely determined. In R&G structures, pseudoknots are formed only by two crossing canonical stems. Reeder and Giegerich structures can be decomposed by split types

$$2'1 \ 12' \ 12 \ 1'21' \ E_1 = 1^c 23^c 41^c 53^c \ E_2 = 12'G2'1,$$

where we introduce another type constraint in  $E_1$  (denoted by  $\cdot^c$ ) claiming that fragments 1 and 3 each form a canonical stem.<sup>6</sup>

The type  $E_1$  represents a split into 5 independent parts, where fragment 1 and 3 form the canonical pseudoknot. To simplify the presentation, so far we limited splits to contain only two fragments, but the concept generalizes to more

<sup>6</sup> Note that our split types (with the exception of  $E_2$ ) correspond to the grammar rules given in [19]:  $S \rightarrow \cdot \mid \cdot S \mid S \cdot \mid SS \mid (S) \mid [^k S \{ ^l S \} ^l]$ .

complex splits without any difficulty.  $E_2$  is used to further decompose the stems corresponding to the first and third fragment of  $E_1$ .

All types have at most  $O(m^4)$  instances, resulting in  $O(nm^4)$  time complexity. The type  $E_1$  deserves special attention, because its unconstrained variant has  $O(m^8)$  many instances. However due to the constraints this is reduced to  $O(m^2)$ , since a split of constrained type  $E_1$  is already determined by fixing start and end position of the parent fragment. Because the RNA structures are fix, the start position of the fragment is the start position of a uniquely determined base pair in stem 1 and this uniquely determines the first stem. The second stem is determined analogously by the end position.

In the same way as the structure prediction algorithm finds the best canonical structure for a sequence, the alignment algorithm finds the best canonical alignment. Here, canonical means that a stem of a pseudoknot can only be aligned to another maximally extended stem.

The space complexity is reduced as follows. For all split types except  $E_1$  and  $E_2$  applies again that they only contain degree 1 or atomic fragments. In  $E_1$ , from all fragments of degree 2 only the  $O(n^2)$  canonical instances are considered. In  $E_2$ , the second fragment is size constrained and the first fragment shares its first and last boundary with the splitted fragment. Hence, the  $O(m^4)$  instances of the first fragment can be grouped into groups of  $O(m^2)$  elements that have a common first and last boundary. This reduces the space complexity to  $O(nm^2)$ .

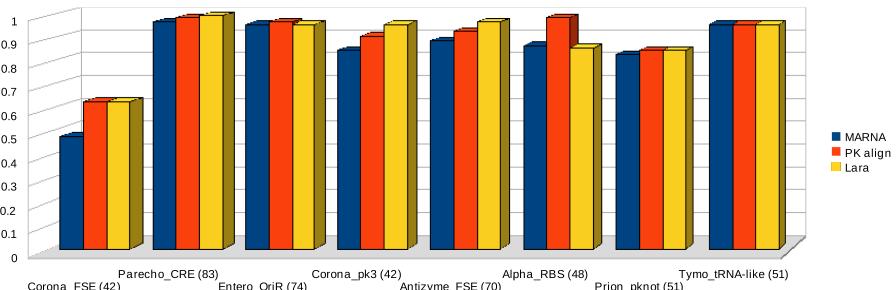
## 6 Results

### 6.1 Accuracy of Pseudoknot Alignment

We use a benchmark set of 8 RNA families of Rfam [30] that are annotated with pseudoknots. Albeit in total Rfam contains 16 such families, we restricted the test set to RNAs with length of at most 125. From each family, we selected the pair of members with the lowest sequence identity, in order to maximally challenge the algorithms. We used the consensus structure (projected to the respective sequence) as structure input, since Rfam does not contain a separate structure for each sequence.

We compared our tool PKalign to MARNA [31] and lara [13]. While lara represents the only other pseudoknot alignment method available so far, the comparison to MARNA allows us to evaluate the benefit of taking pseudoknots into account: MARNA is based on the exact same scoring scheme as PKalign (and we used the same parameter values), but since it is unable to handle pseudoknots, we had to resolve the crossing by removing some base pairs. The accuracy to reproduce the Rfam alignment is measured by the COMPALIGN score. The results are shown in Fig. 5.

The comparison to MARNA shows that taking the pseudoknots into account in general improves the accuracy. The accuracy of PKalign and lara is comparable, the minor differences between their results seem to be caused by differences in the scoring scheme, different choices if several optimal alignments exist and by the fact that PKalign does not yet support affine gap costs.



**Fig. 5.** Comparison MARNA vs. PKalign vs. lara on 8 pseudoknot families. The numbers in brackets give the sequence identity.

## 6.2 Detecting Conserved Pseudoknots

The reliability of pseudoknot de-novo prediction is still very low. Common prediction programs, tend to predict pseudoknots even in pseudoknot free RNA and do not allow to distinguish safely between true pseudoknots and false positives. This behavior could already be observed for pknotsRG [19] in the following small study.

Therefore, it is desirable to increase the specificity of predictions by requiring confirmation due to homologous RNAs. This approach provides much stronger evidence by observing compensatory mutations in conserved crossing base pairs that are predicted in several homologous RNAs. Such a procedure is useful for reliably annotating pseudoknots in unknown RNA, e.g. from genome wide screens for non-coding RNA.

As a preliminary approach towards comparative pseudoknot identification, we suggest a pipeline for detecting potential pseudoknots that starts with a set of homologous RNAs, and performs the following steps: 1.) for each sequence predict locally optimal and suboptimal pseudoknots of the R&G class (using pknotsRG [19] in local mode). 2.) determine candidate pseudoknots that occur at similar positions in  $k$  of our sequences (here,  $k = 3$ ). 3.) using our approach, align the  $k$ -tuples of pseudoknots pairwise all-against-all; this information is used to construct a multiple alignment by T-Coffee [32]. 4.) analyze the alignment for conserved, crossing compensatory mutations.

First, we tested our approach on Rfam data using the same set of 8 families as above. For each family, we randomly selected six sequences for our analysis. We found pseudoknot candidates with crossing compensatory mutations in all of these families. For four families, we could reproduce triplet alignments of the known pseudoknots that showed crossing compensatory mutations for three of the families; an example is given in Fig. 6a. The figure depicts an alignment of the pseudoknotted sub-sequences with start and end position. For each sub-sequence we show the structure predicted by pknotsRG. The last line gives the consensus structure and highlights base pairs of the pseudoknot which are confirmed by crossing compensatory mutations.

a)

X90572.1	6	[[-[[[[[....((....-)).-(((((((..]]])]-]....))))]]))	59
		[[-[[[[[....((....-)).-(((((((..]]])]-]....))))]]))	
X66718.1	6	cu-uguacagaaugguaag-cac-guguaguaggguaca-agcaaccuauugcau	59
		[[[[[[-[.((...-..))...((((((.-]-]])-....))))]]))	
AF058944.1	6	cucuau-cagauugg-augucuugcugcuauaaua-g-augag-aagguaauagcag	58
cons. str.		[[- <b>I</b> I[-[..... <b>I</b> (( <b>CC</b> ( <b>CC</b> .-] <b>I</b> ].... <b>CC</b> ) <b>CC</b> ))	

b)

ci_658349	52	[[[[[[.(((((..]]]).(((((...))))....))))--	99
		[[[[[[.(((((..]]]).(((((...))))....))))--	
cs_658349	55	ucucagggugaaaucugagacgaaacgauucguuuuccuauauuuuc-	104
		[[[[[[.(((((..]]]).((-(((....))-)...-))))-	
od_658349	53	ucucagugugacagcugagaccg-uccuacuggga-cgucuau-uguca-	98
cons. str.		[ <b>I</b> <b>I</b> [[[ <b>I</b> .. <b>I</b> (( <b>CC</b> <b>I</b> )] <b>I</b> ].((-(((....))-)...) <b>CC</b> <b>I</b> ..	

**Fig. 6.** a) Correctly predicted pseudoknot in the Rfam family Corona\_pk3 and its alignment. b) Predicted pseudoknot of potential ncRNA.

The procedure was then applied to the 50 unannotated ncRNA candidates predicted by an RNAAz screen of *Ciona intestinalis* [24]. In this screen, the *C. intestinalis* genome was compared to *C. savignyi* and *O. dioica*, thus per candidate we get three sequences from the three organisms that are analyzed by the above pipeline. In total, we predicted pseudoknot candidates for only 14 of the 50 RNAs; in contrast, pknotsRG predicts pseudoknots in all of the ncRNAs. Fig. 6b shows one prediction by this experiment.

## 7 Conclusions

We presented a general algorithm scheme for pairwise alignment of pseudoknots. This scheme yields an efficient alignment algorithm for arbitrary classes of pseudoknots that can be predicted efficiently by dynamic programming. Moreover, we showed that such an alignment algorithm benefits from restrictions to certain structure classes in the same way as structure prediction algorithms do. This theoretically interesting result actually yields a series of new alignment algorithms for specific pseudoknot classes; for earlier pseudoknot alignment algorithms, it improves time and space complexity.

Our short study for increasing the reliability of pseudoknot prediction by accounting for comparative information is probably the first biologically meaningful application of pseudoknot alignment to biological data and demonstrates the new possibilities due to our method. It points directly to the appealing idea of automatic pseudoknot annotation in unknown, potential ncRNA from genome-wide screens.

**Acknowledgements.** We thank Kristin Reiche for providing data of the Ciona RNAAz screen and our Master student Jörg Bruder for implementing the parsing component of PKalign. M. Möhl is funded by the German Research Foundation.

## References

1. Couzin, J.: Breakthrough of the year. Small RNAs make big splash. *Science* 298(5602), 2296–2297 (2002)
2. Washietl, S., Hofacker, I.L., Lukasser, M., Huttenhofer, A., Stadler, P.F.: Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nat. Biotechnol.* 23(11), 1383–1390 (2005)
3. Staple, D.W., Butcher, S.E.: Pseudoknots: RNA structures with diverse functions. *PLoS Biol.* 3(6), e213 (2005)
4. Xayaphoummine, A., Bucher, T., Thalmann, F., Isambert, H.: Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. *Proc. Natl. Acad. Sci. USA* 100(26), 15310–15315 (2003)
5. Lyngso, R.B., Pedersen, C.N.S.: Pseudoknots in RNA secondary structures. In: *Proc. of the Fourth Annual International Conferences on Computational Molecular Biology (RECOMB 2000)*. ACM Press, New York (2000) BRICS Report Series RS-00-1
6. Do, C.B., Woods, D.A., Batzoglou, S.: CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics* 22(14), e90–e98 (2006)
7. Will, S., Reiche, K., Hofacker, I.L., Stadler, P.F., Backofen, R.: Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLOS Computational Biology* 3(4), e65 (2007)
8. Siebert, S., Backofen, R.: MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics* 21(16), 3352–3359 (2005)
9. Gorodkin, J., Heyer, L., Stormo, G.: Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res.* 25(18), 3724–3732 (1997)
10. Havgaard, J.H., Torarinsson, E., Gorodkin, J.: Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput. Biol.* 3(10), 1896–1908 (2007)
11. Mathews, D.H., Turner, D.H.: Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology* 317(2), 191–203 (2002)
12. Harmanci, A.O., Sharma, G., Mathews, D.H.: Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics* 8, 130 (2007)
13. Bauer, M., Klau, G.W., Reinert, K.: Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics* 8, 271 (2007)
14. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology* 285(5), 2053–2068 (1999)
15. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science* 210, 277–303 (1999)
16. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics* 104, 45–62 (2000)
17. Deogun, J.S., Donis, R., Komina, O., Ma, F.: RNA secondary structure prediction with simple pseudoknots. In: *APBC 2004: Proceedings of the second conference on Asia-Pacific bioinformatics*, Darlinghurst, pp. 239–246. Australian Computer Society, Inc., Australia (2004)

18. Dirks, R.M., Pierce, N.A.: A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput. Chem.* 24(13), 1664–1677 (2003)
19. Reeder, J., Giegerich, R.: Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics* 5, 104 (2004)
20. Evans, P.A.: Finding common RNA pseudoknot structures in polynomial time. In: Lewenstein, M., Valiente, G. (eds.) *CPM 2006. LNCS*, vol. 4009, pp. 223–232. Springer, Heidelberg (2006)
21. Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S.R., Bateman, A.: Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research* 33(Database Issue), D121–D124 (2005)
22. Jiang, T., Lin, G., Ma, B., Zhang, K.: A general edit distance between RNA structures. *Journal of Computational Biology* 9(2), 371–388 (2002)
23. Washietl, S., Hofacker, I.L., Stadler, P.F.: Fast and reliable prediction of noncoding RNAs. *Proc. Natl. Acad. Sci. USA* 102(7), 2454–2459 (2005)
24. Missal, K., Rose, D., Stadler, P.F.: Non-coding RNAs in *Ciona intestinalis*. *Bioinformatics* 21(suppl. 2), ii77–ii78 (2005)
25. Evans, P.A.: Finding common subsequences with arcs and pseudoknots. In: Crochemore, M., Paterson, M. (eds.) *CPM 1999. LNCS*, vol. 1645, pp. 270–280. Springer, Heidelberg (1999)
26. Möhl, M., Will, S., Backofen, R.: Fixed parameter tractable alignment of RNA structures including arbitrary pseudoknots. In: Ferragina, P., Landau, G.M. (eds.) *CPM 2008. LNCS*, vol. 5029, pp. 69–81. Springer, Heidelberg (2008)
27. Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research* 9, 133–148 (1981)
28. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie* 125, 167–188 (1994)
29. Condon, A., Davy, B., Rastegari, B., Zhao, S., Tarrant, F.: Classifying RNA pseudoknotted structures. *Theoretical Computer Science* 320(1), 35–50 (2004)
30. Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., Eddy, S.R.: Rfam: an RNA family database. *Nucleic Acids Research* 31(1), 439–441 (2003)
31. Siebert, S., Backofen, R.: Methods for multiple alignment and consensus structure prediction of RNAs implemented in MARNA. *Methods Mol. Biol.* 395, 489–502 (2007)
32. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology* 302(1), 205–217 (2000)

# Detection of Locally Over-Represented GO Terms in Protein-Protein Interaction Networks

Mathieu Lavallée-Adam<sup>1</sup>, Benoit Coulombe<sup>2</sup>, and Mathieu Blanchette<sup>1,\*</sup>

<sup>1</sup> McGill Centre for Bioinformatics and School of Computer Science  
3775 University, room 332. Montreal (Québec), H2J 3B9, Canada

<sup>2</sup> Institut de recherches cliniques de Montréal  
110, Pine West. Montreal (Québec) H2W 1R7, Canada

**Abstract.** High-throughput methods for identifying protein-protein interactions produce increasingly complex and intricate interaction networks. These networks are extremely rich in information, but extracting biologically meaningful hypotheses from them and representing them in a human-readable manner is challenging. We propose a method to identify Gene Ontology terms that are locally over-represented in a subnetwork of a given biological network. Specifically, we propose two methods to evaluate the degree of clustering of proteins associated to a particular GO term and describe four efficient methods to estimate the statistical significance of the observed clustering. We show, using Monte Carlo simulations, that our best approximation methods accurately estimate the true p-value, for random scale-free graphs as well as for actual yeast and human networks. When applied to these two biological networks, our approach recovers many known complexes and pathways, but also suggests potential functions for many subnetworks.

## 1 Introduction

Gene ontologies provide a controlled, hierarchical vocabulary to describe various aspects of gene and protein function. The Gene Ontology (GO) Annotation project ([2]) is a literature-based annotation of a gene's molecular function, cellular component, and biological processes. GO analyses have become a staple of a number of high-throughput biological studies that produce lists of genes behaving interestingly with respect to a particular experiment. For example, a microarray experiment may result in the identification of a set of genes that are differentially expressed between normal and disease conditions. A GO term (or category)  $\tau$  is said to be over-represented in a given list if the number of genes in the list that are labeled with  $\tau$  is unexpectedly large, given the size of the list and the overall abundance of genes labeled with  $\tau$  in the species under consideration (see tools like GoMiner [34], Fatigo [1], or GoStat [5]). Statistical over-representation is an indication that the GO category is directly or indirectly linked to the phenomenon under study. We say that this kind of set of

---

\* Corresponding author.

differentially expressed genes is *unstructured*, in the sense that all genes within the list contribute equally to the analysis. A slightly more structured approach consists of considering an *ordered* list of genes, where genes are ranked by their "interest" with respect to a particular experiment (e.g degree of differential expression). There, we seek GO terms what are surprisingly enriched near the top of the ranked list. This is the approach taken by the highly popular GSEA method [31], which generalizes this to include many kinds of gene annotations other than GO.

We propose taking this type of analysis one step further and applying GO term enrichment analysis to even more highly structured gene sets: biological networks. In such networks, genes (or their proteins) are vertices and edges represent particular relationships (protein-protein interaction, regulatory interaction, genetic interaction, etc.). Given a fixed biological network  $G$  and a gene ontology annotation database, our goal is to identify every term  $\tau$  such that the genes labeled with  $\tau$  are unexpectedly clustered in the network (i.e. they mostly lie within the same "region" of the network). This local over-representation indicates that  $\tau$  is likely to be linked to the function of that sub-network. Indeed, and unsurprisingly, GO term clustering has been observed to occur in most types biological networks [11][20], and has been used as a criterion to evaluate the accuracy of computational complex or module prediction [21]. However, to our knowledge, the problem of identifying locally over-represented GO terms in a network has never been formulated or addressed before.

Our problem has a number of applications. High-throughput technologies generate large networks (thousands of proteins and interactions) that are impossible to analyze manually. Graph layout approaches (reviewed in [32]), integrated in many network visualization packages such as VisSANT [15] and Cytoscape [28], can help humans extract biological meaning from the data, but revealing all aspects of a complex data set in a single layout is impossible and, often, key components of the network remain unstudied because the layout used did not reveal them visually. Various approaches have been proposed to ease the analysis of biological networks, including packages performing graph clustering and path analysis (e.g. NeAT [28][6]). Several methods have been proposed to identify pathways [30] within PPIs or combine expression data with PPI networks to infer signaling pathways [26]. Another popular strategy starts by identifying dense subnetworks within the network (using, for example, MCL [12]), and then evaluates various biological properties of the subnetwork, including GO term enrichment [27].

Our proposed approach identifies subsets of genes that share the same GO annotation and that are highly interconnected in the network, thus formulating the hypothesis that the function of the subnetwork is related to that GO annotation. This reduces the complexity of the data and allows easier grasp by human investigators. Our approach could be extended to help function prediction: genes with incomplete functional annotation that are found to be highly interconnected with a set of genes of known function can be expected to share that function [9][29].

In this paper, we define formally the problem of identification of locally-enriched GO categories for an unweighted, undirected interaction network. We start by defining two measures of clustering of a set of genes in a given interaction graph. We then discuss the critical question of assessing the statistical significance of the local clustering scores using analytical approaches (empirical approaches for shortest path distance significance have been proposed previously [25]) of a given GO term within the network, under a null hypothesis where vertices are selected randomly. We show that the exact computation of this probability is NP-hard, but we provide several efficient approximation methods. These p-value approximation methods are shown to be accurate on random scale-free graphs, as well as on large-scale yeast [19] and human [16,10] protein-protein interaction networks. Our analysis identifies regions of these two networks with known function. It also suggests interesting functions for regions of the network that are currently poorly understood.

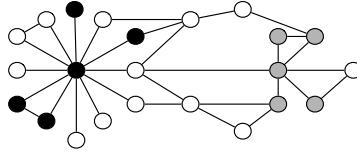
## 2 Methods

We are looking for GO terms whose distribution across a given network is non-random. In particular, are interested in finding terms that are tightly clustered within the network. Let  $G = (V, E)$  be an undirected, unweighted graph, where  $V$  is a set of  $n$  proteins and  $E$  is a set of pairwise interactions between them. The Gene Ontology project assigns to each gene a set of functional annotations, using a controlled vocabulary. For a given GO term  $\tau$ , let  $V(\tau) \subseteq V$  be the subset of the proteins annotated with that term. Our goal is to investigate, for every possible term  $\tau$ , whether  $V(\tau)$  is particularly clustered in  $G$ , which would hint to the fact that  $\tau$  is particularly relevant to the function of that subgraph. To this end, we introduce two measures of clustering (Section 2.1) and show how to assign statistical significance to them (Section 2.2).

### 2.1 Measures of Clustering on a Network

A number of approaches have been proposed to measure the clustering of a set of vertices within a given graph, and to identify dense clusters (e.g. MCL [12]; see [2] for a review). We focus on two simple but effective clustering measures, for which the statistical significance can be accurately approximated analytically.

**Total pairwise distance.** Given two vertices  $u$  and  $v$  in  $V$ , let  $d_G(u, v)$  be the length of a shortest path from  $u$  to  $v$  in  $G$ . Since  $G$  is undirected,  $d_G$  is symmetric. The distance matrix  $d_G$  can be computed in time  $O(|V|^3)$  using the Floyd-Warshall algorithm [14,33]. Let  $W$  be a subset of  $V$ . Then, the *total pairwise distance* for  $W$  is defined as  $TPD(W) = \sum_{u, v \in W, u < v} d_G(u, v)$ . If most of the vertices in  $V(\tau)$  are in the same region of the graph (e.g. the gray or black vertices in Figure 1), then  $TPD(V(\tau))$  will be smaller than that of most random subsets of  $|V(\tau)|$  vertices and  $\tau$  will be reported as potentially interesting.



**Fig. 1.** Example of a toy PPI network. The black and gray subsets of vertices obtain the same Total Pairwise Distance (13), but the gray subset obtains a higher Probability of Stopping within the Family (PSF).

**Random-walk based similarity.** One issue with the TPD clustering measure is that it does not take into consideration the degree of the nodes on the path between the two proteins, in such a way that, for example, the two sets of proteins shown in black and gray in Figure 1 will get the same total pairwise distance (and, eventually, the same p-value), although intuitively the gray cluster appears more interesting. In addition, if the vertices in  $W$  form more than one dense subgraphs, and these clusters are far away from each other, the TPD measure may not reveal anything unusual. We introduce an alternative to the total pairwise distance, which we call the Probability of Staying within the Family (PSF) clustering measure. This random-walk based similarity measure shares a relationship with diffusion kernels [18]. The PSF for a subset of vertices  $W$  is defined based on the following random process (similar to that modeled by MCL [12]), parameterized by a user-defined probability  $p$ : (i) Randomly select a vertex from  $W$  as a starting point; (ii) When at vertex  $u$ , stop with probability  $p$ , or, with probability  $1 - p$ , continue to a vertex  $v$  uniformly chosen from the neighbors of  $u$ . Then,  $PSF_p(W)$  is defined as the probability that the vertex where the process stops is an element of  $W$ . We note first that this process does make a difference between the two subsets in Figure 1 and will also assign a high score to a subset  $W$  that would consist of several dense but widely separated clusters.

If  $A_G$  is the adjacency matrix of  $G$  and  $deg_G(u)$  is the degree of vertex  $u$ , then the transition probability matrix  $T_G$  for this random walk is defined as  $T_G(u, v) = A_G(u, v)/d_G(u)$ , and the probability  $P_{u,v}$  of stopping at vertex  $v$ , starting from vertex  $u$ , is given by

$$P_{u,v} = \sum_{i=0}^{+\infty} p(1-p)^i ((T_G)^i)(u, v).$$

Thus,  $PSF_p(W) = \frac{\sum_{u,v \in W} P_{u,v}}{|W|}$ . So, as for the total pairwise distance, the PSF measure is a sum of pairwise scores, with  $s_G(u, v) = P_{u,v}/|W|$ .

The methods proposed in Section 2.2 to assess statistical significance apply to both TPD and PSF.

## 2.2 Measuring the Statistical Significance

Given a matrix  $M_{|V| \times |V|}$  containing pairwise distances ( $d_G$ ) or similarities ( $s_G$ ), we consider the random variable obtained as follows. Let  $R = \{r_1, r_2, \dots, r_k\} \subseteq$

$\{1, \dots, n\}$  be a randomly selected subset of proteins of cardinality  $k$ . We are interested in the distribution of the random variable  $S_k = \sum_{i,j \in R, i < j} M_{i,j}$ . When using the TPD clustering measure, the p-value for GO term  $\tau$  will be obtained as  $\Pr[S_{|V(\tau)|} \leq TPD(V(\tau))]$ , whereas when using the PSF clustering measure, the p-value will be obtained as  $\Pr[S_{|V(\tau)|} \geq PSF_p(V(\tau))]$ . Note that there is no need to adjust the p-values for  $k$  since we are analyzing a different distribution for each  $S_k$ .

*A note on complexity.* We first observe that computing the exact distribution of  $S_k$  when  $M = d_G$  is NP-hard. Indeed,  $\Pr[S_k = \binom{k}{2}]$  is non-zero if and only if  $G$  contains a  $k$ -clique. Therefore, we cannot expect an exact polynomial time algorithm. Although more difficult to prove, the same is likely true for PSF. We thus investigate three approaches that give approximations to the desired probability distributions.

### 2.3 Normal Approximation

Being a sum of  $\binom{k}{2}$  random variables, the distribution of  $S_k$  should converge to a normal distribution as  $k$  and  $|V|$  become large (Central Limit Theorem), if these random variables were independent. Although these variables are clearly not independent (for example, in the case of TPD, they must satisfy the triangle inequality), it turns out that the normality assumption sometimes yields a useful approximation to the true distribution. The expectation of  $S_k$  can be calculated exactly in time  $O(|V|^2)$ . Let  $E[S_2] = \frac{\sum_{1 \leq a < b \leq n} M_{a,b}}{\binom{n}{2}}$  be the average pairwise score in  $M$ . Then

$$E[S_k] = \binom{k}{2} \cdot E[S_2]$$

The variance of  $S_k$  is more challenging to obtain. We have  $Var[S_k] = E[S_k^2] - E[S_k]^2$ , where

$$\begin{aligned} E[S_k^2] &= E[(\sum_{a,b \in R, a < b} M_{a,b})^2] \\ &= E[\left\{ \sum_{a,b \in R, a < b} (M_{a,b})^2 \right\} + \left\{ 2 \sum_{a,b,c \in R, a < b < c} M_{a,b}M_{a,c} + M_{a,b}M_{b,c} + M_{a,c}M_{b,c} \right\} + \\ &\quad 2 \left\{ \sum_{a,b \in R, a < b} \sum_{c < d \in R, c \neq a, b, d \neq a, b} M_{a,b}M_{c,d} \right\}] \\ &= \binom{\binom{n}{2}}{2} \sum_{1 \leq a < b \leq n} (M_{a,b})^2 + 2 \binom{\binom{n}{3}}{2} \left\{ \sum_{1 \leq a < b < c \leq n} M_{a,b}M_{a,c} + M_{a,b}M_{b,c} + M_{a,c}M_{b,c} \right\} + \\ &\quad 2 \left\{ \sum_{a,b \in R, a < b} M_{a,b} \sum_{c < d \in R, c \neq a, b, d \neq a, b} M_{c,d} \right\} \end{aligned}$$

The running time of the variance computation is thus  $O(n^4)$ , which, in many cases, is prohibitive. However, when  $a \neq b \neq c \neq d$ ,  $M_{a,b}$  is nearly independent from  $M_{c,d}$ , so

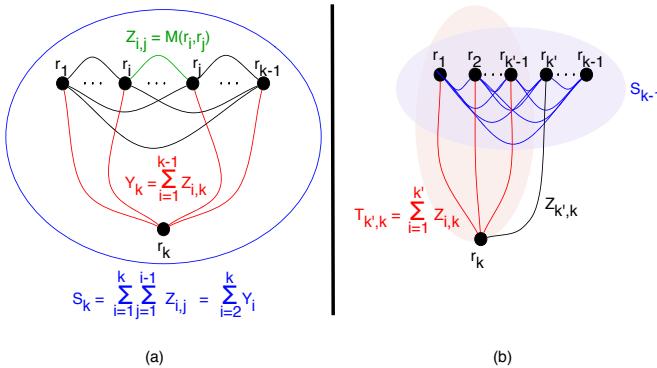
$$E[S_k^2] \approx \frac{\binom{k}{2}}{\binom{n}{2}} \sum_{1 \leq a < b \leq n} (M_{a,b})^2 + 2 \frac{\binom{k}{3}}{\binom{n}{3}} \left\{ \sum_{1 \leq a < b < c \leq n} M_{a,b} M_{a,c} + M_{a,b} M_{b,c} + M_{a,c} M_{b,c} \right\} + 2 \binom{k}{2} \left\{ \binom{k}{2} - 2k + 3 \right\} E[S_2]^2$$

We call this approach the normal approximation method.

## 2.4 Convolution-Based Approaches

Considering again a random subset of vertices  $R = \{r_1, r_2, \dots, r_k\}$ , we define the random variables  $Z_{i,j} = M_{r_i, r_j}$ , for  $1 \leq i < j \leq n$  and  $Y_i = \sum_{j=1}^{i-1} M_{r_j, r_i}$ , for  $i = 2 \dots k$  (refer to Figure 2). In this section, we assume that the scores in  $M$  are integers. This will always be the case when  $M = d_G$ . When  $M = s_G$ , we assume that  $s_G$  has been appropriately discretized to integers. Observe that  $S_k = \sum_{i=1}^k \sum_{j=1}^{i-1} Z_{i,j} = \sum_{i=2}^k Y_i$ . The random variable  $S_k$  is a sum of  $\binom{k}{2}$  random but dependent variables. If we ignored the dependencies, the distribution of  $S_k$  could be obtained as the  $\binom{k}{2}$ -fold self-convolution of the discrete distribution  $f_G$ , where  $f_G(a) = \sum_{1 \leq i < j \leq |V|} \mathbf{1}_{a=M_{i,j}} / \binom{|V|}{2}$  is the fraction of entries in  $M$  with value  $a$ . This turns out to produce a very poor approximation of the distribution of  $S_k$ , severely underestimating the correct probability for small values of  $S_k$ . We can improve the situation by modeling some of the dependencies. Again, the family of  $Y$  random variables are dependent: in particular, if  $S_{k-1} = \sum_{i=2}^{k-1} Y_i$  is small, i.e.  $r_1, \dots, r_{k-1}$  form a tight cluster, then the variance of  $Y_k$  is increased, because the variables  $Z_{*,k}$  are highly dependent on each other (e.g. if  $Z_{i,k}$  is small, then  $Z_{i',k}$  is also likely to be small, because  $i$  and  $i'$  belong to the same tight cluster). We consider two approaches to the problem: the first calculates nearly exactly the distribution of the  $Y_i$ 's but ignores their dependencies, while the second models the dependencies more accurately but is less accurate at the level of each distribution.

*The  $Y$ -convolution method:* Let  $g_i(a) = \sum_{j=1}^n \mathbf{1}_{a=M_{i,j}} / (n-1)$  be the fraction of pairs of vertices  $(i, *)$  with score  $a$  and let  $g_i^{(l)}$  be the  $l$ -fold self-convolution of  $g_i$ . Then,  $y_l(a) = \Pr[Y_l = a] \approx 1/n \sum_{i=1}^n (g_i^{(l-1)})(a)$  (this is an approximation because the convolution models a situation where the random subset  $R$  would be allowed to repeatedly pick the same pair of vertices). Assuming the independence of the  $Y_i$ 's, the distribution of  $S_k$  would be obtained by the convolution  $y_2 * y_3 * \dots * y_k$ . We will refer to this approximation as the  $Y$ -convolution method. Its running time is  $O(|V|^2 k^2 d^2)$ , where  $d$  is the diameter of  $G$ , although the use of Fast Fourier transforms to compute convolutions may yield significant improvements.



**Fig. 2.** Definition of the variables used in the Convolution approaches

*The triangle decomposition methods:* An alternate approach is to use a dynamic programming algorithm to better model dependencies (refer to Figure 2 (b)):

$$\Pr[S_k = a] = \Pr\left[\sum_{i=2}^k Y_i = a\right] = \begin{cases} \sum_{a'=1}^a \Pr[S_{k-1} = a'] \cdot \Pr[Y_k = a - a' | S_{k-1} = a'] & \text{if } k > 1 \\ 0 & \text{if } k = 1, a \neq 0 \\ 1 & \text{if } k = 1, a = 0 \end{cases}$$

Define  $T_{k',k} = \sum_{j=1}^{k'} Z_{j,k}$ , for  $1 \leq k' < k$ , so that  $Y_k = T_{k-1,k}$ . The term of the form  $\Pr[Y_k = b | S_{k-1} = c] = \Pr[T_{k-1,k} = b | S_{k-1} = c]$  is calculated using another convolution-based dynamic programming algorithm.

$$\Pr[T_{k',k} = b | S_{k-1} = c] = \begin{cases} \sum_{d=1}^b \Pr[T_{k'-1,k} = d | S_{k-1} = c] \cdot \Pr[Z_{k',k} = b - d | S_{k-1} = c, T_{k'-1,k} = d] & \text{if } 2 \leq k' < k \\ \Pr[Z_{1,k} = b | S_{k-1} = c] & \text{if } k' = 1 \end{cases}$$

It is most likely impossible to calculate exactly and in polynomial time  $\Pr[Z_{k',k} = b - d | S_{k-1} = c, T_{k'-1,k} = d]$ , as otherwise the derivation above would give the exact probability distribution for  $S_k$ , which we have shown to be an NP-hard problem. Instead, we boil down the information in the condition  $(S_{k-1} = c, T_{k'-1,k} = d)$  to a simpler condition for which the conditional probability is easier to compute. Notice that if  $S_{k-1} = c$ , the average pairwise distance among  $r_1, \dots, r_{k-1}$  is  $l_1 = c/(k-1)$ . Also, if  $T_{k'-1,k} = d$ , then the average pairwise distance between  $r_k$  and  $r_1, \dots, r_{k'-1}$  is  $l_2 = d/(k'-1)$ .

*Rounding approach:* We assume that the desired condition can be represented as the condition  $Z_{1,k'} = Z_{2,k'} = \dots = Z_{k'-1,k'} = [l_1], Z_{1,k} = Z_{2,k} = \dots = Z_{k'-1,k} = [l_2]$ , where  $[l_1]$  is the rounding of  $l_1$ , and similarly for  $l_2$ . The information on  $Z_{k',k}$  thus comes in the form of  $k'-1$  nearly independent pairs  $(Z_{i,k'} = [l_1], Z_{i,k} = [l_2])$ . Let  $t(a, b, c)$  be the number of triplets  $1 \leq i < j < k \leq n$  such that  $M(i, j) = a, M(i, k) = b, M(j, k) = c$ . Assuming the independence of the  $k'-1$  conditions, the desired posterior probability of  $Z_{k',k}$  is obtained as:

$$\begin{aligned} \Pr[Z_{k',k} = b-d | S_{k-1} = c, T_{k'-1,k} = d] &= \Pr[S_{k-1} = c, T_{k'-1,k} = d | Z_{k',k} = b-d] \cdot \Pr[Z_{k',k} = b-d] / \zeta \\ &\approx \left( \frac{t([l_1], [l_2], b-d)}{t(*, *, b-d)} \right)^{k'-1} \cdot f_G(b-d) / \zeta, \end{aligned}$$

where  $\zeta$  is a normalizing constant that does not need to be computed (it is sufficient to normalize the distribution to make it sum to 1).

*Interpolation approach:* The rounding procedure yields a rather crude modeling of the actual posterior probability, especially when  $l_1$  or  $l_2$  are far from  $[l_1]$  or  $[l_2]$  respectively. A better modeling may be obtained as follows. Instead of assuming that all  $k' - 1$  condition pairs have the same values  $[l_1]$  and  $[l_2]$ , we assume  $N_{00} = \text{frac}(l_1) \cdot \text{frac}(l_2) \cdot (k' - 1)$  pairs have values  $([l_1], [l_2])$ ,  $N_{01} = \text{frac}(l_1) \cdot (1 - \text{frac}(l_2)) \cdot (k' - 1)$  pairs have values  $([l_1], [l_2])$ ,  $N_{10} = (1 - \text{frac}(l_1)) \cdot \text{frac}(l_2) \cdot (k' - 1)$  pairs have values  $([l_1], [l_2])$ , and  $N_{11} = (1 - \text{frac}(l_1)) \cdot (1 - \text{frac}(l_2)) \cdot (k' - 1)$  pairs have values  $([l_1], [l_2])$ . We thus approximate:

$$\begin{aligned} \Pr[Z_{k',k} = b-d | S_{k-1} = c, T_{k'-1,k} = d] &\approx \\ \left( \frac{t([l_1], [l_2], b-d)}{t(*, *, b-d)} \right)^{N_{00}} \cdot \left( \frac{t([l_1], [l_2], b-d)}{t(*, *, b-d)} \right)^{N_{10}} \cdot \left( \frac{t([l_1], [l_2], b-d)}{t(*, *, b-d)} \right)^{N_{01}} \cdot \left( \frac{t([l_1], [l_2], b-d)}{t(*, *, b-d)} \right)^{N_{11}} \cdot f_G(b-d) / \zeta \end{aligned}$$

Both triangle convolution approaches run in time  $O(k^6 d^3 + |V|^3)$ , where  $d$  is the diameter of  $G$ .

## 2.5 Identification of Core Subgraphs

If a GO term  $\tau$  obtains a small p-value from one of the methods described above, this means that the genes in  $V(\tau)$  are unexpectedly clustered within  $G$ . This does not, however, mean that every gene in  $V(\tau)$  belongs to that dense cluster, but only that a significant subset of  $V(\tau)$  does. We call the  $\text{core}(\tau) \subseteq V(\tau)$  the subset of  $V(\tau)$  that contributes the most to its statistical significance, i.e. the set of genes in  $V(\tau)$  that is the most significantly clustered. In most situations, it is  $\text{core}(\tau)$ , rather than  $V(\tau)$ , that sheds the most light on the function of a portion of a network. We use a simple greedy strategy to reduce  $V(\tau)$  to  $\text{core}(\tau)$ , by iteratively removing from  $V(\tau)$  the protein that improves the p-value the most (which is necessarily the protein that decreases (resp. increases) the TPD (resp. the PSF) the most), until no further improvement is possible. This algorithm, which runs in  $O(|V(\tau)|^3)$ , does not guarantee optimality but generally succeeds at identifying the key component of  $V(\tau)$ . The results presented in Section 3.2 are the core of the GO terms that obtained good p-values.

## 2.6 Implementation Considerations

The implementation of some of the four approximation schemes described in this section proves quite technically challenging, with issues of numerical precision arising for the two triangle convolution. Our crude approach to the problem is

to make sure that, at every step, the intermediate probability distributions are properly normalized to sum to 1, although more subtle approaches would certainly improve our accuracy. Another issue is the time and memory required for the computation of the triangle convolution approaches, which require the storage of numerous large intermediate tables, currently limiting their utilization to the computation of p-values for values of  $k$  less than 25. Program optimizations were required to accelerate the running time for the triangle convolution approaches. They consist in stopping the computations of a distribution for a given  $S_k$  when the only probabilities left to compute are those at the right tail of the distribution that are smaller than the 64-bit double precision.

## 3 Results

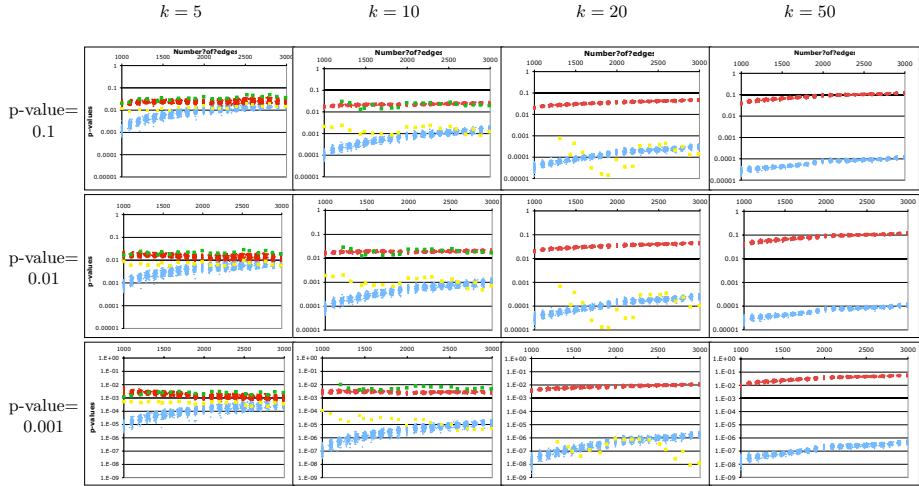
### 3.1 Accuracy of p-Value Approximation Methods

The accuracy of our four p-value approximation schemes can be assessed by Monte Carlo simulations: for a given graph  $G$ , repeatedly sample randomly a subset of  $k$  vertices and compute the sum of pairwise scores to eventually obtain an unbiased estimate of the true distribution. The limit of this approach is of course that the accuracy of the estimation depends on the number of samples, making small p-values difficult to estimate quickly.

We have measured the accuracy of our approximation approaches on both simulated and actual biological networks. Protein-protein interaction networks have been reported to be accurately modeled by scale-free random graphs [3], although geometric random graphs have also been used [24]. We randomly generated scale-free graphs with 1000 vertices and a number of edges ranging from 1000 to 3000. In total, 2100 random graphs were generated. The distributions of the TPD and PSF score were estimated empirically, using  $10^6$  samples, for each graph and each value of  $k = 5, 10, 20, 50$ . For each combination, critical values  $Z_{0.1}$ ,  $Z_{0.01}$ , and  $Z_{0.001}$  were estimated as being the value of TPD and PSF that obtains the empirical p-value 0.1, 0.01, and 0.001, respectively. Each of the four analytical approximation methods<sup>1</sup> were then used to estimate the p-values for  $Z_{0.1}$ ,  $Z_{0.01}$ , and  $Z_{0.001}$ . Figures 3 and 4 reports the accuracy of the p-values produced by each of our methods for the TPD and PSF clustering measures, for the target p-values 0.1, 0.01, and 0.001, and for  $k = 5, 10, 20, 50$ . We start by observing that although our p-value approximation methods apply in principle to both the TPD and PSF clustering measures, specificities of these data sets result in our methods behaving quite differently. This is due to the fact that the similarity scores that constitute the PSF clustering scores exhibit much stronger inter-dependencies than the pairwise distances that constitute the TPD clustering score, resulting in worse approximations when independence is assumed. Our observations are summarized below.

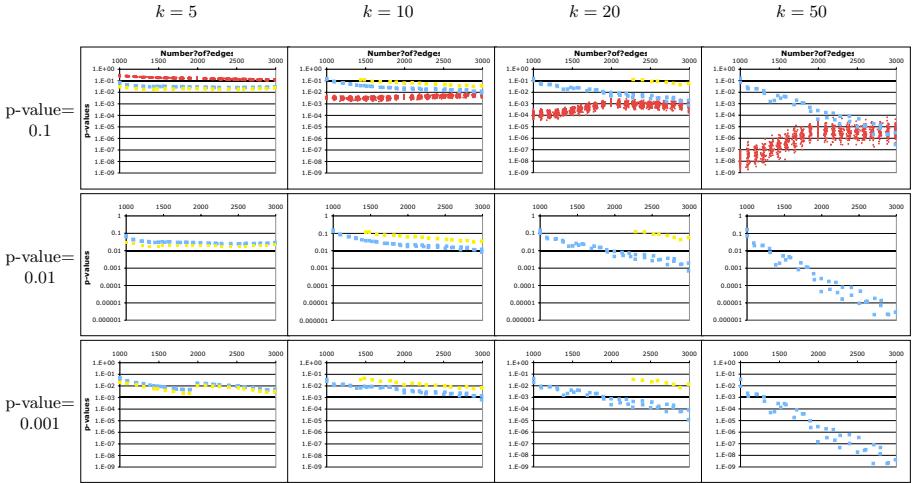
---

<sup>1</sup> Note that the triangle decomposition with interpolation approximation was not performed for PSF because of its high memory and running time requirements.



**Fig. 3.** P-values predicted by our four approximation schemes (Normal: red; Y-convolution: blue; Triangle convolution with rounding: yellow; Triangle convolution with interpolation: green) for the TPD clustering measure. Each data point records the approximated p-value (y-axis) for the TPD score that obtained the given empirical p-value (0.001, 0.01, 0.1), on a random scale-free graph with 1000 vertices and the given number of edges (x-axis). The triangle convolution with rounding method was too slow to be evaluated for  $k > 20$ , and that with interpolation could only be run for  $k = 5$  and  $k = 10$ .

- **Y-convolution.** In the case of TPD, this method severely underestimates small p-values, by a factor ranging from 2 to 100 for  $k = 5$  to more than  $10^4$  for  $k = 50$ . This due to the fact that dependencies in the graph are greatly underestimated. However, the approximation improves with the edge density. On the contrary, the method works quite well on PSF clustering for graphs with low edge density, but it severely underestimates p-values of highly connected graphs.
- **Normal approximation.** This approximation obtains much better results than the Y-convolution approximation in the case of TPD clustering, producing p-values that generally slightly over-estimate the correct p-value (1- to 3-fold for small  $k$ , 10- to 50-fold for  $k=50$ ). Surprisingly, although, for small  $k$ , the quality of the approximation improves with the edge density, the opposite trend is observed for larger  $k$ . However, for PSF clustering, this yields an extremely poor approximation for all values of  $k$ , erring by a factor ranging from  $10^{10}$  to  $10^{60}$  for a true p-value of 0.001.
- **Triangle decomposition with rounding.** We found that this method is an improvement to the Y-convolution approximation for TPD clustering since it does not underestimate as much p-values for small  $k$  (factor ranging from 2 to 10 for  $k = 5$  and from 10 to 100 for  $k = 10$ ). However, it behaves more irregularly for  $k = 20$ , underestimating the p-values by a factor greater than 100. This approach also yield good approximations for PSF clustering,

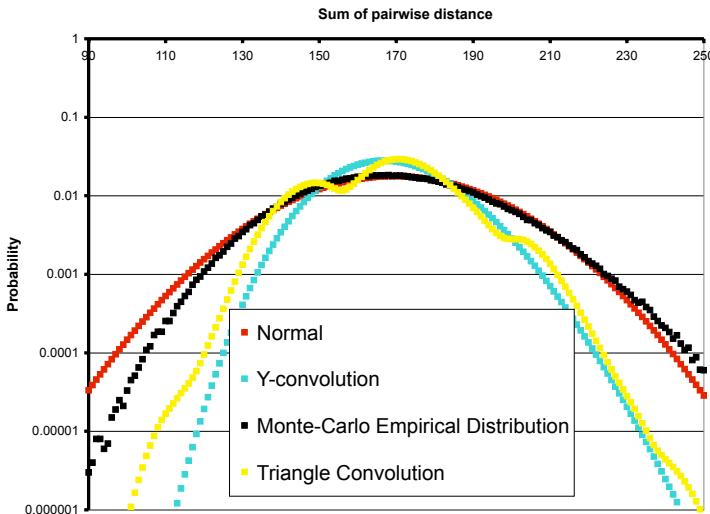


**Fig. 4.** P-values predicted by three approximation schemes (Normal: red; Y-convolution: blue; Triangle convolution with rounding: yellow) for the PSF clustering measure. See caption of Figure 3. The triangle convolution with rounding method was too slow to be evaluated for  $k > 20$  and some graphs for  $k = 20$ . The triangle convolution with interpolation was too slow for all  $k$ . The Normal approximation method produced p-value estimates that were too poor to show on these graphs, usually erring by a factor of  $10^{10}$  or more.

overestimating small p-values for any  $k$  by a small margin. Interestingly, for both clustering measures, the accuracy of this approximation does not seem to be affected by the edge density of the network.

- **Triangle decomposition with interpolation.** The results obtained from this method on TPD clustering are comparable to the normal approximation estimation. For p-values 0.01 or less, computed p-values are slightly overestimating the correct p-values (1- to 4- fold for small  $k$ ). It sometimes even provides a tighter upper bound on the correct p-values. Again the accuracy of the p-value estimation for this method is not influenced by the edge density. We were unable to use this approximation for PSF because of high running time and memory requirements of the method.

Overall, we conclude that given how quickly it can be computed, the normal approximation approach is the best tradeoff between running time and accuracy for TPD. However, the quality of that approximation degrades with the edge density, which is not the case for the two Triangle convolution approaches. This is an important point since we expect protein-protein interaction networks to gain in edge density as new high-throughput assays become available. The Triangle convolution approach is also the most accurate for PSF. It is the only method providing tight upper bounds on p-values even for large  $k$  in highly connected graphs. However, because of its intensive use of memory and slow running time,



**Fig. 5.** Empirical and approximated TPD distributions for the yeast PPI network, for  $k=10$

it is hard to obtain p-value approximations for very large  $k$ . Since it produces p-value approximations in a much more reasonable time, the Y-convolution method can be used in this situation.

Our results on two larger actual PPI networks in yeast [19] and human [16] (see Section 3.2) largely confirm our observations on random graphs. Figure 5 shows the complete TPD distributions (for  $k = 10$ ) obtained by Monte Carlo simulations, as well as each of our approximation methods, for the Krogan et al.'s yeast PPI network, which consists of more than 2500 proteins and 7000 interactions.

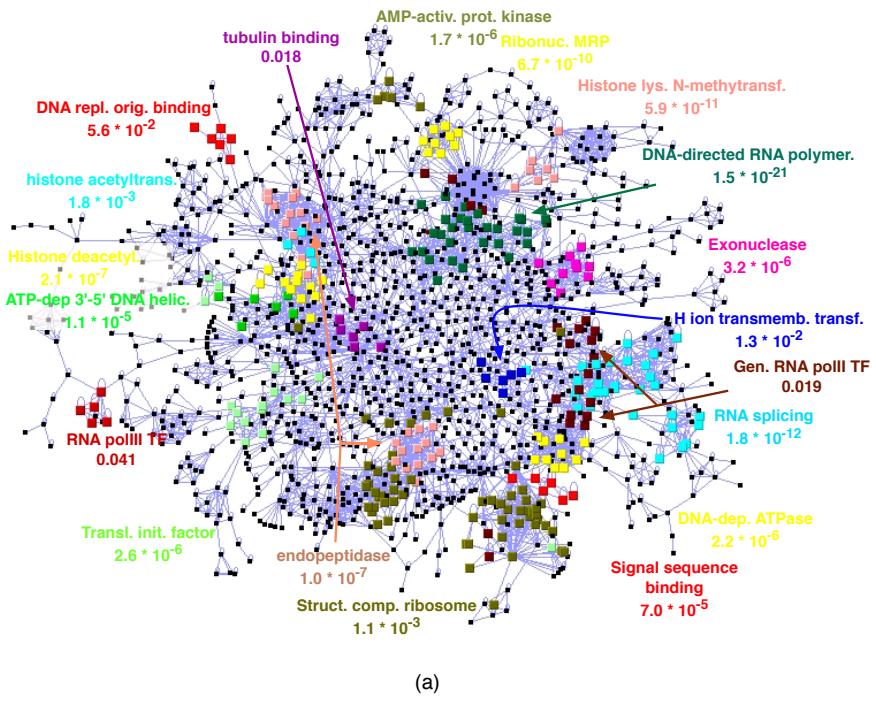
Of the four approximation methods proposed, the fastest is the normal approximation (Table II). The Y-convolution method is approximately 10-fold slower, while the two triangle-based convolution approaches are several orders of magnitude slower. Note that for PSF, Triangle convolution with interpolation runs several order of magnitude slower than the values presented. However, code optimization is likely to yield over the next months and significant speedups are likely to be obtained.

### 3.2 Biological Analyses

We first applied our analysis using TPD to the yeast protein-protein interaction data set produced by Krogan et al. [19]. We analyzed the largest connected component of their "core" network, which consists of 2559 proteins and 7037 interactions. Of the 299 GO terms present more than twice in the network, 91 obtained a normal approximation (conservative) p-value below 0.05 (corresponding to a  $FDR = \frac{299 \times 0.05}{91} \approx 16\%$ ), and 42 obtain a p-value below 0.001

**Table 1.** Approximate running time, in minutes, to calculate one clustering p-value for a 1000-vertex scale-free graph with 2000 edges, for the TPD clustering measure. This running time is representative of that obtained on larger graphs.  $10^6$  samplings were performed for the Monte Carlo simulations.

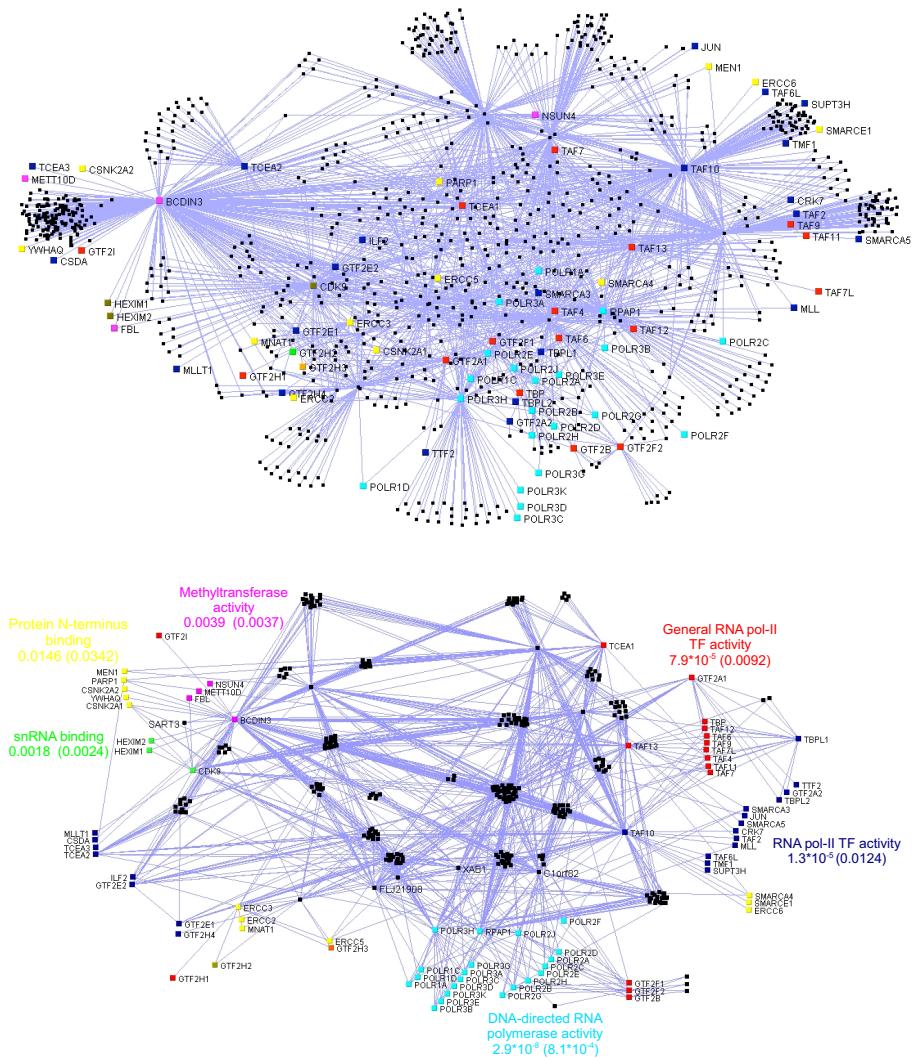
	$k = 10$	$k = 20$	$k = 50$
Monte Carlo simulation	2	5	20
Normal	0.3	0.7	1
Y-Convolution	1	5	15
Triangle with rounding	2	300	>1000
Triangle with interpolation	5	600	>1000



(a)

**Fig. 6.** Yeast PPI network from Krogan et al. [19], annotated with the cores of some of the GO categories with significant clustering. The p-values given were obtained using the Normal approximation approach, which is almost always conservative. For readability, not all significant GO categories are shown.

( $FDR = \frac{299 \times 0.001}{42} \approx 0.7\%$ ). As seen on Figure 6, the GO terms with significant p-values allow the automated annotation of much of the network. For many of the GO terms reported, our results reflect known protein complexes (e.g. ribosome, ribonuclease MRP, general pol-II transcription factors, etc.). Other clusters, often the larger, more diffuse ones, do not correspond to complexes but rather contain proteins that interact with many of the same partners (e.g. the translation



**Fig. 7.** (Top) Human PPI network from Jeronimo et al. [16], laid out using the “relaxed” automatic layout procedure of VisAnt [15]. Groups of protein with a significant PSF clustering p-value are highlighted in colors. (The Triangle convolution was used when the group size was small enough; otherwise, the Y-Convolution was used. Monte Carlo estimated p-value are between parentheses). (Bottom) Network laid out manually to highlight the connectivity of the proteins within each GO category reported (to improve readability, proteins that do not belong to any shortest path between pairs of proteins of the selected groups are not shown). GTF2H3 (in orange) is part of both red and yellow groups. GTF2H2 (in khaki green) is part of both the yellow and blue groups. Clearly, without the information provided by our GO clustering approach, the PPI network showed at the top would be hard to interpret.

initiation factors, or the signal sequence binding proteins). While most GO terms for a single, dense cluster, some, such as the structural components of the ribosome, the geneal RNA pol-II TFs, and the endopeptidases, as broken into two or three dense subgroups. Many of the fundamental functional interactions between groups of proteins of different function immediately stand out, for example the interplay between histone deacetylases (yellow), histone acetyltransferases (in cyan), and ATP-dependent 3'-5' DNA helicases (in green). The annotated network is clearly more interpretable and readily allows the formulation of specific hypotheses about the function of various unannotated proteins and of the various interactions observed. See Supplementary material for complete results.

Finally, we analyzed a human protein-protein interaction network published by Jeronimo *et al.* [10] using PSF. The network contains 1053 proteins and 2014 interactions, built from 32 tagged proteins and their interactors in the soluble fraction of HEK293 cells. The tagged proteins are predominantly proteins related to the (extended) transcription machinery. As can be seen from Figure 7(a), the network is quite dense and existing automated layout systems fail to reveal much of the biological information contained in the graph. We ran our analyses on the network to identify which of the 185 GO categories present more than twice in the graph show unexpected clustering. 24 GO categories obtained p-values below 0.05 (FDR =  $\frac{185 \times 0.05}{24} \approx 38, 5\%$ ; see Supplementary material). Genes belonging to some of these categories are colored coded in Figure 7(b) (several categories are somewhat redundant; only one representative per group is shown). When the graph is manually laid out to highlight the connectivity among the selected protein groups (Figure 7(b)), the role of several subnetworks is clearly revealed. For example, we can easily identify subunits of the RNA polymerase I, II and III, classified by GO as "DNA-directed RNA polymerase activity", which are clustered together. We also notice that RPAP1 is tightly connected to the POLR2 subunits within that cluster. This corroborates the observation of Jeronimo et al. where RPAP1, XAB1, C1ORF82 and FLJ21908 (now referred as RPAP2 and RPAP3 respectively) are forming an interface between the RNA polymerase II subunits and some molecular chaperone and prefoldins. We can also see that our method, by highlighting this GO term, facilitated the visualization of the interactions between the POLR2 subunits with the XAB1, RPAP2 and RPAP3 proteins. Hexamethylene bis-acetamide inducible (HEXIM) proteins were also found to be clustered with cyclin-dependent kinase 9 (CDK9). Originally, CCNT1, member of the P-TEFb complex with CDK9 [23] was also included in this clustering but got removed by the greedy algorithm improving the p-value of the GO term "snRNA binding". Interestingly, HEXIMs are known to be inhibitors of the cyclin-dependent kinase activity of P-TEFb [18]. In addition, BCDIN3 (also known as MEPCE) and SART3 which are proteins part of the 7SK snRNP complex, itself containing P-TEFb, are closely associated with HEXIMs and CDK9 [16,10]. Finally, numerous TATA box binding protein (TBP)-associated factors (TAFs) and general transcription factor II (GTF2s), all sharing the "general RNA polymerase II transcription factor activity" GO function, were found to be significantly clustered. Many of these

TAFs and GTF2s are interacting with TBPL1, another protein playing a key role in transcription [22].

## 4 Discussion and Future Work

The idea described in this paper, of seeking gene attributes that cluster within a given network, can be used to annotate PPI networks with any type of gene features. Besides gene ontologies, we are currently expanding our tool to use protein domains from the PFAM database [13], pathways from the KEGG database [17], and gene expression data. Indeed, any annotation coming in the form of gene sets can be used to annotate the network, including, for example, those collected through the laudable efforts of the GSEA [31] team.

In the future, we will try to improve the accuracy and efficiency of our approximation algorithm. We will also seek provable approximation bounds for the p-value estimation problem. Currently, one of the main computational issues is that some of our best approximation methods are quite slow and require a lot of memory. More efficient implementations will soon be made available.

In this paper, we only studied the simplest version of a family of interesting problems. A number of extensions will be considered. One important generalization is to consider weighted graphs, where edge weights represent either the confidence or the strength of the interaction. Both clustering measures (TPD and PSF) are easily adapted to this case, and so will most of the p-value approximation schemes presented. We are also considering the problem where gene annotations are not in the binary form but are more quantitative measures.

As we discussed previously, our method could be used for protein function prediction. For a given set of proteins sharing the same GO term that are surprisingly clustered, uncharacterized proteins co-clustering with the GO term could be expected to share the GO annotation. Another exciting prospect is to use this type of local over-representation to search sequence motifs. One would seek motifs that are locally enriched in a subnetwork of the graph. Locally over-represented motifs found in protein sequences may correspond to new domains or localization signals. Those found in the 5' or 3' UTRs of genes may contain mRNA localization signals or post-transcriptional regulatory elements relevant to the subnetwork, while those found in the regulatory regions (promoters and enhancers) would allow the coordinated transcription of the proteins in the subnetwork.

## 5 Supplementary Material

The Java program used to identify GO terms enriched in subnetworks is available at: <http://www.mcb.mcgill.ca/~blanchem/GoNet>. All other supplementary files are available at the same location.

## Acknowledgements

This work was funded by a CIHR operating grant to BC and MB and an NSERC USRA scholarship to MLA. We thank Pablo Cingolani for his help on the GOA database and Ethan Kim and Ashish Sabharwal for useful suggestions.

## References

1. Al-Shahrour, F., Daz-Uriarte, R., Dopazo, J.: FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. *Bioinformatics* 20(4), 578–580 (2004)
2. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Go, G.S.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.* 25(1), 25–29 (2000)
3. Barabasi, Albert: Emergence of scaling in random networks. *Science* 286(5439), 509–512 (1999)
4. Barboric, M., Kohoutek, J., Price, J.P., Blazek, D., Price, D.H., Peterlin, B.M.: Interplay between 7SK snRNA and oppositely charged regions in HEXIM1 direct the inhibition of P-TEFb. *EMBO J.* 24(24), 4291–4303 (2005)
5. Beissbarth, T., Speed, T.P.: GOstat: find statistically overrepresented Gene Ontologies within a group of genes. *Bioinformatics* 20(9), 1464–1465 (2004)
6. Brohe, S., Faust, K., Lima-Mendez, G., Vanderstocken, G., van Helden, J.: Network Analysis Tools: from biological networks to clusters and pathways. *Nat. Protoc.* 3(10), 1616–1629 (2008)
7. Brohe, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics* 7, 488 (2006)
8. Byers, S., Price, J., Cooper, J., Li, Q., Price, D.: HEXIM2, a HEXIM1-related protein, regulates positive transcription elongation factor b through association with 7SK. *J Biol. Chem.* 280(16), 16360–16367 (2005)
9. Chua, H., Sung, W., Wong, L.: Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. *Bioinformatics* 22(13), 1623–1630 (2006)
10. Coulombe, B., Blanchette, M., Jeronimo, C.: Steps towards a repertoire of comprehensive maps of human protein interaction networks: the Human Proteotheque Initiative (HuPI). *Biochem. Cell Biol.* 86(2), 149–156 (2008)
11. Daraselia, N., Yuryev, A., Egorov, S., Mazo, I., Ispolatov, I.: Automatic extraction of gene ontology annotation and its correlation with clusters in protein networks. *BMC Bioinformatics* 8, 243 (2007)
12. Enright, A.J., Dongen, S.V., Ouzounis, C.A.: An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.* 30(7), 1575–1584 (2002)
13. Finn, R.D., Tate, J., Mistry, J., Coggill, P.C., Sammut, S.J., Hotz, H.-R., Ceric, G., Forslund, K., Eddy, S.R., Sonnhammer, E.L.L., Bateman, A.: The Pfam protein families database. *Nucleic Acids Res.* 36(Database issue), D281–D288 (2008)
14. Floyd, R.W.: Algorithm 97: Shortest path. *Communications of the ACM* 5(6), 345 (1962)
15. Hu, Z., Mellor, J., DeLisi, C.: Analyzing networks with VisANT. *Curr Protoc Bioinformatics*, Chapter 8:Unit 8.8 (December 2004)

16. Jeronimo, C., Forget, D., Bouchard, A., Li, Q., Chua, G., Poitras, C., Thrien, C., Bergeron, D., Bourassa, S., Greenblatt, J., Chabot, B., Poirier, G.G., Hughes, T.R., Blanchette, M., Price, D.H., Coulombe, B.: Systematic analysis of the protein interaction network for the human transcription machinery reveals the identity of the 7SK capping enzyme. *Mol. Cell* 27(2), 262–274 (2007)
17. Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T., Yamanishi, Y.: KEGG for linking genomes to life and the environment. *Nucleic Acids Res.* 36(Database issue), D480–D484 (2008)
18. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: *Proceedings of the ICML*, pp. 315–322 (2002)
19. Krogan, N.J., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., Li, J., Pu, S., Datta, N., Tikuisis, A.P., Punna, T., Peregrn-Alvarez, J.M., Shales, M., Zhang, X., Davey, M., Robinson, M.D., Paccanaro, A., Bray, J.E., Sheung, A., Beattie, B., Richards, D.P., Canadien, V., Lalev, A., Mena, F., Wong, P., Starostine, A., Canete, M.M., Vlasblom, J., Wu, S., Orsi, C., Collins, S.R., Chandran, S., Haw, R., Rilstone, J.J., Gandi, K., Thompson, N.J., Musso, G., Onge, P.S., Ghanny, S., Lam, M.H.Y., Butland, G., Altaf-Ul, A.M., Kanaya, S., Shilatifard, A., O'Shea, E., Weissman, J.S., Ingles, C.J., Hughes, T.R., Parkinson, J., Gerstein, M., Wodak, S.J., Emili, A., Greenblatt, J.F.: Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. *Nature* 440(7084), 637–643 (2006)
20. Li, Y., Agarwal, P., Rajagopalan, D.: A global pathway crosstalk network. *Bioinformatics* 24(12), 1442–1447 (2008)
21. Mete, M., Tang, F., Xu, X., Yuruk, N.: A structural approach for finding functional modules from large biological networks. *BMC Bioinformatics* 9(suppl. 9), S19 (2008)
22. Ohbayashi, T., Makino, Y., Tamura, T.A.: Identification of a mouse TBP-like protein (TLP) distantly related to the drosophila TBP-related factor. *Nucleic Acids Res.* 27(3), 750–755 (1999)
23. Peng, J., Zhu, Y., Milton, J., Price, D.: Identification of multiple cyclin subunits of human P-TEFb. *Genes Dev* 12(5), 755–762 (1998)
24. Przulj, N., Corneil, D.G., Jurisica, I.: Modeling interactome: scale-free or geometric? *Bioinformatics* 20(18), 3508–3515 (2004)
25. Said, M., Begley, T., Oppenheim, A., Lauffenburger, D., Samson, L.: Global network analysis of phenotypic effects: protein networks and toxicity modulation in *saccharomyces cerevisiae*. *Proc. Natl. Acad. Sci. USA* 101(52), 18006–18011 (2004)
26. Scott, J., Ideker, T., Karp, R.M., Sharan, R.: Efficient algorithms for detecting signaling pathways in protein interaction networks. *J. Comput. Biol.* 13(2), 133–144 (2006)
27. Sen, T.Z., Kloczkowski, A., Jernigan, R.L.: Functional clustering of yeast proteins from the protein-protein interaction network. *BMC Bioinformatics* 7, 355 (2006)
28. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* 13(11), 2498–2504 (2003)
29. Sharan, R., Ulitsky, I., Shamir, R.: Network-based prediction of protein function. *Mol. Syst. Biol.* 3, 88 (2007)
30. Shlomi, T., Segal, D., Ruppin, E., Sharan, R.: QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics* 7, 199 (2006)

31. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., Mesirov, J.P.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. USA* 102(43), 15545–15550 (2005)
32. Suderman, M., Hallett, M.: Tools for visually exploring biological networks. *Bioinformatics* 23(20), 2651–2659 (2007)
33. Warshall, S.: A theorem on boolean matrices. *Journal of the ACM* 9(1), 11–12 (1962)
34. Zeeberg, B.R., Feng, W., Wang, G., Wang, M.D., Fojo, A.T., Sunshine, M., Narasimhan, S., Kane, D.W., Reinhold, W.C., Lababidi, S., Bussey, K.J., Riss, J., Barrett, J.C., Weinstein, J.N.: GoMiner: a resource for biological interpretation of genomic and proteomic data. *Genome Biol.* 4(4), R28 (2003)

# Protein Fragment Swapping: A Method for Asymmetric, Selective Site-Directed Recombination

Wei Zheng<sup>1</sup>, Karl E. Griswold<sup>2</sup>, and Chris Bailey-Kellogg<sup>1</sup>

<sup>1</sup> Department of Computer Science, Dartmouth College  
6211 Sudikoff Laboratory, Hanover NH 03755, USA

wei.zheng@dartmouth.edu, cbk@cs.dartmouth.edu

<sup>2</sup> Thayer School of Engineering, Dartmouth College  
8000 Cummings Hall, Hanover, NH 03755, USA  
karl.e.griswold@dartmouth.edu

**Abstract.** This paper presents a new approach to site-directed recombination, swapping combinations of selected discontiguous fragments from a source protein in place of corresponding fragments of a target protein. By being both asymmetric (differentiating source and target) and selective (swapping discontiguous fragments), our method focuses experimental effort on a more restricted portion of sequence space, constructing hybrids that are more likely to have the properties that are the objective of the experiment. Furthermore, since the source and target need to be structurally homologous only locally (rather than overall), our method supports swapping fragments from functionally important regions of a source into a target “scaffold”; e.g., to humanize an exogenous therapeutic protein. A protein fragment swapping plan is defined by the residue position boundaries of the fragments to be swapped; it is assessed by an average potential score over the resulting hybrid library, with singleton and pairwise terms evaluating the importance and fit of the swapped residues. While we prove that it is NP-hard to choose an optimal set of fragments under such a potential score, we develop an integer programming approach, which we call SWAGMER, that works very well in practice. We demonstrate the effectiveness of our method in two types of swapping problem: selective recombination between beta-lactamases and activity swapping between glutathione transferases. We show that the selective recombination approach generates a better plan (in terms of resulting potential score) than a traditional site-directed recombination approach. We also show that in both cases the optimized experiment is significantly better than one that would result from stochastic methods.

## 1 Introduction

Protein recombination constructs libraries of hybrids by recombining fragments from two or more parents, with the goal of discovering hybrids with beneficial properties such as improved thermostability, activity, or substrate specificity [1][2][3][4][5][6][7][8][9][10][11][12][13]. For example, Stemmer demonstrated the development of beta-lactamase hybrids with a 32,000-fold increase in the required minimum inhibitory concentration of the antibiotic cefotaxime [1]. In contrast with mutagenesis techniques, recombination uses amino acid combinations that already exist in wild-type

proteins and thus are likely to produce viable proteins. Site-directed techniques seek to improve the “hit rate” of good hybrids by recombining the parents at selected break-point positions, rather than stochastically. For example, Arnold, Mayo, and co-workers showed that selecting breakpoints so as to minimize disruption of interacting amino acid pairs yields beta-lactamase hybrids that are more likely to be stably folded and functional than random ones [4].

Typically site-directed recombination is both exhaustive and symmetric: a combinatorial library of hybrids is constructed from fragments covering all residues (exhaustive) and taken uniformly from all parents (symmetric). However, in many applications it may be desirable to relax these requirements. A *selective* approach may be warranted if the parents have regions that are significantly “gappy” (insertions/deletions) in a sequence alignment or that are significantly different structurally. In such a case much experimental effort may be wasted on constructing and screening a large number of poor quality hybrids, instead of focusing on those that recombine the non-gappy and structurally analogous regions, and thus are more likely to be stably folded and functional. An *asymmetric* approach may be in order if the goal is to swap portions of particular functional importance from one protein into another. One such application is humanization of exogenous therapeutic proteins, where part of a foreign source is swapped into a human protein target that acts as a scaffold (and will not elicit an immune response). Antibodies have long been humanized this way, e.g., combining murine variable regions with human constant regions [14][15]. An approach for the much more difficult task of humanizing enzymes (which lack the overtly modular nature of antibodies) was recently demonstrated [10][11], introducing activity from a rat glutathione transferase into a human one.

In order to enable the optimization of asymmetric, selective site-directed recombination experiments, we develop here a new approach that we call *protein fragment swapping* (Fig. 1). We distinguish a *source parent* and a *target parent*, and construct a library that swaps combinations of selected discontiguous fragments *from* the source *to* the target. By swapping from source to target, our approach is asymmetric; by swapping fragments that can be discontiguous, our approach is selective. Furthermore, fragment swapping does not require the parents to be homologous (in sequence or structure) overall, but only requires there to be corresponding regions of the source and target in which we may swap fragments. Thus it directly supports the humanization application discussed in the previous paragraph. Traditional combinatorial site-directed library construction is a special case of fragment swapping, where the swapped fragments must be contiguous. By enabling the protein engineer to define sequence regions of interest, swapping focuses the experimental effort on a smaller portion of sequence space that is believed to be more relevant. Thus it improves the chance of finding beneficial new hybrids in the resulting library.

We develop an algorithm, which we call “SWAGMER” (a word created by swapping part of “FRAGMEnt” into “swAPPEr”) for planning protein fragment swapping experiments. The objective is to select, from the corresponding source and target sequence regions of interest, “good” source fragments to be combinatorially swapped in for corresponding target fragments. To assess possible plans, we employ a statistical potential score analogous to those used in combinatorial recombination to help ensure stability

of the resulting hybrids [4][6][7][8]. The potential averages over the entire resulting hybrid library a set of singleton and pairwise terms evaluating the importance of the residues and how well they match. While the averages can be computed efficiently (i.e., without enumerating the exponential number of hybrids), we show that the inclusion of pairwise terms leads to an NP-hard optimization problem. To solve the problem in practice, we develop an integer programming approach that represents swapping assignments for the residues by binary variables, and optimizes the potential score for the resulting library. To demonstrate the effectiveness of our approach, we planned experiments for selective recombination between beta-lactamases and for activity swapping between glutathione transferases. In both cases, the optimized plans outperform all randomly-generated plans (as would result from stochastic recombination methods). We further compared the selective plan with an optimized traditional site-directed recombination plan, and show that the swapping library has a better average potential score, increasing the probability of obtaining functional variants.

## 2 Methods

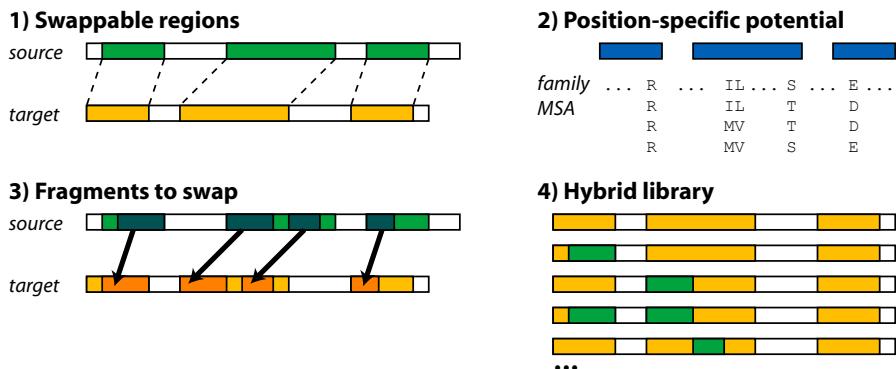
There are three main steps (Fig. II) to planning a fragment swapping experiment for a given source and target. We assume here that we are given a single source protein  $S$  of length  $m$  and target protein  $T$  of length  $n$ ; the approach can readily be generalized to multiple proteins.

1. Identify a set of *swappable regions*,  $R = \{(s_1, t_1, \ell_1), (s_2, t_2, \ell_2), \dots\}$ , where  $s_i \in [1..m]$  and  $t_i \in [1..n]$  are the starting residue positions in the source and target respectively, and  $\ell_i$  is the length of the swappable region such that  $s_i + \ell_i \leq s_{i+1}$  and  $t_i + \ell_i \leq t_{i+1}$ . Thus a swappable region defines corresponding residue substrings; regions can be separated by gaps of different lengths in the two proteins.
2. Define a *potential*  $\phi$  to evaluate a possible swapping plan. We compute the average score over the hybrids in the library, employing position-specific singleton terms  $g_i(a)$  and pairwise terms  $g_{i,j}(a, b)$  to assess the importance and fit of the swapped residue  $a$  at position  $i$  and residue pairs  $a, b$  at positions  $i, j$  with respect to amino acid statistics for related sequences.
3. Select from the swappable regions a set of *fragments* for swapping,  $F = \{(r_1, a_1, b_1), (r_2, a_2, b_2), \dots\}$ , such that for all  $i$ , we have  $r_i \in [1..|R|]$ ;  $a_i, b_i \in [1..\ell_{r_i}]$ ;  $l_{min} \leq b_i - a_i + 1 \leq l_{max}$ ; and for  $j > i$ , if  $r_i = r_j$  then  $b_i < a_j$ . The minimum and maximum fragment length constraints,  $l_{min}$  and  $l_{max}$ , control the number of residues participating in the swapping.

Our goal is to optimize the selected fragments:

**Fragment swapping problem.** Given swappable regions  $R$  and potential score  $\phi$ , find within  $R$  a set  $F$  of  $\lambda$  fragments maximizing  $\phi$ .

Modified versions of existing site-directed recombination techniques may be employed to construct the swapping library defined by a set of fragments (step 4 in Fig. II). We propose to use SPLISO [19] and RoboMix [20], hierarchically assembling hybrids by ligating fragments with short (e.g., 3-nucleotide) overhangs common to both parents, and robotically ensuring that only the desired asymmetric combinations are constructed (swapping source into target but not vice-versa).



**Fig. 1.** Overview of fragment swapping method. (1) Identify swappable regions (colored), indicating corresponding portions of the source and target proteins between which fragments may be swapped. The regions may cover most or all of the sequences, or they may be discontiguous. (2) Define a potential score to assess the library resulting from a possible swapping. The example illustrates conservation (R in the first region) and covariation (IL/MV in the second region and SE/TD across the second and third regions) within one of the families (source or target). Hybrids in a possible library can be evaluated for satisfaction of these conservation and covariation constraints. (3) Select fragments (darker colors) within the swappable regions to be swapped from the source into the target, so as to optimize the library potential. (4) Construct a hybrid library by swapping all combinations of the source fragments into the target, replacing its corresponding fragments.

## 2.1 Swappable Regions

In combinatorial site-directed protein recombination, parent proteins are typically selected from the same family [4][6][21]. It is assumed by homology that the same overall structure is common to the parents and the resulting hybrids. On a more local level, the assumption is that corresponding amino acids in an alignment of the parents are in similar local structural environments, so that the residues in the resulting hybrids will likewise be in favorable environments. The common structural context among parents and hybrids is thereby “factored out” of the planning.

In fragment swapping, we no longer require the source and target to be related or to be similar overall. However, we would still like to ensure that the hybrids maintain the overall structure of the target, and that the swapped source fragments are likely to be placed in suitable local environments in the target. This allows us to focus our optimization efforts on the specific amino acid content (the potential score in the next section). Thus we start with a set of *swappable regions*, pairs of corresponding substrings from the source and target (the first step of Fig. 1).

In cases of sufficient homology, sequence alignment suffices to determine swappable regions. We eliminate the “gappy” parts of the alignment (insertions/deletions) and use the remaining contiguous portions as swappable regions. When structures are available for both source and target, and the structures are similar enough, swappable regions can be found by standard topological structural alignment techniques [22][23][24]. We keep the portions that structurally align well and eliminate insertions/deletions and portions

with poor structural correspondence. In the most challenging cases, global structural alignment yields poor correspondence, but some local regions align well and may serve as swappable regions. Methods for establishing such local structural alignments are beyond the scope of the present work, but may be based on geometric hashing [25] or extension of aligned fragment pairs [23][24].

For the purposes of planning, we only consider the residues within the swappable regions (the inter-region residues from the target are of course included in library construction). Thus we can re-index the two protein sequences with indices from 1 to  $\ell = \sum_i \ell_i$  covering the swappable regions, where the  $\ell_i$  are the lengths of the swappable regions as previously defined. We employ this indexing in the remainder, and use brackets to get residues in  $S$  and  $T$ ; e.g.,  $S[3]$  is the third swappable-region residue in  $S$ .

## 2.2 Potential Score

Swapping can be seen as making clusters of simultaneous mutations, and our goal is to choose sets of mutations that are in some sense optimal, in that they transfer the desired function without disrupting the current scaffold. As in previous work [4][17][8][18][26][27][28][29][30], we assume that constraints on amino acid choices required to maintain structure and function are revealed in the sequence record, and devise an objective function seeking to satisfy those constraints. (In fact, related contact potentials have long been the basis for many protein structure prediction techniques [31][32].) We base the potential function here on the statistical framework from our earlier site-directed recombination work [29], but the planning method can use any potential score of the same form.

We deal here with two types of sequence constraint displayed by a multiply-aligned set of sequences: position-dependent residue conservation and covariation (see again Fig. 1, step 2). For example, if a residue is highly conserved in the source family, it may be important to swap it into the target in order to introduce the desired function. Likewise, if a pair of residues are highly correlated in the source family, it may be necessary to ensure that they are swapped as part of the same fragment, since placing them in different fragments will result in the other combinations less frequently observed in the family. While we do not include in our potential any contribution from residues outside the swappable regions (even by way of pairwise terms with a residue in the swappable regions), the potential can be generalized to do so, or an overall “environment” effect can be incorporated into the singleton terms.

More formally, let us consider conservation and covariation in a multiple sequence alignment  $\mathcal{S}$  for the source family. For the singleton terms, we define  $s_i(a)$  as the log probability of amino acid type  $a$  at residue position  $i$ :

$$s_i(a) = \log \frac{|\{P \in \mathcal{S} : P[i] = a\}|}{|\mathcal{S}|} \quad (1)$$

For the pairwise terms, we only consider residue pairs  $i$  and  $j$  that are in contact in a representative structure for the protein family (assumed common to all, by homology), as contacting pairs have the greatest direct impact on establishing a suitable local environment. We define  $s_{i,j}(a, b)$  as the log probability of the pair of amino acid types  $a$  and  $b$ , vs. what would be expected if they were independent:

$$s_{i,j}(a, b) = \log \frac{|\{P \in \mathcal{S} : P[i] = a \wedge P[j] = b\}|}{|\mathcal{S}|} - s_i(a) - s_j(b) \quad (2)$$

By subtracting the independent terms from the joint term,  $s_{i,j}$  contains only the additional information regarding the correlation between the two positions, and we can correctly compute a total score by summing up all the singleton and pairwise terms without “double-counting” the singleton contributions.

We can likewise compute  $t_i(a)$  and  $t_{i,j}(a, b)$  for the target, based on a multiple sequence alignment and representative structure. We then define the overall constraint on a position or pair of positions as a convex combination of these terms:

$$g_i(a) = \alpha \times s_i(a) + (1 - \alpha) \times t_i(a) \quad (3)$$

$$g_{i,j}(a, b) = \alpha \times s_{i,j}(a, b) + (1 - \alpha) \times t_{i,j}(a, b) \quad (4)$$

The choice of  $\alpha$  reflects whether it is more important for the hybrids to satisfy the source constraints ( $\alpha$  near 0), the target constraints ( $\alpha$  near 1), or both ( $\alpha$  in between). Note that the formula readily handles the special case where the source and target are from the same family. Other means of combining the potential are possible; however, we find this one to be both powerful and easy to interpret.

Given a hybrid with sequence  $P$ , we can evaluate how well it satisfies the conservation constraints as:

$$\phi(P) = \sum_i g_i(P[i]) + \sum_{i,j} g_{i,j}(P[i], P[j]). \quad (5)$$

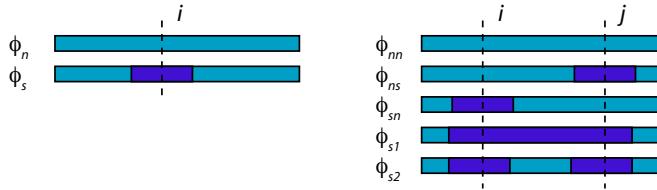
Since we would like all hybrids to satisfy the constraints, we evaluate a possible library in terms of the sum of the hybrid scores according to Eq. 5 seeking to maximize the total. More precisely, if we have selected  $\lambda$  fragments, then  $2^\lambda$  hybrids  $P_h$  ( $1 \leq h \leq 2^\lambda$ ) are created, and we seek to maximize:

$$\begin{aligned} \sum_h \phi(P_h) &= \sum_h \left( \sum_i g_i(P_h[i]) + \sum_{i,j} g_{i,j}(P_h[i], P_h[j]) \right) \\ &= \sum_i \sum_h g_i(P_h[i]) + \sum_{i,j} \sum_h g_{i,j}(P_h[i], P_h[j]). \end{aligned} \quad (6)$$

Let us define  $\phi_1(i)$  as the average over all hybrids of  $g_i(P_h[i])$ , and  $\phi_2(i, j)$  as the average of  $g_{i,j}(P_h[i], P_h[j])$ . Then we can rewrite the total potential as

$$\sum_h \phi(P_h) = 2^\lambda \times \left( \sum_i \phi_1(i) + \sum_{i,j} \phi_2(i, j) \right). \quad (7)$$

To evaluate the total potential in an efficient planning algorithm we cannot afford to enumerate the exponential number of hybrids in each possible fragment swapping. Fortunately, given the definition of a fragment swapping we can compute average potentials  $\phi_1$  and  $\phi_2$  for a given position or pair of positions in constant time, and thus the overall average potential  $\phi$  in at most quadratic time (though in practice the number of



**Fig. 2.** Swapping patterns for (left) single residue  $i$  or (right) pair of residues  $i, j$ . Fragments being swapped are shaded darker.

pairwise terms is likely to be linear, due to the contact restriction). The key insight is that each residue or residue pair participates in a well-defined pattern of hybrids, depending on the selection of fragments to be swapped. That is, the “projection” of the hybrid library onto a single column or pair of columns can be partitioned into a few cases, each with the same number of hybrids in the overall library as in Fig. 2, and we simply need to average over the cases.

For  $\phi_1(i)$  there are two possibilities, depending on whether or not residue  $i$  is swapped (Fig. 2, left).

$$\phi_n(i) = g_i(T[i]) \quad (8)$$

$$\phi_s(i) = 1/2 \times (g_i(S[i]) + g_i(T[i])) \quad (9)$$

When residue  $i$  is not being swapped ( $\phi_n$ ), all the hybrids have the target residue; when it is ( $\phi_s$ ), half the hybrids have the source residue and the other half have the target residue.

For  $\phi_2(i, j)$  there are five cases (Fig. 2, right): neither  $i$  nor  $j$  is swapped ( $\phi_{nn}$ ), only  $i$  is swapped ( $\phi_{sn}$ ), only  $j$  is swapped ( $\phi_{ns}$ ), both are swapped in the same fragment ( $\phi_{s1}$ ), or both are swapped in different fragments ( $\phi_{s2}$ ).

$$\phi_{nn}(i, j) = g_{i,j}(T[i], T[j]) \quad (10)$$

$$\phi_{sn}(i, j) = 1/2 \times (g_{i,j}(S[i], T[j]) + g_{i,j}(T[i], S[j])) \quad (11)$$

$$\phi_{ns}(i, j) = 1/2 \times (g_{i,j}(T[i], T[j]) + g_{i,j}(T[i], S[j])) \quad (12)$$

$$\phi_{s1}(i, j) = 1/2 \times (g_{i,j}(S[i], S[j]) + g_{i,j}(T[i], T[j])) \quad (13)$$

$$\phi_{s2}(i, j) = 1/4 \times (g_{i,j}(S[i], S[j]) + g_{i,j}(T[i], T[j]) + g_{i,j}(T[i], S[j]) + g_{i,j}(S[i], T[j])) \quad (14)$$

### 2.3 Fragment Selection

Recall that our goal is to select a set of fragments from the swappable regions, so that the average potential score over the resulting hybrid library is maximized. Unfortunately, we have proved that this optimization problem is NP-hard when using a potential score with pairwise terms. The detailed proof is in an appendix for the interested reader.

*Claim.* The fragment swapping problem is NP-hard.

*Proof sketch.* The proof is by reduction from MAX-2SAT. Literals in a 2-CNF formula map to residues in a swapping problem, with a correspondence between a literal being

true and a residue being in a swapping fragment. Pairwise swapping potential terms are defined so that maximizing the swapping score results in satisfying each clause and consistently treating (swapping or not) all literals using each variable.  $\square$

Computationally, the fragment swapping problem is somewhat analogous to the threading (sequence-structure alignment) problem, in which secondary structure “fragments” from a template “source” are aligned to the primary sequence for a target, according to a potential score that typically includes both singleton (environment) and pairwise (contact) terms [33][34][35][36]. (Like swapping, threading is also NP-hard [37].) The most important difference is that in threading, we know the lengths of the fragments (secondary structure elements) that must be aligned, whereas in fragment swapping, that is part of the optimization. We make use of the analogy in developing an integer programming approach to the fragment swapping problem, since RAPTOR [38] is a very successful threader based on an integer programming formulation. While drawing inspiration from that work, our formulation must employ different variables (since the fragments have unknown lengths), different constraints (to maintain a valid fragment swapping), and of course a different objective function.

In a swapping of  $\lambda$  fragments, conceptually the source and the target (those residues in swappable regions) are partitioned into a total of  $2\lambda + 1$  fragments, alternating between  $\lambda + 1$  non-swapping fragments and  $\lambda$  swapping fragments. The length of any non-swapping fragment can be 0, yielding adjacent swapping fragments or ensuring that the first or last fragment is swapping rather than non-swapping. We index the fragments from 1 to  $2\lambda + 1$ , with odd numbers for non-swapping fragments and even numbers for swapping fragments. Let  $B = \{\ell_1, \ell_1 + \ell_2, \dots, \sum_{i=1}^{|R|-1} \ell_i\}$  be the indices defining the  $|R| - 1$  internal boundaries between the regions (again, using residue indexing for swappable regions, as discussed above). We ensure that no fragment crosses an index in  $B$ .

The potential score contributions are determined by the fragments to which single residues belong and the fragment pairs to which residue pairs belong. Thus in order to develop an integer programming approach, we define singleton and pairwise binary variables,  $s_{i,f}$  and  $p_{i,j,f,g}$ , representing the assignment of residues and residue pairs to fragments.

$$s_{i,f} = \begin{cases} 1 & \text{if residue } i \text{ is in fragment } f \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

$$p_{i,j,f,g} = \begin{cases} 1 & \text{if residue } i \text{ is in fragment } f \text{ and residue } j \text{ is in fragment } g, \\ 0 & \text{otherwise,} \end{cases} \quad (16)$$

where  $1 \leq i, j \leq \ell$  and  $1 \leq f, g \leq 2\lambda + 1$ . If  $f$  is even and  $s_{i,f} = 1$ , then residue  $i$  is in the  $(f/2)$ th swapping fragment; otherwise it is in a non-swapping fragment. For efficiency, we only define  $p_{i,j,f,g}$  if there is a contact between  $i$  and  $j$  (so there is a non-zero potential score), and if  $i < j$  and  $f \leq g$  (to avoid redundancy).

With this representation, our objective function, to optimize the average potential, can be written as:

$$\Phi = \sum_i \sum_{\text{even } f} s_{i,f} \times \phi_s(i) + \sum_i \sum_{\text{odd } f} s_{i,f} \times \phi_n(i) + \sum_{i,j} \sum_{\text{odd } f,g} p_{i,j,f,g} \times \phi_{nn}(i, j)$$

$$\begin{aligned}
& + \sum_{i,j} \sum_{\text{odd } f, \text{even } g} p_{i,j,f,g} \times \phi_{ns}(i,j) + \sum_{i,j} \sum_{\text{even } f, \text{odd } g} p_{i,j,f,g} \times \phi_{sn}(i,j) \\
& + \sum_{i,j} \sum_{\text{even } f} p_{i,j,f,f} \times \phi_{s1}(i,j) + \sum_{i,j} \sum_{\text{even } f, g; f \neq g} p_{i,j,f,g} \times \phi_{s2}(i,j).
\end{aligned} \tag{17}$$

To guarantee the variable assignments yield a valid fragment swapping, we have constraints:

$$\forall i : \sum_f s_{i,f} = 1, \tag{18}$$

$$\forall i, f : s_{i,f} + \sum_{f' < f} s_{i+1,f'} \leq 1, \tag{19}$$

$$\forall \text{even } f : \sum_i s_{i,f} \geq l_{min}, \tag{20}$$

$$\forall \text{even } f : \sum_i s_{i,f} \leq l_{max}, \tag{21}$$

$$\forall i, j, f : \sum_{g \geq f} p_{i,j,f,g} = s_{i,f}, \tag{22}$$

$$\forall i, j, g : \sum_{f \leq g} p_{i,j,f,g} = s_{j,g}, \tag{23}$$

$$\forall \text{even } f \ \forall i \in B : s_{i,f} + s_{i+1,f} \leq 1. \tag{24}$$

Eq. 18 guarantees a residue can participate in only one fragment. Eq. 19 maintains the sequential order of residues and fragments. Eq. 20 and Eq. 21 enforce the minimum and maximum fragment length constraints. Eq. 22 and Eq. 23 ensure consistent single and pairwise assignments; see the claim below. Eq. 24 guarantees that no swapping fragment crosses the boundary of a swappable region.

*Claim.* For  $i < j$  and  $f \leq g$ , Eq. 18, Eq. 22 and Eq. 23 guarantee that  $p_{i,j,f,g}$  is 1 if and only if  $s_{i,f} = 1$  and  $s_{j,g} = 1$ .

*Proof.* Assume  $p_{i,j,f,g}$  has value 1. By Eq. 22, we have  $s_{i,f} \geq 1$ . Then by Eq. 18,  $s_{i,f}$  must have value 1. Similarly, by Eq. 23 and Eq. 18, we get  $s_{j,g} = 1$ .

If  $s_{i,f} = 1$  and  $s_{j,g} = 1$ , then Eq. 22 guarantees that there is a  $g' \geq f$  such that  $p_{i,j,f,g'} = 1$ . It must be the case that  $g' = g$ , because otherwise we would have  $s_{j,g'} = 0$  by Eq. 18, since  $s_{j,g} = 1$ . Then we would have  $\sum_{f' \leq g'} p_{i,j,f',g'} = s_{j,g'} = 0$  by Eq. 23, contradicting  $p_{i,j,f,g'} = 1$ .  $\square$

*Claim.* Any fragment swapping is a solution to our integer program, and any solution to our integer program defines a fragment swapping.

*Proof.* The first part is straightforward. In a fragment swapping, a residue is in only one fragment as in Eq. 18. Residue  $i$  must be in a fragment with index no larger than the one of residue  $i + 1$ , satisfying Eq. 19. The length of each swapping fragment is between  $l_{min}$  and  $l_{max}$ , satisfying Eq. 20 and Eq. 21. By the definition of Eq. 16, the value of

$p_{i,j,f,g}$  satisfies Eq. 22 and Eq. 23. Finally, a fragment does not cross swappable region boundaries, so Eq. 24 is satisfied.

Now assume we have a solution to our integer program, and let us construct a fragment swapping. To do so, we must determine the start and end of each swapping fragment, and ensure that the fragment is of the right size and remains within a swappable region. Let us consider even (swapping) fragment number  $f$  in the solution. By Eq. 20 and Eq. 21 we have  $l_{min} \leq \sum_i s_{i,f} \leq l_{max}$ , so  $f$  is of the right size. By Eq. 24, its residues do not cross a swappable region boundary. In order to obtain the start and end of  $f$ , we must ensure that its residues (i.e., the variables  $i$  with  $s_{i,f} = 1$ ) are consecutive. Assume they aren't. Then there are two residues  $i, j$ , with  $i + 1 < j$ , such that  $s_{i,f} = 1, s_{i+1,f} = 0$  and  $s_{j,f} = 1$ . Considering residue  $i$ , by Eq. 18 and Eq. 19, there is fragment  $e$ , with  $f < e$ , such that  $s_{i+1,e} = 1$ . Then there must be a residue  $k$ , with  $i + 1 \leq k < j$ , such that  $k$  is in a fragment with larger index than that of  $k + 1$ , since otherwise residue  $i + 1$  could not be in a fragment with a larger index than that of residue  $j$ . But such a  $k$  would contradict Eq. 19. Thus the residues of  $f$  must be consecutive, and we can determine the start and end of  $f$  by finding the minimum and maximum  $i$  with  $s_{i,f} = 1$ .  $\square$

Thus by maximizing the objective function, we will find the optimal selection of swapping fragments.

As mentioned in the introduction, traditional site-directed recombination between two proteins in a single family is a special case of fragment swapping. We arbitrarily call one parent protein the source and the other one the target. After aligning the sequences by standard techniques, we have a single swappable region of length  $n$  including all residues. We add the constraint  $\sum_{i,\text{even } f} s_{i,f} = n$ , and Eq. 21 is no longer needed. Then the asymmetric swapping will result in a symmetric combinatorial recombination.

### 3 Results and Discussion

To study the effectiveness of SWAGMER, we applied it to two different types of fragment swapping experiments. First we analyzed, using beta-lactamases, the difference between selective swapping and traditional site-directed recombination. Next we turned to activity swapping for enzyme humanization, using glutathione transferases, and explored planning swaps from rat source to human target.

#### 3.1 Selective Swapping of Beta-Lactamases

Beta-lactamases are enzymes produced by some bacteria; they hydrolyze the beta-lactam found in certain antibiotics (e.g., penicillin). They have been the object of much chimeragenesis work, including the pioneering site-directed studies of Arnold and colleagues [4][6]. We have also previously developed experiment planning methods for traditional site-directed recombination and applied them to beta-lactamases [29]. We use here the dataset from our previous study, consisting of 136 beta-lactamases multiply aligned to 263 residues with an average sequence identity of 41.8%, along with the representative 3D structure from *E. coli* TEM-1 beta-lactamase (pdb id 1BTL). We

derived the potential score as discussed above; we note that our previous work demonstrated that the potential is predictive of folded and functional hybrids [29]. We used as parents the proteins studied by Arnold, TEM-1 and PSE-4. Because of their highly similar structures and nearly identical disruption profiles [4], we adopted the same potential score for TEM-1 and PSE-4 in fragment selection, arbitrarily choosing TEM-1 as source. Here the objective is to explore selective site-directed recombination compared with traditional recombination covering all residues.

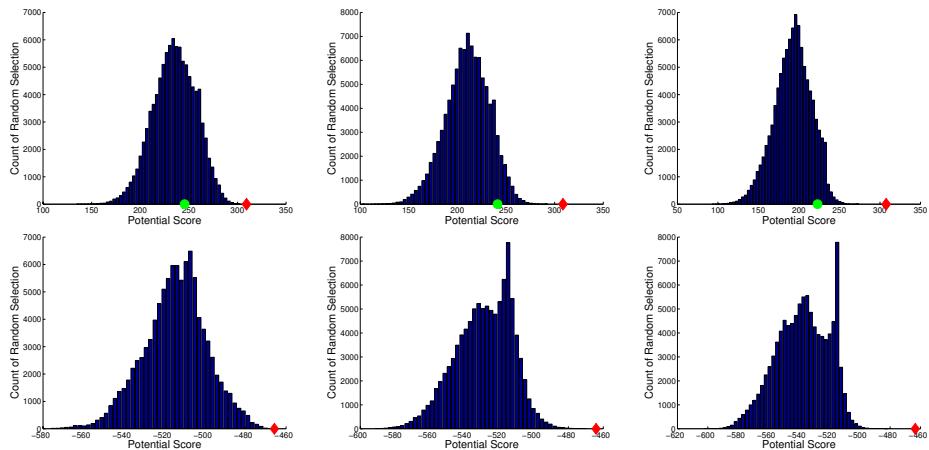
We compared the libraries optimized by SWAGMER to randomly generated plans and to plans optimized by our earlier method [29] for traditional site-directed recombination. We call the traditional approach “exhaustive” as the experiment is defined by selecting breakpoint positions at columns in a multiple sequence alignment, so that recombination necessarily covers the entire sequence rather than focusing on specific fragments. Based on the number of residues in TEM-1 and PSE-4, we set the fragment length constraints to be a minimum of 10 and a maximum of 50. We generated plans with 2, 3, or 4 fragments (yielding a manageable sized library) by SWAGMER and the random approach, and plans with the same number of hybrids by the exhaustive approach (2 swapped fragments corresponds to 1 exhaustive breakpoint, etc.). For the random approach, we generated  $10^5$  random plans, requiring more than 1 hour for 2 fragments and roughly 2 hours each for 3 and 4 fragments. We implemented SWAGMER using the CBC integer programming solver provided in COIN-OR (<https://projects.coin-or.org/Cbc>). The running times were 32 seconds (2 fragments), 776 seconds (3 fragments), and 3359 seconds (4 fragments).

The top three panels in Fig. 3 summarize the qualities of the resulting plans, in terms of average potential scores. Clearly the optimal plan is much better than would be obtained at random, as would be obtained by stochastic recombination rather than a planned approach. By focusing experimental effort on selected fragments, rather than spreading it out over the entire protein, SWAGMER also significantly outperforms the exhaustive approach. Thus the resulting library better explores this region of sequence space, giving us the opportunity to find hybrids that probably would not be generated under other methods.

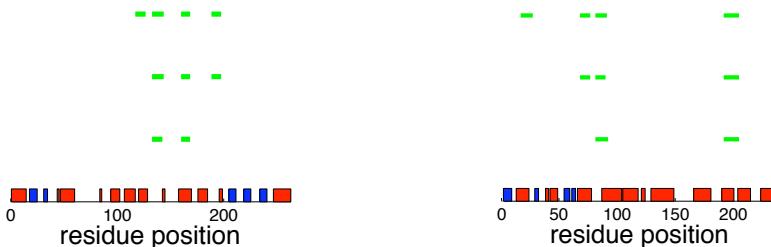
Fig. 4(left) illustrates the structure of the swapping plan. It employs minimum-length fragments, perhaps because PSE-4 and TEM-1 are distantly related [8] and thus short fragments are preferable to minimize the disruption introduced by swapping. The plans for the exhaustive approach likewise place breakpoints so as to minimize fragment length (breakpoints are stacked up at either the N- or C-terminus). The swapping fragments selected are all within protein modules identified by profile disruption [8], which it is hypothesized must be maintained in recombination to yield folded and functional hybrids.

### 3.2 Activity Swapping in Glutathione Transferases

Glutathione transferases (GSTs) are enzymes that help eliminate reactive electrophilic compounds by conjugating them to glutathione. As mentioned in the introduction, Griswold *et al.* recently demonstrated the use of chimeragenesis to swap activity from a rat GST into a human one [10]. They employed stochastic techniques to construct



**Fig. 3.** Average potential scores for generated libraries: (top) beta-lactamases, (bottom) glutathione transferases; (left) 2, (middle) 3, and (right) 4 fragments. The histogram is taken over  $10^5$  random libraries. The red diamond is the SWAGMER-optimized library. The green circle in the beta-lactamase panels is the optimal library for the “exhaustive” approach.



**Fig. 4.** Swapping plans relative to the reference structures: (left) beta-lactamases, (right) glutathione transferases. Green blocks represent optimal fragment selections for 2, 3, and 4 fragments. Red blocks represent alpha helices and blue blocks represent beta sheets.

libraries of  $\theta$ -class GSTs, recombining human GST  $\theta$ -1-1 (hGSTT1-1) and rat GST  $\theta$ -2-2 (rGSTT2-2). They identified a hybrid with 83% of the hGSTT1-1 sequence but a swapped-in rat activity. This is a powerful demonstration of the potential for activity swapping, but we show here that optimizing an experiment plan can result in a library with significantly higher average score, while focusing experimental effort on a smaller region in sequence space, thus potentially yielding a much better hit rate.

We started with sequence alignments for the two subclasses (rat and human) of  $\theta$ -class GSTs, with four sequences each, aligned to 239 residues, with an overall sequence identity of 53%. Given the small number of sequences, we followed our previous sparse data approach [29], augmenting the family statistics with database statistics, thereby introducing an amino acid-specific pseudocount. Since  $\theta$ -class GSTs have a highly conserved GST 3D fold [10] we used the hGSTT1-1 structure (pdb idb id 2C3N) as the reference structure for both subclasses. We used a weight  $\alpha = 0.5$  in Eq. 3 and 4,

placing equal importance on maintaining the human scaffold and introducing the rat activity.

The bottom three panels of Fig. 3 show the comparison between SWAGMER-optimized plans and  $10^5$  random ones, for 2, 3, and 4 fragments. As with beta-lactamases, the average potential of the optimal plans is much better than we would get from stochastic plans. The running times are 5 seconds, 44 seconds, and 1067 seconds for 2, 3, and 4 fragments, respectively. Also as with beta-lactamases, the plans seek small fragments (Fig. 4(right)). This suggests a line for further investigation: balancing maximization of the swapping potential against another metric like diversity, as we have done for traditional exhaustive site-directed recombination [39].

## 4 Conclusion

We have developed a new general framework for recombination, protein fragment swapping. By swapping only selected discontiguous regions, fragment swapping can focus on functionally important regions in parent sequences, is applicable to parents with heterogeneous structures, and is flexible in the number of residues participating in recombination. Furthermore, the asymmetric role of the source and target parents enables specific construction of libraries seeking to introduce activities from one parent into the other. Our SWAGMER method provides an efficient, effective approach to optimizing fragment swapping experiments.

**Acknowledgments.** This work was supported in part by an Alfred P. Sloan Fellowship and an NSF CAREER award to CBK (IIS-0444544).

## References

1. Stemmer, W.: Rapid evolution of a protein in vitro by DNA shuffling. *Nature* 370, 389–391 (1994)
2. Ostermeier, M., Shim, J., Benkovic, S.: A combinatorial approach to hybrid enzymes independent of DNA homology. *Nat. Biotechnol.* 17, 1205–1209 (1999)
3. Lutz, S., Ostermeier, M., Moore, G., Maranas, C., Benkovic, S.: Creating multiple-crossover DNA libraries independent of sequence identity. *PNAS* 98, 11248–11253 (2001)
4. Voigt, C., Martinez, C., Wang, Z., Mayo, S., Arnold, F.: Protein building blocks preserved by recombination. *Nat. Struct. Biol.* 9, 553–558 (2002)
5. O'Maille, P., Bakhtina, M., Tsai, M.: Structure-based combinatorial protein engineering (SCOPE). *J. Mol. Biol.* 321, 677–691 (2002)
6. Aguinaldo, A., Arnold, F.: Staggered extension process (StEP) in vitro recombination. *Methods Mol. Biol.* 231, 105–110 (2003)
7. Coco, W.: RACHITT: Gene family shuffling by random chimeragenesis on transient templates. *Methods Mol. Biol.* 231, 111–127 (2003)
8. Otey, C., Silberg, J., Voigt, C., Endelman, J., Bandara, G., Arnold, F.: Functional evolution and structural conservation in chimeric cytochromes P450: calibrating a structure-guided approach. *Chem. Biol.* 11, 309–318 (2004)
9. Castle, L., Siehl, D., Gorton, R., Patten, P., Chen, Y., Bertain, S., Cho, H.J., Duck, N., Wong, J., Liu, D., Lassner, M.: Discovery and directed evolution of a glyphosate tolerance gene. *Science* 304, 1151–1154 (2004)

10. Griswold, K., Kawarasaki, Y., Ghoneim, N., Benkovic, S., Iverson, B., Georgiou, G.: Evolution of highly active enzymes by homology-independent recombination. *PNAS* 102, 10082–10087 (2005)
11. Griswold, K., Aiyappan, N., Iverson, B., Georgiou, G.: The evolution of catalytic efficiency and substrate promiscuity in human theta class 1-1 glutathione transferase. *J. Mol. Biol.* 364, 400–410 (2006)
12. Taly, V., Urban, P., Truan, G., Pompon, D.: A combinatorial approach to substrate discrimination in the P450 CYP1A subfamily. *Biochim. Biophys. Acta* 1770, 446–457 (2006)
13. Kurtovic, S., Modén, O., Shokeer, A., Mannervik, B.: Structural determinants of glutathione transferases with azathioprine activity identified by DNA shuffling of alpha class members. *J. Mol. Biol.* 375, 1365–1379 (2008)
14. Morrison, S., Johnson, M., Herzenberg, L., Oi, V.: Chimeric human antibody molecules: Mouse antigen-binding domains with human constant region domains. *PNAS* 81, 6851–6855 (1984)
15. Jones, P., Dear, P., Foote, J., Neuberger, M., Winter, G.: Replacing the complementarity-determining regions in a human antibody with those from a mouse. *Nature* 321, 522–525 (1986)
16. Meyer, M., Silberg, J., Voigt, C., Endelman, J., Mayo, S., Wang, Z., Arnold, F.: Library analysis of SCHEMA-guided protein recombination. *Protein Sci.* 12, 1686–1693 (2003)
17. Moore, G., Maranas, C.: Identifying residue-residue clashes in protein hybrids by using a second-order mean-field approach. *PNAS* 100, 5091–5096 (2003)
18. Saraf, M., Horswill, A., Benkovic, S., Maranas, C.: Famclash: A method for ranking the activity of engineered enzymes. *PNAS* 12, 4142–4147 (2004)
19. Saftalov, L., Smith, P., Friedman, A., Bailey-Kellogg, C.: Site-directed combinatorial construction of chimaeric genes: general method for optimizing assembly of gene fragments. *Proteins* 64, 629–642 (2006)
20. Avramova, L., Desai, J., Weaver, S., Friedman, A., Bailey-Kellogg, C.: Robotic hierarchical mixing for the production of combinatorial libraries of proteins and small molecules. *J. Comb. Chem.* 10, 63–68 (2008)
21. Otey, C., Landwehr, M., Endelman, J., Hiraga, K., Bloom, J., Arnold, F.: Structure-guided recombination creates an artificial family of cytochromes P450. *PLoS Biol.* 4, e112 (2006)
22. Holm, L., Sander, C.: Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.* 233, 123–138 (1993)
23. Shindyalov, I., Bourne, P.: Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.* 11, 739–747 (1998)
24. Ye, Y., Godzik, A.: Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics* (suppl. 2), ii246–ii255 (2003)
25. Nussinov, R., Wolfson, H.: Efficient detection of three-dimensional motifs in biological macromolecules by computer vision techniques. *PNAS* 88, 10495–10499 (1992)
26. Saraf, M., Gupta, A., Maranas, C.: Design of combinatorial protein libraries of optimal size. *Proteins* 60, 769–777 (2005)
27. Russ, W., Lowery, D., Mishra, P., Yaffee, M., Ranganathan, R.: Natural-like function in artificial WW domains. *Nature* 437, 579–583 (2005)
28. Socolich, M., Lockless, S., Russ, W., Lee, H., Gardner, K., Ranganathan, R.: Evolutionary information for specifying a protein fold. *Nature* 437, 512–518 (2005)
29. Ye, X., Friedman, A., Bailey-Kellogg, C.: Hypergraph model of multi-residue interactions in proteins: sequentially-constrained partitioning algorithms for optimization of site-directed protein recombination. *J. Comput. Biol.* 14, 777–790 (2007); Conference version: Proc. RECOMB, pp. 15–29 (2006)
30. Thomas, J., Ramakrishnan, N., Bailey-Kellogg, C.: Graphical models of residue coupling in protein families. *IEEE/ACM Trans. Comput. Biol. Bioinf.* 5, 183–197 (2008)

31. Tanaka, S., Scheraga, H.: Medium and long range interaction parameters between amino acids for predicting three dimensional structures of proteins. *Macromolecules* 9, 945–950 (1976)
32. Miyazawa, S., Jernigan, R.: Estimation of effective interresidue contact energies from protein crystal structures: Quasi-chemical approximation. *Macromolecules* 18, 531–552 (1985)
33. Bowie, J., Luthy, R., Eisenberg, D.: A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253, 164–170 (1991)
34. Jones, D., Taylor, W., Thornton, J.: A new approach to protein fold recognition. *Nature* 358, 86–89 (1992)
35. Lathrop, R., Smith, T.: Global optimum protein threading with gapped alignment and empirical pair score functions. *J. Mol. Biol.* 255, 651–665 (1996)
36. Godzik, A.: Fold recognition methods. *Methods Biochem. Anal.* 44, 525–546 (2003)
37. Lathrop, R.: The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng.* 7, 1059–1068 (1994)
38. Xu, J., Li, M., Kim, D., Xu, Y.: RAPTOR: Optimal protein threading by linear programming. *J. Bioinf. Comp. Biol.* 1, 95–117 (2003)
39. Zheng, W., Friedman, A., Bailey-Kellogg, C.: Algorithms for joint optimization of stability and diversity in planning combinatorial libraries of chimeric proteins. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 300–314. Springer, Heidelberg (2008)

## A NP-Hardness of Protein Fragment Swapping

We prove the NP-hardness of the protein fragment swapping problem by reduction from MAX-2SAT. For simplicity our construction uses a single swappable region, only a pairwise potential score  $\phi_2$ , and trivial fragment length constraints  $l_{min} = 1$  and  $l_{max} = \infty$ .

Let  $C_1 \wedge C_2 \wedge \dots \wedge C_\tau$  be a boolean formula in 2-CNF with  $\tau$  clauses. Let  $N_+$  be the number of pairs of identical literals, and  $N_-$  be the number of pairs of complementary literals.

Let us first define the types of residue positions in the source and target proteins.

- **Clause:** for each clause  $C_r = (c_{r,1} \vee c_{r,2})$  with literals  $c_{r,1}$  and  $c_{r,2}$ , add two residues  $v_{r,1}$  and  $v_{r,2}$  sequentially.
- **Separator:** for each pair  $v_{r,p}, v_{r',p'} (p, p' \in \{1, 2\})$  of instances of the same literal in clauses  $r$  and  $r'$  (i.e.,  $c_{r,p}, c_{r',p'}$  are the same variable or  $c_{r,p}, c_{r',p'}$  are both the negation of the same variable), add two “separator” residues  $v_{d,1}$  and  $v_{d,2}$  sequentially between  $v_{r,2}$  and  $v_{r+1,1}$ . (Multiple pairs of separator residues may be strung in the region between clauses.)
- **Trivial:** add  $2\tau + 2N_+$  trivial residues at the end of the sequence.

The mapping between MAX-2SAT and fragment swapping is:  $v_{r,s}$  is in a swapping fragment if and only if  $c_{r,s}$  is true ( $1 \leq r \leq \tau, s \in \{1, 2\}$ ).

We need not specify the amino acid sequences for the source and targets, as the swapping problem is defined in terms of the potential. To this end, there are four types of residue pairs contributing to the potential, with  $g_{i,j}$  values in Tab. 1 yielding  $\phi_2$  values in Tab. 2 according to Eq. 9–Eq. 14.

- **Clause**, for each  $v_{r,1}, v_{r,2}$  corresponding to a clause  $C_r = (c_{r,1} \vee c_{r,2})$
- **Identical**, for each  $v_{r,p}, v_{r',p'}$  corresponding to identical literals used in clauses  $r$  and  $r'$
- **Complementary**, for each  $v_{r,p}, v_{r',p'}$  corresponding to complementary literals used in clauses  $r$  and  $r'$
- **Separator**, for each pair  $v_{d,1}, v_{d,2}$  of separator residues for the same identical literal

Fig. 5 illustrates one construction.

The construction takes polynomial time. We establish  $4\tau + 4N_+$  residues:  $2\tau$  for the clauses,  $2N_+$  separator residues, and  $2\tau + 2N_+$  trivial residues. There are  $\tau + 2N_+ + N_-$  terms in the potential:  $\tau$  for the clauses,  $N_+$  for the identical pairs with a corresponding  $N_+$  for the separator pairs, and  $N_-$  for the complementary pairs.

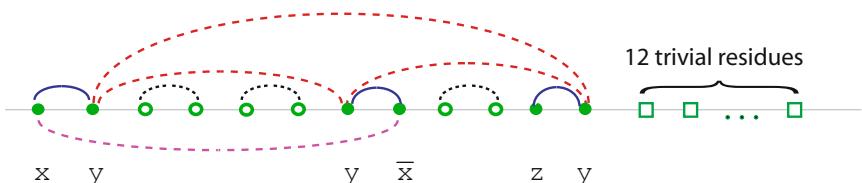
Now we prove a correspondence between the MAX-2SAT solution for  $C_1 \wedge C_2 \wedge \dots \wedge C_\tau$  and the optimal fragment swapping for the constructed  $2\tau + 2N_+$  swapping fragments. We separate the two directions of the proof.

**Table 1.** Family statistics for different types of residue pairs

	$g_{i,j}(T[i], T[j])$	$g_{i,j}(S[i], T[j])$	$g_{i,j}(T[i], S[j])$	$g_{i,j}(S[i], S[j])$
clause	0	2	2	0
identical	1	-1	-1	5
complementary	0	2	2	-4
separator	-3	3	3	-3

**Table 2.** Average potential scores for different types of residue pairs

	$\phi_{nn}$	$\phi_{ns}$	$\phi_{sn}$	$\phi_{s1}$	$\phi_{s2}$
clause	0	1	1	0	1
identical	1	0	0	3	1
complementary	0	1	1	-2	0
separator	-3	0	0	-3	0



**Fig. 5.** Residue pairs contributing to the potential  $\phi_2$  for the MAX-2SAT instance  $(x \vee y) \wedge (y \vee \neg x) \wedge (z \vee y)$ . Filled dots are residues mapping to literals in the clauses. Empty dots are separator residues. Squares represent trivial residues at the end. There are 3 clause pairs (blue solid), 3 identical pairs (red dashed above), 1 complementary pair (purple dashed below), and 3 separator pairs (black dashed).

*Claim.* If the MAX-2SAT solution satisfies  $k$  clauses, then the fragment swapping solution achieves an average potential score of  $k + N_+ + N_-$ .

*Proof.* We must show how to select fragments based on the MAX-2SAT solution. For each separator pair, create two single-residue swapping fragments. For each literal, if the literal value is true, create a single-residue fragment for the corresponding residue; otherwise the residue is not in any swapping fragment. Since there are  $2\tau + 2N_+$  non-trivial residues, after this step, there are at most  $2\tau + 2N_+$  single residue fragments. If the number of fragments is less than  $2\tau + 2N_+$ , add a sufficient number of single-residue fragments using the trivial residues at the end of the sequence.

Following Tab. 2 we have the following contributions to the potential score:

- clause: 0 for each unsatisfied clause (so that neither residue is in a swapping fragment); 1 for each satisfied clause. Note that  $\phi_{s1}$  cannot happen since we only have single-residue fragments, and each of the remaining possibilities yields 1.
- identical: 1 each. We must have either  $\phi_{nn}$  or  $\phi_{s2}$ , each of which yields 1.  $\phi_{s1}$  cannot happen due to the separator residue pairs between the two identical residues.
- complementary: 1 each. We must have either  $\phi_{ns}$  or  $\phi_{sn}$ , each of which yields 1.
- separator: 0 each

The total is  $k + N_+ + N_-$ . □

*Claim.* If the fragment swapping solution achieves an average potential score of  $k - N_+ - N_-$ , then the MAX-2SAT solution satisfies  $k$  clauses.

*Proof.* We must show how to find an assignment of literals based on the fragment swapping solution. To do this, we show that separator pairs contribute 0 to the potential, while identical and complementary pairs contribute 1 each. Thus there must be  $k - N_+ - N_-$  clause pairs contributing 1 each (the only non-zero possibility in Tab. 2). By mapping the swapped residues to literals, we can determine which literals are true and which  $k - N_+ - N_-$  clauses are satisfied.

We first prove that each separator pair contributes 0. Assume for contradiction that some separator pair contributes  $-3$  (the only other possibility in Tab. 2). Let us modify the swapping by making two single-residue fragments for the two separator residues, increasing the potential score by 3. If this increases the total number of swapping fragments above  $2\tau + 2N_+$ , then there must be some swapping fragments in the trivial residues (since there are only  $2\tau + 2N_+$  residues in the main sequence), some of which we can eliminate to leave the total at  $2\tau + 2N_+$ . The change does not affect the potential contributed by any clause pair. An identical pair can be affected if it involves the same literal as the separator pair and the two residues were previously in the same swapping fragment as the separator residue pair. In that case, the change replaces a single swapping fragment with separate swapping fragments, decreasing the potential by  $\phi_{s1} - \phi_{s2} = 2$ , which is outweighed by the increase of 3. (If multiple identical pairs are affected, then they are balanced by a corresponding number of separator pairs.) The possible analogous effect on a complementary pair can only be beneficial, yielding a net increase of  $\phi_{s2} - \phi_{s1} = 2$ . Thus we have increased the total potential, contradicting our assumption that  $k$  is the maximum.

Now let us show that identical and complementary pairs contribute 1 each. They must contribute either 0 or 1, since by the above the separator pairs contribute 0 and thus must break up any swapping fragment, eliminating the  $\phi_{s1}$  possibility in Tab. 2. If any pair contributes 0, we can modify the swapping as follows to make them all contribute 1. Let  $V$  be the set of residues for all the literals involving a particular variable (either the variable or its negation). Let  $V_+$  be the residues for the variable and  $V_-$  for its negation. Let  $V_{+s}$  and  $V_{+n}$  partition  $V_+$  into residues in swapping fragments and those not in swapping fragments, respectively; and similarly with  $V_{-s}$  and  $V_{-n}$ . By Tab. 2, each residue pair in  $V_{+n} \cup V_{-s}$  contributes 1 (for either the identical or complementary term, as appropriate), and similarly for each residue pair in  $V_{+s} \cup V_{-n}$ . The remaining residue pairs (one in  $V_{+s} \cup V_{-n}$  and one in  $V_{+n} \cup V_{-s}$ ) each contribute 0. Now let us complement the swapping assignment for each residue in  $V_{+s} \cup V_{-n}$ —if the residue is in a swapping fragment, shorten or break the fragment to make this residue not in swapping fragments; if it is not, create a single-residue swapping fragment. (As discussed above, we can modify the swapping in the trivial residues to ensure that the total number of swapping fragments is  $2\tau + 2N_+$ .) Following the above discussion, this change won't affect the potential contributed by separator pairs. It decreases the contribution from clause pairs with residues in  $V$  by at most 1 and doesn't affect other clause pairs. Pairs in  $V_{+s} \cup V_{-n}$  still contribute 1, but pairs (identical or complementary) between  $V_{+s} \cup V_{-n}$  and  $V_{+n} \cup V_{-s}$  now contribute 1 instead of 0. The total increase is  $|V_{+s} \cup V_{-n}| \times |V_{+n} \cup V_{-s}|$ , while the total decrease is at most  $|V_{+s} \cup V_{-n}|$ . We have  $|V_{+s} \cup V_{-n}| \times |V_{+n} \cup V_{-s}| \geq |V_{+s} \cup V_{-n}|$ . In this manner, we can change all identical and complementary pairs to contribute 1, which means the swapping fragment assignments of residues for these pairs are consistent.  $\square$

# Simultaneous Alignment and Folding of Protein Sequences

Jérôme Waldspühl<sup>1,3</sup>, Charles W. O'Donnell<sup>2,3</sup>, Sebastian Will<sup>4</sup>,  
Srinivas Devadas<sup>2,3</sup>, Rolf Backofen<sup>4,\*</sup>, and Bonnie Berger<sup>1,3,\*</sup>

<sup>1</sup> Department of Mathematics, MIT, Cambridge, USA

<sup>2</sup> Electrical Engineering and Computer Science, MIT, USA

<sup>3</sup> Computer Science and AI Lab, MIT, Cambridge, USA

<sup>4</sup> Institut für Informatik, Albert-Ludwigs-Universität, Freiburg, Germany

[bab@mit.edu](mailto:bab@mit.edu), [backofen@informatik.uni-freiburg.de](mailto:backofen@informatik.uni-freiburg.de)

**Abstract.** Accurate comparative analysis tools for low-homology proteins remains a difficult challenge in computational biology, especially sequence alignment and consensus folding problems. We present *partiFold-Align*, the first algorithm for simultaneous alignment and consensus folding of unaligned protein sequences; the algorithm's complexity is polynomial in time and space. Algorithmically, *partiFold-Align* exploits sparsity in the set of super-secondary structure pairings and alignment candidates to achieve an effectively cubic running time for simultaneous pairwise alignment and folding. We demonstrate the efficacy of these techniques on transmembrane  $\beta$ -barrel proteins, an important yet difficult class of proteins with few known three-dimensional structures. Testing against structurally derived sequence alignments, *partiFold-Align* significantly outperforms state-of-the-art pairwise sequence alignment tools in the most difficult low sequence homology case and improves secondary structure prediction where current approaches fail. Importantly, *partiFold-Align* requires no prior training. These general techniques are widely applicable to many more protein families. *partiFold-Align* is available at <http://partiFold.csail.mit.edu>.

## 1 Introduction

The consensus fold of two proteins is their common minimum energy structure, given a sequence alignment, and is an important consideration in structural bioinformatics analyses. In structure-function relationship studies, proteins that have the same consensus fold are likely to have the same function and be evolutionarily related [1]; in protein structure prediction studies, consensus fold predictions can guide tertiary structure predictors; and in sequence alignment algorithms [2], consensus fold predictions can improve alignments. The primary limitations in achieving accurate consensus folding, however, is the difficulty of obtaining reliable sequence alignments for divergent protein families and the inaccuracy of folding algorithms.

---

\* Corresponding authors.

The specific problem we address is predicting consensus folds of proteins from their unaligned sequences. This definition of consensus fold should not to be confused with the agreed structure between unrelated predictors [3]. Our approach succeeds by *simultaneously* aligning and folding protein sequences. By concurrently optimizing unaligned protein sequences for both sequence homology and structural conservation, both higher fidelity sequence alignment and higher fidelity structure prediction can be obtained. For sequence alignment, this sidesteps the requirement of correct initial profiles (because the best sequence aligners require profile/profile alignment [4]). For structure prediction, this harnesses powerful evolutionary corollaries between structure.

While this class of problems has received much attention in the RNA world [5,6,7,8,9,10], it has not yet been applied to proteins. Applying these techniques to proteins is more difficult and less defined. For proteins, the variety of structures is much more complicated and diverse than the standard RNA structure model, requiring our initial step of constructing an abstract template for the structure. Moreover, for proteins, there is no clear chemical basis for compensatory mutations [11], the energy models that define  $\beta$ -strand pairings are more complex, and the larger residue alphabet vastly increases the complexity of the problem.

This class of problems is also different than any that have been attempted for structure analysis. The closest related structure-prediction methods rely on sequence profiles, as opposed to consensus folds. Current protein threading methods such as Raptor [12] often construct sequence profiles of the query sequence before threading it onto solved structures in the PDB; however, given two query sequences, even if they are functionally related, it will output two structure matches but does not try to form a consensus from these. There are  $\beta$ -structure specific methods that ‘thread’ a profile onto an abstract template representing a class of structures [13,14], but do not generate consensus folds. Further, a new class of “ensemble” methods, e.g., partiFold TMB [15], “threads” a profile onto an abstract template, yet does not incorporate sequence alignment information nor generate consensus folds.

In this paper, we describe *partiFold-Align*, the first algorithm for simultaneous alignment and folding of pairs of unaligned protein sequences. Pairwise alignment is an important component in achieving reliable multiple alignments. Our strategy uses dynamic programming schemes to simultaneously enumerate the complete space of structures and sequence alignments and compute the optimal solution (as identified by a convex combination of ensemble-derived contact probabilities and sequence alignment matrices [16,17,18]). To overcome the intractability of this problem, we exploit sparsity in the set of likely amino acid pairings and aligned residues (inspired from the LocARNA algorithm [19]). partiFold-Align is thus able to achieve effectively cubic time and space in the length of its input sequences.

We demonstrate the efficacy of this approach by applying it to transmembrane  $\beta$ -barrel (TMB) proteins, one of the most difficult classes of proteins in terms of both sequence alignment and structure prediction [15,14]. In tests

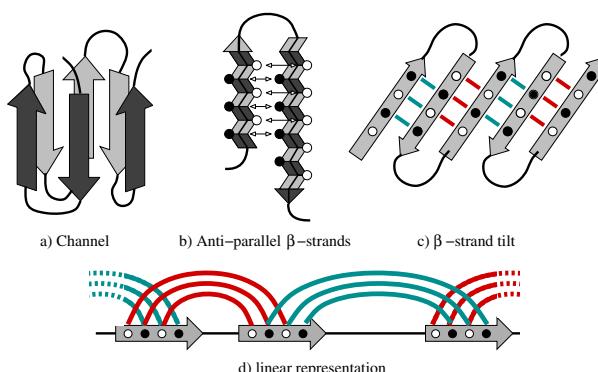
on sequence alignments derived from structure alignments, we obtain significantly better pairwise sequence alignments, especially in the case of low homology. In tests comparing single-sequence versus consensus structure predictions, *partiFold-Align* obtains improved accuracy, considerably for cases where single-sequence results are poor. The methods we develop in this paper specifically target the difficult case of alignment of low homology sequences and aim to improve the accuracy of such alignments.

**Contributions:** The main contribution of this work is that we introduce the new concept of consensus folding of unaligned protein sequences. Our algorithm *partiFold-Align* is the first to perform simultaneous folding and alignment for protein sequences. We use this to provide better sequence alignments and structure predictions for the important and difficult TMB proteins, particularly in the case of low-homology. Given the broad generality of this approach and its proven impact on the RNA world, we hope that this will become a standard in protein structure prediction.

## 2 Approach

To design an algorithm for simultaneous alignment and folding we must overcome one fundamental problem: predicting a consensus fold (structure) of two unaligned protein sequences requires a correct sequence alignment on hand, however, the quality of any sequence alignment depends upon the underlying unknown structure of the proteins. We adopt our solution to this issue from the approach introduced by Sankoff [5] to solve this problem in the context of RNAs — by predicting *partial* structural information that is then aligned through a dynamic programming procedure.

For our consensus folding algorithm, we define this partial information using probabilistic contact maps (i.e., a matrix of amino acid pairs with a high likelihood of forming hydrogen bonding partners in a protein conformation), based on Boltzmann ensemble methods, which predict the likelihood of possible

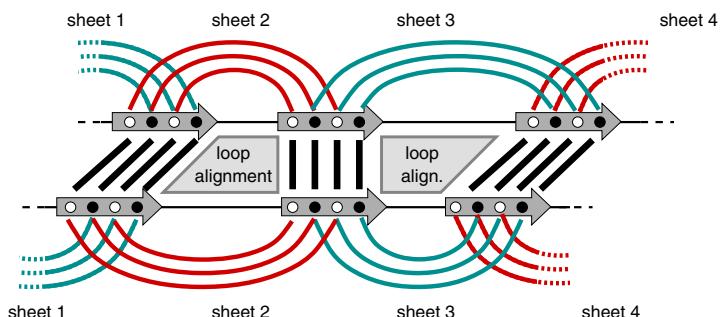


**Fig. 1.** Different structural elements of transmembrane  $\beta$ -barrels

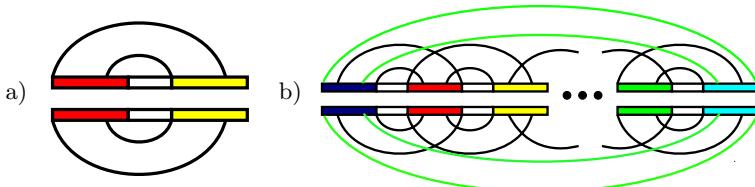
residue-residue interactions given all possible in-vivo protein conformations [14]. This is inspired by the recent *LocARNA* [19] algorithm, which improves upon Sankoff's through the use of such probabilistic contact maps. This technique is also somewhat related to the problem of *maximum contact map overlap* [20], although in such problems, contact maps implicitly signify the biochemical strength of a contact in a *solved* structured and not a well-distributed likelihood of interaction taken from a complete ensemble of possible structures.

Using such ensemble-based contact maps for simultaneous alignment and folding can be applied to other classes of proteins, however, in this work we describe our application to the class of transmembrane  $\beta$ -barrels. Unlike the RNA model used by Sankoff, TMB protein structure takes a complex form, with inclined, anti-parallel hydrogen-bonding  $\beta$ -strand forming a circular barrel structure, as depicted in Fig. 1. Partitioning such diversity of structure presents an intractable problem, so we apply a fixed parameter approach to restrict structural elements such as  $\beta$ -strand length, coil size, and the amount of strand inclination to biologically meaningful sizes.

Broadly speaking, our simultaneous alignment and folding procedure begins by predicting the ensemble-based probabilistic contact map of two unaligned sequences through an algorithm extended from partiFold TMB [14]. Importantly,  $\beta$ -strand contacts below a parameterizable threshold are excluded to allow for an efficient alignment of the most likely interactions. Alignment is then broken into two structurally different parts: the alignment of  $\beta$ -sheets, and the alignment of coils (seen in Fig. 2). Coil alignments can be performed independently at each position, however  $\beta$ -sheet alignments must respect residue pairs. Finally, to decompose the problem (Fig. 3), we first consider the optimal alignment of a single  $\beta$ -sheet with a given inclination, including the enclosed coil alignment. For energetic considerations, we must note the orientation of the  $\beta$ -strand residues (core-facing or membrane-facing), as well as whether the coil extends into the extra-cellular or periplasmic side of the membrane. Once all single alignments have been found, we “chain” these subproblems to arrive at a single consensus alignment and structure.



**Fig. 2.** Elements of TMB-alignment. Differently colored amino acids in the sheet denote exposure to the membrane and to the channel, respectively. In a valid sheet alignment, only amino acids of the same type can be matched, whereas no further constraint (except length restriction) are applied to the loop alignment.



**Fig. 3.** Problem decomposition; a) alignment of a single sheet including the enclosed loop with positive shear; b) chaining of single sheet alignment to form a  $\beta$ -barrel. Green arcs indicate the closing sheet connecting beginning and end.

## 2.1 The TMB Alignment Problem

Formally, we define an alignment  $\mathcal{A}$  of two sequences  $a, b$  as a set of pairs  $\{(p_1, p_2) \mid p_1 \in [1..|a|] \cup \{-\} \wedge p_2 \in [1..|b|] \cup \{-\}\}$  such that (i) for all  $(i, j), (i', j') \in (\mathcal{A} \cap [1..|a|] \times [1..|b|])$  we have  $i < j \implies i' < j'$  (non-crossing) and (ii) there is no  $i \in [1..|a|]$  (resp.  $j \in [1..|b|]$ ) where there are two different  $p, p'$  with  $(i, p), (i, p') \in \mathcal{A}$  (resp.  $(p, j), (p', j) \in \mathcal{A}$ ). Furthermore, for any position in both sequences, we must have an entry in  $\mathcal{A}$ . We say that  $\mathcal{A}$  is a *partial alignment* if there are some sequence positions for which there is no entry in  $\mathcal{A}$ . In this case, we denote with  $\text{def}(a, \mathcal{A})$  (resp.  $\text{def}(b, \mathcal{A})$ ) the set of positions in  $a$  (resp.  $b$ ) for which an entry in  $\mathcal{A}$  exists.

With this, the result of structure prediction is not a single structure, but a set of putative structural elements, namely the set of possible contacts for the  $\beta$ -strand. As indicated in Fig. II, we have two different side chain orientations, namely facing the channel (C) and facing the membrane (M). Since contacts can form only if both amino acids share the same orientation, a *TMB probabilistic contact map*  $P$  of any TMB  $a$  is a matrix  $P = (P(i, i', x))_{1 \leq i \leq i' \leq |a|, x \in \{C, M\}}$  where  $P(i, i', x) = P(i', i, x)$  and  $\forall x \in \{C, M\} : \sum_i P(i, i', x) \leq 1$ . To overcome the intractability of this problem, we exploit sparsity in the set of likely amino acid pairings. Thus, we use only those entries in the matrix  $P$  which have a likelihood above a parameterizable threshold.

We weight the alignments with a scoring function that sums a folding energy term  $\mathcal{E}()$  with an alignment score  $\mathcal{W}()$ , where the energy term  $\mathcal{E}()$  corresponds to the sum of the folding energies of the consensus structure mapped onto the two sequences. To allow a convex optimization of this function, we introduce a parameter  $\alpha$  distributing the weights of the two terms. Thus, given two sequences  $a, b$ , an alignment  $\mathcal{A}$  and a consensus TMB structure  $\mathcal{S}$  of length  $|\mathcal{A}|$ , the score of the alignment is:

$$\text{score}(\mathcal{A}, \mathcal{S}, a, b) = (1 - \alpha) \cdot \mathcal{E}(\mathcal{A}, \mathcal{S}, a, b) + \alpha \cdot \mathcal{W}(\mathcal{A}, a, b)$$

Let  $E_{ct}(x, y)$  be the energy value of a pairwise residue contact. Since by definition of a consensus structure these contacts are aligned, we define the energy component of the score() as:

$$\mathcal{E}(\mathcal{A}, \mathcal{S}, a, b) = \sum_{\substack{(i) \in \mathcal{A}, (i') \in \mathcal{A} \\ (i, i') \in \mathcal{S}_a^{\text{arcs}}, (j, j') \in \mathcal{S}_b^{\text{arcs}}}} \tau(i, i', j, j'), \text{ where } \tau(i, i', j, j') = E_{ct}(i, i') + E_{ct}(j, j')$$

In practice, *partiFold-Align* implements a slightly more complex stacking pair energy model as described in [15]. However, for pedagogical clarity, we use here only pairwise residue contact potentials.

Now, let  $\sigma(x, y)$  be the substitution score of the amino acids  $x$  by  $y$ , and  $g(x)$  an insertion/deletion cost. Then, the sequence alignment component of the score() is given by:

$$\mathcal{W}(\mathcal{A}, a, b) = \sum_{(i) \in \mathcal{A}} \sigma(a_i, a_j) + \sum_{(-) \in \mathcal{A}} g(a_i) + \sum_{(-) \in \mathcal{A}} g(a_j)$$

Again, in practice, a penalty for opening gaps is added but not described here for clarity. Finally, the optimization problem our algorithm solves is, given two sequences  $a$  and  $b$ :

$$\arg \max_{\substack{\mathcal{A} \text{ TMB alignment of } a \text{ and } b, \\ \mathcal{S} \text{ TMB structure of length } |\mathcal{A}|}} \{\text{score}(\mathcal{A}, \mathcal{S}, a, b)\}.$$

To account for the side-chain orientation of residues in TM  $\beta$ -strands toward the channel or the membrane, the  $\mathcal{E}()$  and  $\mathcal{W}()$  recursion equations require a slightly more detailed version of the scoring. An additional condition is that contacts only happen between amino acids with the same orientation, and that this orientation alternates between consecutive contacts. Hence, we introduce in  $\tau$  an additional parameter *env* standing for this side-chain orientation environment feature. The same holds for the edit scores  $\sigma$  and  $g$ , where the orientation can also be the loop environment. For the strands, we use  $\sigma_s(i, j, env)$ , while for loops we distinguish inner from outer loops (indicated by the loop type *lt*) with the amino acids in the loops scored using  $\sigma_l(i, j, lt)$ . The gap function is treated analogously.

## 2.2 Decomposition

We now define the dynamic programming tables used for the decomposition of our problem. The alignment of a single anti-parallel strand pair as shown in Fig. 3a has nested arcs and an outdegree of at most one. We introduce for this configuration a table *ShA()* (where *ShA* stands for *sheet alignment*) aligning pairs of subsequences  $a_{i..i'}$  and  $b_{j..j'}$ . Another parameter to account for is the shear number which represents the inclination of the strands in the TM  $\beta$ -barrel. Since the strand pair alignments also include a loop alignment, and the scoring function of this loop depends on the loop type (inner/outer loop), we need to set the loop type as an additional parameter. Similarly, we need to know the orientation of the final contact to ensure the succession of channel and membrane orientations. Given an orientation environment of a contact *env*, the term *next<sub>c</sub>(env)* return the orientation of the following contact. Thus, we have a table *ShA*( $i, i'; j, j'; env; lt; s$ ) with the following

recursion:

$$\text{ShA}(i, i'; j, j'; \text{env}, \text{lt}; s) = \max \begin{cases} \text{ShAgap}(i, i'; j, j'; \text{env}, \text{lt}; s) \\ \text{ShAshear}(i, i'; j, j'; \text{env}, \text{lt}; s) & \text{if } s \neq 0 \\ \text{ShAcontact}(i, i'; j, j'; \text{env}, \text{lt}) & \text{if } s = 0 \\ \text{LA}(i, i'; j, j'; \text{lt}) & \text{if } s = 0 \end{cases}$$

where

$$\begin{aligned} \text{ShAcontact}(i, i'; j, j'; \text{env}, \text{lt}) = & \text{ShA}(i+1, i'-1; j+1, j'-1; \text{next}_c(\text{env}); \text{lt}; 0) \\ & + \tau(i, i'; j, j'; \text{env}) + \sigma_s(a_i, b_j, \text{env}) + \sigma_s(a_{i'}, b_{j'}, \text{env}) \end{aligned}$$

$$\begin{aligned} \text{ShAgap}(i, i'; j, j'; \text{env}, \text{lt}; s) = & \text{ShAshear}(i, i'; j, j'; \text{env}, \text{lt}; s) = \\ \max \begin{cases} \text{ShA}(i+1, i'; j, j'; \text{env}, \text{lt}; s) + g_s(a_i, \text{env}) \\ \text{ShA}(i, i'-1; j, j'; \text{env}, \text{lt}; s) + g_s(a_{i'}, \text{env}) \\ \text{ShA}(i, i'; j+1, j'; \text{env}, \text{lt}; s) + g_s(b_j, \text{env}) \\ \text{ShA}(i, i'; j, j'-1; \text{env}, \text{lt}; s) + g_s(b_{j'}, \text{env}) \end{cases} & \max \begin{cases} \text{ShA}(i+1, i'; j+1, j'; \text{env}, \text{lt}; s+1) \\ + \sigma_s(a_i, b_j, \text{env}) & \text{if } s < 0 \\ \text{ShA}(i, i'-1; j, j'-1; \text{env}, \text{lt}; s-1) \\ + \sigma_s(a_{i'}, b_{j'}, \text{env}) & \text{if } s > 0 \end{cases} \end{aligned}$$

*ShAgap*, *ShAcontact* and *ShAshear* are introduced for better readability and will not be tabulated. The matrix  $\text{LA}(i, i'; j, j'; \text{lt})$  represents an alignment of two loops  $a_{i..i'}$  and  $b_{j..j'}$ , with a loop type *lt*. This table can be calculated using the usual sequence alignment recursion. Thus, we have

$$\text{LA}(i, i'; j, j'; \text{lt}) = \begin{cases} \text{LA}(i, i'-1; j, j'; \text{lt}) + g_l(a_{i'}, \text{lt}) \\ \text{LA}(i, i'; j, j'-1; \text{lt}) + g_l(b_{j'}, \text{lt}) \\ \text{LA}(i, i'-1; j, j'-1; \text{lt}) + \sigma_l(a_{i'}, b_{j'}, \text{lt}) \end{cases}$$

As we have already mentioned in the definition of a contact map, we use a probability threshold to reduce both space and time complexity of the alignment problem, in a similar way as is done in the *LocARNA*-approach [19]. Thus, we will tabulate only values in the ShA-matrix for those positions  $i, i'$  and  $j, j'$  where the contact probability is above a threshold in both sequences. This is handled at the granularity of strand pairs in practice to reduce complexity.

### 2.3 Chaining

The next problem is to chain the different single sheet alignments, as indicated by Fig. 3b. To build a valid overall alignment, we have to guarantee that the sub-alignments agree on overlapping regions. A *strand alignment*  $\mathcal{A}_s$  is just a partial alignment. The solution is to extend the matrices for sheet alignments by an additional entry for the alignment of strand regions. Albeit there are exponentially many alignments in general, there are several restrictions on the set of allowed alignments since they are alignments of strand regions. In the case of TMB-barrels, we assume no strand bulges since they are a rare event. Hence, one can insert or delete only a complete contact instead of a single amino acid. When chaining sheet alignments, the gap in one strand is then transferred to the chained sheet (by the agreement of sub-alignments).

The first step is to extend the matrices of sheet alignments by an alignment descriptor which is used to ensure the compatibility of sub-solutions used in the recursion. Note that although the alignment is fixed for the strands of a sheet, the scoring is not since we could still differentiate between a match of two bases or a match of a contact. Thus, the new matrix is  $\text{ShA}(i, i'; j, j'; \text{env}; \text{lt}; s; \mathcal{A}_s)$ , where we enforce  $\mathcal{A}_s$  to satisfy  $\text{def}(a, \mathcal{A}_s) = [i..l_1] \cup [r_1..i']$  and  $\text{def}(b, \mathcal{A}_s) = [j..l_2] \cup [r_2..j']$  for some  $i < l_1 < r_1 < i'$  and  $j < l_2 < r_1 < j'$ . The new version of  $\text{ShA}()$  is

$$\text{ShA}(i, i'; j, j'; \text{env}, \text{lt}, s; \mathcal{A}_s) = \max \begin{cases} \text{ShAgap}(i, i'; j, j'; \text{env}, \text{lt}, s; \mathcal{A}_s) \\ \text{ShAshear}(i, i'; j, j'; \text{env}, \text{lt}, s; \mathcal{A}_s) & \text{if } s \neq 0 \\ \text{ShAcontact}(i, i'; j, j'; \text{env}, \text{lt}; \mathcal{A}_s) & \text{if } s = 0 \\ \text{LA}(i, i'; j, j'; \text{lt}) & \text{if } s = 0 \end{cases}$$

$\text{LA}(i, i'; j, j'; \text{lt})$  does receive an additional parameter since sub-alignment agreement in chaining is restricted to strands. For definitions  $\text{ShAgap}()$ ,  $\text{ShAcontact}()$  and  $\text{ShAshear}()$ , we now must check whether the associated alignment operations are compatible with  $\mathcal{A}_s$ . Thus, the new definition of  $\text{ShAcontact}()$  is

$$\text{ShAcontact}(i, i'; j, j'; \text{env}, \text{lt}; \mathcal{A}_s) = \max \begin{cases} \text{ShA}(i + 1, i' - 1; j + 1, j' - 1; \text{env}, \text{lt}, 0; \mathcal{A}_s) & \text{if } (i, j) \in \mathcal{A}_s \\ + \tau(i, i'; j, j'; \text{env}) + \sigma_s(a_{i'}, b_{j'}, \text{env}) & \text{and } (i', j') \in \mathcal{A}_s \\ -\infty & \text{else} \end{cases}$$

If all entries are incompatible with  $\mathcal{A}_s$ , then  $-\infty$  is returned. Note that we add an amino acid match score only for a single specified end of the contact. Thus,  $\sigma_s(a_i, b_j)$  is skipped. The reason is simply that otherwise this score would be added twice in the course of chaining. The new definition of  $\text{ShAshear}$  is then

$$\text{ShAshear}(i, i'; j, j'; \text{env}, \text{lt}, s; \mathcal{A}_s) = \max \begin{cases} \text{ShA}(i + 1, i'; j + 1, j'; \text{env}, \text{lt}, s + 1; \mathcal{A}_s) & \text{if } s < 0 \wedge (i, j) \in \mathcal{A}_s \\ \text{ShA}(i, i' - 1; j, j' - 1; \text{env}, \text{lt}, s - 1; \mathcal{A}_s) & \text{if } s > 0 \wedge (i', j') \in \mathcal{A}_s \\ + \sigma_s(a_{i'}, b_{j'}, \text{env}) & \text{else} \end{cases}$$

The new variant of  $\text{ShAgap}()$  is defined analogously. Now we can define the matrix  $Dchain()$  for chaining the strand pair alignments. At the end of its construction, the sheet is closed by pairing its first and last strands to create the barrel. To construct this, we need to keep track of the leftmost and rightmost strand alignments  $\mathcal{A}_s^{\text{chain}}$  and  $\mathcal{A}_s^{\text{cyc}}$  of the sheet. We add two parameters, first, a variable  $ct$  used to determine if the closing strand pair has been added or not. Here,  $ct = c$  means that the sheet is not closed while  $ct = l_f$  indicates that the barrel has been built. Second, to control the number of strand in the barrel, we add the variable  $nos$  storing the number of strands in the sheet.

We initialize the array  $Dchain$  for every  $i, j$  and any strand alignment  $\mathcal{A}_s^{\text{cyc}}$  such that  $\text{def}(a, \mathcal{A}_s^{\text{cyc}}) = [i..i']$  and  $\text{def}(b, \mathcal{A}_s^{\text{cyc}}) = [j..j']$ . This initializes the array to a non-barrel solution. Then

$$Dchain(i, j; \mathcal{A}_s^{\text{cyc}}, \mathcal{A}_s^{\text{cyc}}, c; \text{lt}, 1) = \text{LA}(i, |a|; j, |b|; \text{lt}, 1),$$

where  $lt$  represents the orientation environment. Note that the strand alignment has not yet been scored. We now describe the chain rules used to build a sheet (an unclosed barrel). To account for the alignment of the first strand of this sheet (so far unscored in ShA) we introduce a function  $ShA_{start}(\mathcal{A}, nos)$  returning the cost of this alignment when  $nos = 2$ , and returning 0 otherwise. A function  $prev()$  returning the previous loop type is also used to alternate loop environments between both sides of the membrane. In addition, given two alignments  $\mathcal{A}_s, \mathcal{A}'_s$ , we say that  $\mathcal{A}_s, \mathcal{A}'_s$  agree on the strands  $i..i'$  in the first sequence and  $j..j'$  in the second sequence, written  $agr(\mathcal{A}'_s; \mathcal{A}_s; i, i'; j, j')$ . With this notation, the recursion used to build the unclosed sheets is:

$$Dchain(i, j; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; c; lt; nos) = \max_{\substack{i', j', \mathcal{A}'_s, s, lt', env \\ \text{with} \\ \text{ShA}(i, i'; j, j'; lt'; s; \mathcal{A}'_s) > -\infty, \\ \text{def}(a, \mathcal{A}_s) = [i..l_1] \cup [r_1..i'], \\ \text{def}(b, \mathcal{A}_s) = [j..l_2] \cup [r_2..j'], \\ \text{and } agr(\mathcal{A}'_s; \mathcal{A}_s; i, l; j, l')}} \left( \begin{array}{l} \text{ShA}(i', i; j, j'; env; lt'; s; \mathcal{A}'_s) \\ + Dchain(r_1, r_2; \mathcal{A}'_s; \mathcal{A}_s^{\text{cyc}}; c; prev(lt); nos - 1) \\ + ShA_{start}(\mathcal{A}'_s, nos) \end{array} \right).$$

We conclude this section by defining the recursions used to close the barrel and perform a sequence alignment of the N-terminal sequences. Since the anti-parallel or parallel nature of the closing strand pair depends of the number of strands in the barrel, we introduce here a function  $ShAclose()$  which returns the folding energy of the parallel strand pairings of the leftmost and rightmost strands of the sheet if the number of strands  $nos$  is odd, and folding energy of the anti-parallel strand pairings if  $nos$  is even.

$$Dchain(i, j; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; l_f; lt) = \max \left\{ \begin{array}{l} Dchain(i + 1, j; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; l_f; lt) + g_l(a_i, lt) \\ Dchain(i, j + 1; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; l_f; lt) + g_l(b_j, lt) \\ Dchain(i + 1, j + 1; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; l_f; lt) + \sigma_l(a_i, b_j, lt) \\ \max_{i', j', env, nos} \left\{ \begin{array}{l} Dchain(i, i'; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; c; lt) \\ + ShAclose(i, i'; j, j'; env; s; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; dir(nos)) \end{array} \right\} \end{array} \right\}$$

The final value of the consensus folding problem is then found in the function  $Dchain(1, 1; \mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; l_f; lt)$  for some  $lt$  and  $\mathcal{A}_s, \mathcal{A}_s^{\text{cyc}}$  with  $agr(\mathcal{A}_s; \mathcal{A}_s^{\text{cyc}}; 1, i; 1, j)$ , where  $\text{def}(a, \mathcal{A}_s) = [1..i] \cup [r..i']$  and  $\text{def}(b, \mathcal{A}_s) = [1..j] \cup [r..j']$ . Solutions are built using classical backtracking procedures.

These final  $Dchain()$  equations assume that the strand inclinations, modeled using the shear number  $s$ , are independent. However, in practice this parameter must be used to determine when a strand pair can be concatenated at the end of an existing sheet to ensure the coherency of the barrel structure and conserve a constant inclination of the strands (see Fig. II).

### 3 Results

Here we demonstrate the benefits of the *partiFold-Align* algorithm when applied to the problems of pairwise sequence alignment and structure prediction of transmembrane  $\beta$ -barrel proteins. Our sequence alignment performance greatly improves upon comparable alignment techniques, and surpasses state-of-the-art alignment tools (which use additional algorithmic filters) in the case of low homology sequences. It is also shown that a *partiFold-Align* consensus fold can better predict secondary structure when aligning proteins within the same superfamily. We begin with a description of our test dataset and scoring metrics as well as the *partiFold-Align* parameters chosen for the analysis, followed by our specific sequence alignment and structure prediction results.

#### 3.1 Dataset and Evaluation Technique

By implementing our algorithmic framework to align and fold transmembrane  $\beta$ -barrels, we highlight how this approach can significantly improve the alignment accuracy of protein classes with which current alignment tools have difficulty. Specifically, few TMB structures have been solved through X-ray crystallography or NMR (less-than 20 non-homologous to-date), and often known TMB sequences exhibit very low sequence homology (e.g. less-than 20%), despite sharing structure and function.

To judge how well *partiFold-Align* aligns proteins in this difficult class, we select 13 proteins from five superfamilies of TMBs found in the Orientation of Proteins in Membranes (OPM) database [21] (using the OPM database definition of class, superfamily, and family). This constitutes all solved TMB proteins with a single, transmembrane,  $\beta$ -barrel domain, and excludes proteins with significant extracellular or periplasmic structure and limits the sequence length to a computationally-tractable maximum of approximately 300 residues. With the assumption that structural alignment best mimics the intended goal of identifying evolutionary and functional similarities, we perform structural alignments between all pairs of proteins within large superfamilies, and across smaller superfamilies (28 alignments, see supplementary material for an illustration of the breakdown), and for testing purposes consider this the “correct” pairwise alignment. For structural alignments, the *Matt* [22] algorithm is used, which has demonstrated state-of-the-art structural alignment accuracy. During analysis, the resulting alignments are then sorted by relative sequence identity<sup>1</sup> (assuming the *Matt* alignment) [23,24].

Our *partiFold-Align* alignments are then compared against structural alignments using the  $Q_{Cline}$  [25,26] scoring metric, restricted to transmembrane regions as defined by the OPM (since structural predictions in the algorithm only contribute to transmembrane  $\beta$ -strand alignments; coils are effectively aligned on sequence-alone).  $Q_{Cline}$  can be considered a percentage accuracy, and resembles the simplistic  $Q_{combined}$  score<sup>2</sup>, measuring combined under- and over-prediction of aligned

<sup>1</sup> Sequence Identity % =  $\frac{\text{Identical positions}}{\text{aligned positions} + \text{internal gap positions}}$

<sup>2</sup>  $Q_{combined} = \frac{\# \text{ correct pairs}}{\# \text{ unique pairs in sequence \& structure alignments}}$

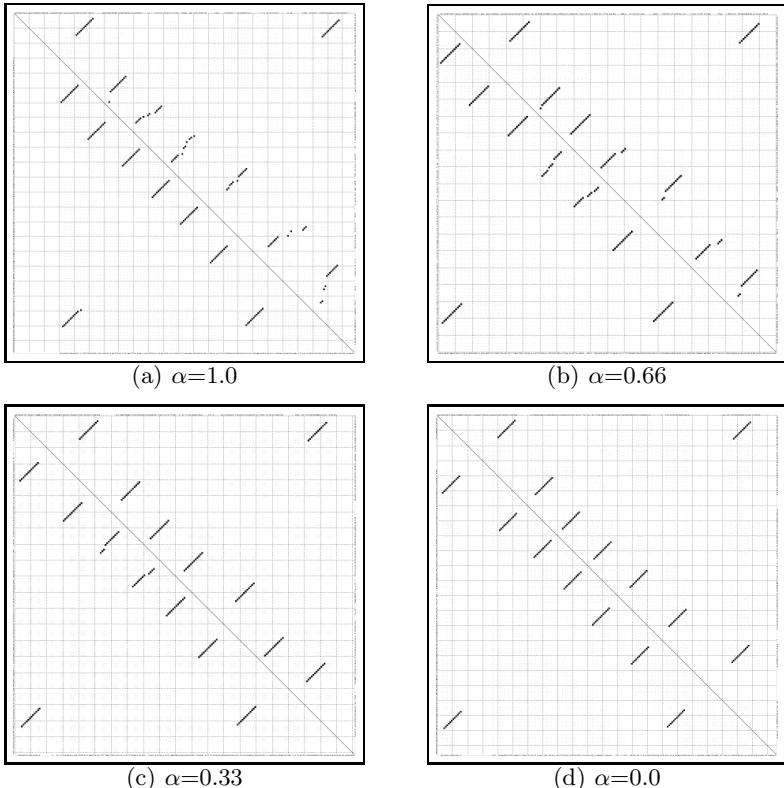
pairs, but more fairly accounts for off-by- $n$  alignments. Such shifts often occur from energetically-favorable off-by- $n$   $\beta$ -strand pairings that remain useful alignments. The  $Q_{Cline}$  parameter  $\epsilon$  is chosen to be 0.2, which allows alignments displaced by up to five residues to contribute (proportionally) toward the total accuracy. The higher the  $Q_{Cline}$  score, the more closely the alignments match (ranging  $[-\epsilon, 1]$ ).

To judge the accuracy of a *partiFold-Align* consensus structure against a structure predicted from single-sequence alone, we test against the same OPM database proteins described above. For all 13 proteins, a structure prediction is computed using the exact same ensemble structure prediction methodology as in *partiFold-Align*, only applied to a single sequence. The transmembrane-region  $Q_2$  secondary structure prediction score between the predicted structures and the solved PDB structure (annotated by STRIDE [27]) can then be computed; where  $Q_2 = (TP + TN)/(sequence\ length)$ .

### 3.2 Model Parameter Selection

For our analyses, parameters must be chosen for the abstract structural template defined in Section 2. In transmembrane  $\beta$ -barrels, the choice of allowable (minimum and maximum)  $\beta$ -strand and coil region lengths, as well as shear numbers can be assigned based on biological quantities such as membrane thickness, etc. (Even in the absence of all other information, the sequence length alone of a putative transmembrane  $\beta$ -barrel can suggest acceptable ranges.) Other algorithmic parameters, such as the pairwise contact threshold (which filters which  $\beta$ -strand pairs are used in the alignment), the Boltzmann  $Z$  constant (found within  $E_{ct}()$  in  $\mathcal{E}()$ , effecting the structural energy score [15]), the gap penalty, the choice of substitution matrix, and the  $\alpha$  balance parameter require selection without as clear a biological interpretation.

For results presented in this paper, one of three sets of structural parameters were chosen according to protein superfamily, with a fairly wide range of values permitted. To determine the algorithmic parameters listed above in a principled manner, we chose a single set of algorithmic parameters for all alignments, with the exception of varying the  $\beta$ -strand pair probability threshold used in the initial step of the algorithm, and the  $\alpha$  score-balancing parameter. In all cases, our choices are made oblivious to the known structures in our testing sets. The substitution matrix we use is a combination of the BATMAS [16] matrix for transmembrane regions, and BLOSUM [17] for coils. For alignments with a sequence homology below 10%, we chose a higher probability threshold value ( $1 \times 10^{-5}$  versus  $1 \times 10^{-10}$ ) to restrict alignments to highly-likely  $\beta$ -strand pairs, reducing signal degradation from low-likelihood  $\beta$ -strand pairs with very distant sequence similarities. For these same alignments (below 10%), we chose a lower  $\alpha$  parameter (0.6 versus 0.7) to boost the contribution of the structural prediction to the overall solution when less sequence homology could be exploited. As seen in Fig. 4, consensus predictions from lower  $\alpha$  parameters more closely resemble predictions based solely on structural scores, and thus, an optimal alignment should correlate  $\alpha$  with sequence homology.

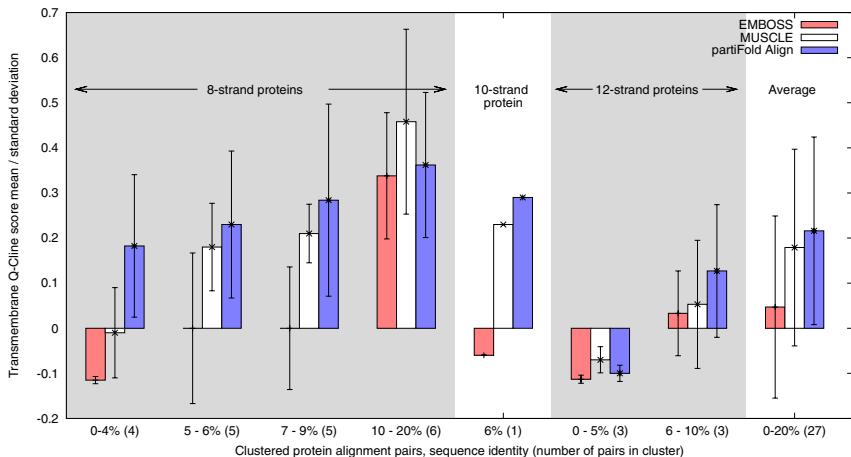


**Fig. 4.** Stochastic contact maps from a *partiFold-Align* run on the proteins 1BXW and 2F1V. For each of the four plots, the sequence of 1BXW and 2F1V is given on the axes (with gaps), and high probability residue-residue interactions indicated for 1BXW on the lower left half of the graph and 2F1V on the upper right half (i.e., the single-sequence probabilistic contact maps). Structural contact map alignment can be judged by how well the plot is mirrored across the diagonal. Subfigure (a) ( $\alpha = 1.0$ ) shows an alignment which ignores the contribution of the structural contact map, while (d) ( $\alpha = 0.0$ ) shows an alignment wholly-dependent on the structural contact map, and ignorant of sequence alignment information.

Admittedly, this naive, single (or few) parameter solution does not enable the full potential of our algorithm. A protein-specific machine learning approach would allow for a better parameter fit, and is the focus of ongoing research.

### 3.3 Alignment Accuracy of Low Sequence Identity TMBs

To compare the accuracy of alignments generated by *partiFold-Align* against current sequence alignment algorithms, we perform the same TMB pairwise sequence alignments using *partiFold-Align*, EMBOSS (Needleman-Wunsch) [18], and MUSCLE [28][29]. EMBOSS may be considered the best Needleman-Wunsch style global sequence alignment algorithm (a straight-forward, widely applicable method of



**Fig. 5.** Mean and standard deviation  $Q_{Cline}$  scores for 8-, 10-, and 12-stranded TMBs. Each of the 3 categories of proteins are clustered and ordered according to sequence identity, with the number of alignments in each cluster in parentheses. Note: By definition,  $Q_{Cline}$  scores range between  $-\epsilon$  and 1.0, where  $\epsilon = 0.2$ ; negative values indicate very poor alignments.

alignment), while MUSCLE is widely thought the most accurate of the “fast” alignment programs. Though it incorporates several position-specific gap penalty heuristics similar to those found in MAFFT and LAGAN [30] <sup>3</sup> Since the *partiFold-Align* algorithm utilizes Needleman-Wunsch style dynamic programming, comparisons between EMBOSS and *partiFold-Align* represent a fair analysis of what simultaneous folding and alignment algorithms specifically contribute to the problem. Comparisons with MUSCLE alignment scores necessitate inclusion to portray the practical benefits *partiFold-Align* provides. However, no technical reason prevents MUSCLE’s gap penalty heuristics to be incorporated with *partiFold-Align*; this stands as future work.

Fig. 5 presents transmembrane  $Q_{Cline}$  accuracy scores for EMBOSS, MUSCLE, and *partiFold-Align* across 27 TMB pairwise alignments. (The absent 28<sup>th</sup> alignment, between 1BXW and 2JMM (50% sequence-homologous), is aligned with a nearly-perfect  $Q_{Cline}$  score of 0.98 by all three algorithms). Results are separated into the 3 categories according to the number of circling strands within a protein’s  $\beta$ -barrel: seven 8-stranded OMPA-like proteins account for 21 alignments, two 10-stranded OMPT-like proteins account for one alignment, and finally, four 12-stranded Autotransporters, OM phospholipases,<sup>3</sup> and Nucleoside-specific porins make up the final six alignments (a full summary can be found in supplementary material). Equal-sized clusters of pairwise alignments are then formed and ordered according to sequence identity, with cluster mean  $Q_{Cline}$  and standard deviation

<sup>3</sup> We note, that while EMBOSS uses only the BLOSUM substitution matrix, and *partiFold-Align* a combination of BATMAS and BLOSUM, Forrest et al. [4] show that BATMAS-style matrices do not show improvement for EMBOSS-style algorithms.

reported. All individual alignment-pair statistics, as well as alternative accuracy metrics (e.g.  $Q_{combined}$ ) can be found in supplementary material.

Across all TMBs, *partiFold-Align* alignments are more accurate than EMBOSS alignments by an average  $Q_{Cline}$  of 16.9% (4.5x). Most importantly, *partiFold-Align* significantly improves upon the EMBOSS  $Q_{Cline}$  score for all alignments with a sequence identity lower than 9% (by a  $Q_{Cline}$  average of 28%), and roughly matches or improves 24/28 alignments overall. Excluding the 12-strand alignments, which align proteins across different superfamilies, our intra-superfamily alignments exhibit even higher improvements in average  $Q_{Cline}$ , besting EMBOSS by 20.3% (27.4% versus 7.1%). Even compared with MUSCLE alignments, *partiFold-Align* is able to achieve a 4% increased  $Q_{Cline}$  on average, despite its lack of gap penalty heuristics employed by MUSCLE.

### 3.4 Secondary Structure Prediction Accuracy of Consensus Folds

Here we investigate how the consensus structure resulting from our simultaneous alignment and folding algorithm can improve structure prediction accuracy over a prediction computed from a single sequence alone. We report in Tab. II  $Q_2$  accuracies computed from alignments of all pairs of TMB sequences within the same  $n$ -stranded category. For each protein, the  $Q_2$  score from the single sequence minimum folding energy (m.f.e.) structure is given (as done in [14]), and compared against the  $Q_2$  score from the best alignment partner, and the average  $Q_2$  score obtained when aligning that protein with all others in its category.

**Table 1.** Secondary structure assignment accuracy. Percentage  $Q_2$  of secondary structure prediction correctly assigned residues (transmembrane and non-transmembrane regions). Third column reports the performance of a single strand folding (no alignments). Fourth and fifth columns report respectively the best and the average  $Q_2$  scores of a consensus structure over all possible alignment pairs for this PDB ID.

Category	PDB id	single seq.	consensus	
			best	average
8-stranded	1BXW	72	70(-2)	63(-9)
	1P4T	60	68(+8)	58(-2)
	1QJ8	65	68(+3)	66(+1)
	2F1V	47	63(+22)	62(+15)
	1THQ	50	69(+13)	52(+2)
	2ERV	57	67(+10)	59(+2)
10-stranded	2JMM	62	65(+3)	62(+0)
	1K24	60	69(+9)	69(+9)
12-stranded	1I78	76	83(+7)	83(+7)
	1QD6	54	61(+7)	56(+2)
	1TLY	59	59(+0)	58(-1)
	1UYN	56	56(+0)	53(-3)
	2QOM	51	55(+4)	53(+2)

The results for 8- and 10-stranded categories show a clear improvement (more than 8%) by the best consensus fold in 6/9 instances (1P4T, 2F1V, 1THQ, 2ERV, 1K24, 1I78), and roughly equivalent results for the remaining 3 (2F1V, 1K24, 1I78). Further, on average, nearly all proteins show equivalent or improved scores when aligned with any other protein, with the exception of 1BXW. However, the single sequence structure prediction  $Q_2$  for 1BXW is not only high, but significantly higher than all other 8-stranded proteins; the contact maps of any other aligning partner may simply add noise, diluting accuracy. Conversely, the proteins which have poor single sequence structure predictions benefit the greatest from alignment (e.g. 2F1V). This relationship is certainly not unidirectional, though, as we see that the consensus fold of 1K24 and 1I78 improves upon both proteins' single sequence structure prediction.

In contrast, the results compiled on the 12-strands category do not show any clear change in the secondary structure accuracy. However, recalling that this category covers 3 distinct superfamilies in the OPM database, such results may make sense. The Autotransporter, OM phospholipase, and Nucleoside-specific porin families all exhibit reasonably different structures, and perform quite unrelated tasks. Further, unlike the original partiFold TMB algorithm [15], the abstract structural template used in this work does not take into account  $\beta$ -strands that extend far beyond the cell membrane (since our alignments focus on membrane regions). This may also effect the structure prediction accuracy of more complex TMBs.

We conclude from this benchmark that the consensus folding approach can be used to improve the structure prediction of low homology sequences, provided both belong to the same superfamily. However, we emphasize the importance parameter selection may play in these results; a different parameter selection method may enable accuracy improvement for higher-level classes of proteins.

## 4 Conclusions

We have presented *partiFold-Align*, a new approach to the analysis of proteins, which simultaneously aligns and folds pairs of unaligned protein sequences into a consensus to achieve both improved sequence alignment and structure prediction accuracy. To demonstrate the efficacy of this approach, we designed and tested the algorithm for the difficult class of transmembrane  $\beta$ -barrel, low sequence homology proteins. However, we believe this technique to be generally applicable to many classes of proteins where the structure can be defined through a chaining procedure as described in Section 2 (e.g., most  $\beta$ -sheet structures). This could open new areas of analysis that were previously unattainable given current tools' poor ability to construct functional alignments on low sequence homology proteins.

While we have shown that consensus folds can significantly improve upon pairwise sequence alignment, we believe this approach can also translate into considerable improvements in multiple sequence alignments. This is because many multiple alignment procedures use pairwise alignment information at their core [25]. Such an extension would be an obvious next step for our approach to be added in combination with other, more elaborate techniques found in sequence alignment algorithms (e.g., MUSCLE).

Similarly, we believe that the effectiveness of *partiFold-Align* can be enhanced significantly by a well-formulated machine learning approach to parameter optimization as has been applied to the case of RNA [631]. Supporting this notion, we experimented with parameters selected based on a known test set, and saw pairwise sequence alignment accuracies with an average  $Q_2$  accuracy  $\sim 20\%$  greater than MUSCLE (versus the reported  $\sim 4\%$  improvement for test-set blind parameter selections). However, for the case of TMBs, one notable problem that would need to be overcome is the relatively small set of known structure or alignments with which to use for training. Supplementary materials, including more detailed results, can be found at <http://partiFold.csail.mit.edu>.

## References

- Shakhnovich, B.E., Deeds, E., Delisi, C., Shakhnovich, E.: Protein structure and evolutionary history determine sequence space topology. *Genome Res.* 15(3), 385–392 (2005)
- Edgar, R.C., Batzoglou, S.: Multiple sequence alignment. *Curr. Opin. Struct. Biol.* 16(3), 368–373 (2006)
- Selbig, J., Mevissen, T., Lengauer, T.: Decision tree-based formation of consensus protein secondary structure prediction. *Bioinform.* 15(12), 1039–1046 (1999)
- Forrest, L.R., Tang, C.L., Honig, B.: On the accuracy of homology modeling and sequence alignment methods applied to membrane proteins. *Biophys J.* 91(2), 508–517 (2006)
- Sankoff, D.: Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Comput.* 45(5), 810–825 (1985)
- Do, C.B., Foo, C.S., Batzoglou, S.: A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics* 24, i68–i76 (2008)
- Hofacker, I.L., Bernhart, S.H.F., Stadler, P.F.: Alignment of RNA base pairing probability matrices. *Bioinformatics* 20(14), 2222–2227 (2004)
- Mathews, D.H., Turner, D.H.: Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.* 317(2), 191–203 (2002)
- Havgaard, J.H., Torarinsson, E., Gorodkin, J.: Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput. Biol.* 3(10), 1896–1908 (2007)
- Backofen, R., Will, S.: Local sequence-structure motifs in RNA. *J. Bioinform. Comput. Biol.* 2(4), 681–698 (2004)
- Fariselli, P., Olmea, O., Valencia, A., Casadio, R.: Progress in predicting inter-residue contacts of proteins with neural networks and correlated mutations. *Proteins* (suppl. 5), 157–162 (2001)
- Xu, J., Li, M., Kim, D., Xu, Y.: RAPTOR: Optimal protein threading by linear programming. *J. of Bioinform. and Comp. Biol.*, JBCB (2003)
- Bradley, P., Cowen, L., Menke, M., King, J., Berger, B.: Betawrap: Successful prediction of parallel beta-helices from primary sequence reveals an association with many microbial pathogens. *Proceedings of the National Academy of Sciences* 98(26), 14819–14824 (2001)
- Waldspühl, J., Berger, B., Clote, P., Steyaert, J.M.: Predicting transmembrane beta-barrels and interstrand residue interactions from sequence. *Proteins* 65(1), 61–74 (2006)

15. Waldspuhl, J., O'Donnell, C.W., Devadas, S., Clote, P., Berger, B.: Modeling ensembles of transmembrane beta-barrel proteins. *Proteins* 71(3), 1097–1112 (2008)
16. Sutormin, R.A., Rakhmaninova, A.B., Gelfand, M.S.: Batmas30: amino acid substitution matrix for alignment of bacterial transporters. *Proteins* 51, 85–95 (2003)
17. Henikoff, S., Henikoff, J.: Amino acid substitution matrices from protein blocks. *PNAS* 89, 10915–10919 (1992)
18. Rice, P., Longden, I., Bleasby, A.: Emboss: the european molecular biology open software suite. *Trends Genet.* 16(6), 276–277 (2000)
19. Will, S., Reiche, K., Hofacker, I.L., Stadler, P.F., Backofen, R.: Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput. Biol.* 3(4), e65 (2007)
20. Caprara, A., Carr, R., Istrail, 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)
21. Lomize, M., Lomize, A., Pogozheva, I., Mosberg, H.: OPM: Orientations of Proteins in Membranes database. *Bioinformatics* 22, 623–625 (2006)
22. Menke, M., Berger, B., Cowen, L.: Matt: local flexibility aids protein multiple structure alignment. *PLoS Comp. Bio.* 4(1), e10 (2008)
23. Doolittle, R.: Similar amino acid sequences: chance or common ancestry? *Science* 214, 149–159 (1981)
24. Raghava, G., Barton, G.: Quantification of the variation in percentage identity for protein sequence alignments. *BMC Bioinformatics* 7, 415 (2006)
25. Dunbrack, R.L.J.: Sequence comparison and protein structure prediction. *Curr. Opin. Struct. Biol.* 16(3), 374–384 (2006)
26. Cline, M., Hughey, R., Karplus, K.: Predicting reliable regions in protein sequence alignments. *Bioinformatics* 18(2), 306–314 (2002)
27. Frishman, D., Argos, P.: Knowledge-based protein secondary structure assignment. *Proteins* 23, 566–579 (1995)
28. Edgar, R.C.: Muscle: multiple sequence alignment with high accuracy and high throughput. *NAR* 32(5), 1792–1797 (2004)
29. Edgar, R.C.: Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* 5, 113 (2004)
30. Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., Green, E.D., Sidow, A., Batzoglou, S.: Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna. *Genome Res.* 13(4), 721–731 (2003)
31. Do, C.B., Woods, D.A., Batzoglou, S.: CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics* 22(14), e90–e98 (2006)

# Shared Peptides in Mass Spectrometry Based Protein Quantification

Banu Dost<sup>1</sup>, Nuno Bandeira<sup>1</sup>, Xiangqian Li<sup>2</sup>, Zhouxin Shen<sup>2</sup>,  
Steve Briggs<sup>2</sup>, and Vineet Bafna<sup>1</sup>

<sup>1</sup> Department of Computer Science and Engineering,  
University of California, San Diego, CA 92093, USA  
`{bdost,nbandeira,vbafna}@cs.ucsd.edu`

<sup>2</sup> Department of Biological Sciences,  
University of California, San Diego, CA 92093, USA  
`{xli,z1shen,sbriggs}@ucsd.edu`

**Abstract.** In analyzing the proteome using mass spectrometry, the mass values help identify the molecules, and the intensities help quantify them, relative to their abundance in other samples. Peptides that are shared across different protein sequences are typically discarded as being uninformative w.r.t each of the parent proteins.

In this paper, we investigate the use of shared peptides which are ubiquitous ( $\sim 50\%$  of peptides) in mass spectrometric data-sets. In many cases, shared peptides can help compute the relative amounts of different proteins that share the same peptide. Also, proteins with no unique peptide in the sample can still be analyzed for relative abundance. Our paper is the first attempt to use shared peptides in protein quantification, and makes use of combinatorial optimization to reduce the error in relative abundance measurements. We describe the topological and numerical properties required for robust estimates, and use them to improve our estimates for ill-conditioned systems. Extensive simulations validate our approach even in the presence of experimental error. We apply our method to a model of *Arabidopsis* root knot nematode infection, and elucidate the differential role of many protein family members in mediating host response to the pathogen.

**Keywords:** shared peptides, protein quantification, linear programming, optimization, iTRAQ, mass spectrometry.

## 1 Introduction

The analysis of the proteome using mass spectrometry involves the separation of molecules (often, enzymatically digested peptides from expressed proteins) followed by accurate measurement of mass of each molecule, termed as the mass-spectrum. Together with mass, the spectrum also measures peak-intensity for each molecule. For any constituent peptide from a protein sequence, its spectral intensity is a measurement of *abundance*, the amount of the expressed protein. However, the actual value is hard to interpret, as it depends upon a number of

poorly understood factors, including instrument types, energetics of the process, and physico-chemical properties of the peptide itself. Consequently, it is often the *relative-abundance* of a peptide, measured as the ratio of intensities of a peptide across samples, that is investigated [48]. By the same token, intensity values of different peptides are usually not comparable.

The relative abundance of a peptide is a proxy for the relative abundance of the parent protein. This is acceptable only when the peptide sequence is unique to the protein. By contrast, when a peptide is shared across proteins (Ex: proteins that share domains), its abundance (and relative abundance) depends upon contributions from multiple proteins. For this reason, shared peptides have been traditionally disregarded in protein-level quantification analysis. However, this may significantly decrease the number of proteins for which abundance estimates can be obtained. While often unreported, a significant portion of the data (as much as 50%) is ignored. In our own experiment with *Arabidopsis* proteins, 4,145(48%) of the 8,584 expressed proteins were not represented by a unique peptide and would normally be discarded.

The goal of this paper is to demonstrate that shared peptides are a resource that adds value, and we make the point with two simple examples. Consider a case with two proteins  $p_1, p_2$ , and 3 constituent peptides  $s_1, s_2, s_3$ , where  $s_1, s_2$  are unique, and  $s_3$  is shared. See Figure Ia, where a peptide is connected to a protein by an edge only if it is contained in it. Consider an experiment that revealed the relative abundances  $(r_1, r_2, r_3)$  of the 3 peptides over two samples  $B$  and  $A$  as 16, 1, 4 respectively. The typical approach is to discard the shared peptide  $s_3$ , and to assert that  $p_1$  is  $16\times$  over-expressed, while  $p_2$  is unchanged. Formally, if  $Q_j^A, Q_j^B$  represent the actual abundance of protein  $j$  in samples  $A, B$  respectively, then

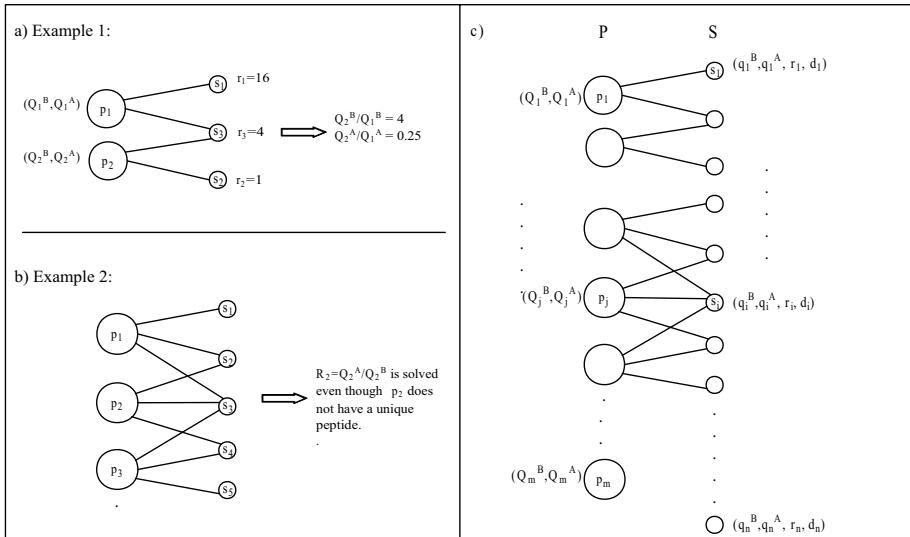
$$\frac{Q_1^A}{Q_1^B} = r_1 = 16 \text{ and } \frac{Q_2^A}{Q_2^B} = r_2 = 1.$$

Our point is that the ignored peptide  $s_3$  also provides information because

$$r_3 = 4 \simeq \frac{Q_2^A + Q_1^A}{Q_2^B + Q_1^B} = \frac{Q_2^B + 16Q_1^B}{Q_2^B + Q_1^B} = \frac{16 + \frac{Q_2^B}{Q_1^B}}{1 + \frac{Q_2^B}{Q_1^B}}.$$

Solving, we learn that  $\frac{Q_2^B}{Q_1^B} = 4$ , indicating that  $p_2$  is  $4\times$  more abundant than  $p_1$  in sample  $B$ . Here, we have 4 unknowns from the 2 proteins, and 3 constraints, one from each of the peptide. By using ratios, (Ex:  $R_j = Q_j^A/Q_j^B$ ), we reduce the number of unknowns, and can solve to get the extra information. Note that the unit of measurement for  $Q_j^A, Q_j^B$  is immaterial. For this reason, we always reduce one degree of freedom, typically by adding the constraint  $\sum_j Q_j^B = 100$ , or solving for ratios, as we do here. As long as the number of constraints matches the number of unknowns, we can solve to get the relative abundances of different proteins, possibly only with shared peptides.

Consider a second example, a more complex one this time, with 3 proteins  $p_1-p_3$  and 5 peptides  $s_1-s_5$ , as shown in Figure Ib. Here, protein  $p_2$  does not have



**Fig. 1.** (a, b) Two examples illustrating our approach for protein quantification via shared peptides. (c) Protein-peptide bi-partite graph  $G = (P \cup S, E)$  representing the mapping between  $m$  proteins and  $n$  peptides.

any unique peptide, and would normally be discarded. However, the system has 5+1 constraints, and 6 unknowns. Therefore, solving the system gives us  $Q_2^A$ ,  $Q_2^B$ , and therefore, the relative abundance  $\frac{Q_2^A}{Q_2^B}$  of  $p_2$ .

To summarize, shared peptides provide extra information in protein quantification. Under certain conditions, they allow us to a) compute relative abundance of a protein even when it does not contain a unique peptide, and b) compute relative abundance values of two different proteins in a sample. To our knowledge, this is the first paper to exploit shared peptides in this manner. However, the simple idea is confounded by the realities of missing data, and error in experiments. Here, we lay out the theoretical foundations and practical considerations in determining when the shared peptide abundances can be used reliably. We show that the solvability must depend upon the topological properties of the peptide-protein relationships as well as numerical properties of the experimentally determined intensity values. It is often the case that interesting cases cannot be resolved because of missing data, or numerical instability.

As an extension to our approach, we also consider some intrinsic properties of peptides. Informally, define the *detectability* of a peptide as the probability that it will be detected via MS, when the parent protein is expressed. We propose an alternative formulation that estimates the peptide detectabilities in addition to absolute and relative abundances of proteins when appropriate data is available.

Furthermore, we suggest two improvements to increase the number of cases that can be solved. First, we describe a algebraic technique based on singular value decomposition to make robust inferences for numerically ill-conditioned systems.

Recent results have shown that detectability is indeed an intrinsic characteristic of peptides that can be computed in independent experiments, and maintained for future use [1]. We also point out that incorporating detectabilities as known variables in our formulation, it is possible to solve a much larger number of cases.

In Section 2, we describe the theoretical and empirical considerations for shared peptide analysis. In Section 3.1, we validate our approach with extensive simulations. We apply our methods to data from iTRAQ experiments comparing an *Arabidopsis* model of root-knot infection versus wild-type in Section 3.2. Our results elucidate the relative abundance among different members of a family in over 55 *Arabidopsis* protein families.

## 2 Protein Quantification via Shared Peptides

We represent the protein quantification data using a bipartite graph  $G = (P \cup S, E)$  where  $P$  is the set of proteins and  $S$  is the set of peptides. For all  $p \in P, s \in S$ ,  $(p, s) \in E$  if and only if peptide  $s$  is a substring of the protein sequence  $p$ . Note that different connected components of  $G$  do not influence each other, and we treat each component independently. W.l.o.g, assume that  $G$  is connected, and let  $|P| = m$  and  $|S| = n$ . See Figure 1c. Consider the case where only two samples are involved. In many experiments, the abundances are measured before and after a treatment, so we denote the samples as  $B$ , and  $A$ . We associate two variables  $(Q_j^B, Q_j^A)$  corresponding to the ‘before’ and ‘after’ abundance for each protein  $p_j \in P$ . As mentioned earlier, we also add the constraint  $\sum_j Q_j^B = 100$ .

Analogous to proteins, we associate values  $q_i^B, q_i^A, r_i$  with each peptide  $s_i \in S, i = 1 : n$  where  $r_i = q_i^A/q_i^B$  denotes the ratio of the peptide  $s_i$  abundance between samples. It is possible to generalize the representation for the data with more than two samples. While this abstraction hides many of the complexities of protein quantification via mass spectrometry, it is useful to present our approach which can be applied to many different quantification protocols, including labeled and label-free approaches.

Key to our computation are equations that connect all proteins  $p_j$  which contain a single peptide  $s_i$ . In the absence of experimental error, the abundance values must satisfy the following  $n + 1$  constraints over  $2m$  variables.

$$\begin{aligned} \sum_{(p_j, s_i) \in E} Q_j^A - r_i \times \sum_{(p_j, s_i) \in E} Q_j^B &= 0 \text{ for all } s_i \in S \\ \sum_j Q_j^B &= 100 \end{aligned}$$

With no errors, we can solve this equation uniquely as long as  $n + 1 \geq 2m$ . To incorporate errors, we consider a *linear-programming* formulation that minimizes the total error. (See Figure 2, F<sub>1</sub> formulation.)

Note that the ratios are not symmetric about 1, so we always choose a constraint where the ratio contribution is greater than 1. To simplify notation, we will also represent the LP formulation in a matrix form as

$$\min \sum_i |\varepsilon_i| \text{ where } \varepsilon = Ax - b, x \geq 0 \quad (1)$$

	Input	Output	Formulation
F <sub>1</sub>	$r_i, \forall s_i \in S$	$Q_j^B, Q_j^A, \forall p_j \in P$	$\min \sum_{i=1}^n  \varepsilon_i $ $\text{s.t. } \sum_{p_j \in P} Q_j^B = 100$ $\varepsilon_i = \sum_{(p_j, s_i) \in E} Q_j^A - r_i \times \sum_{(p_j, s_i) \in E} Q_j^B \quad \forall s_i \in S, r_i \geq 1$ $\varepsilon_i = r_i \times \sum_{(p_j, s_i) \in E} Q_j^A - \sum_{(p_j, s_i) \in E} Q_j^B \quad \forall s_i \in S, r_i \leq 0$ $Q_j^B \geq 0, Q_j^A \geq 0 \quad \forall p_j \in P.$
F <sub>2</sub>	$q_i^B, q_i^A, r_i, \forall s_i \in S$	$Q_j^B, Q_j^A, \forall p_j \in P$ $d_i, \forall s_i \in S$	$\min \sum_{i=1}^n  \varepsilon_i^B  +  \varepsilon_i^A $ $\text{s.t. } \sum_{p_j \in P} Q_j^B = 100$ $\varepsilon_i^B = \sum_{(p_j, s_i) \in E} Q_j^B - q_i^B f_i, \forall s_i \in S$ $\varepsilon_i^A = \sum_{(p_j, s_i) \in E} Q_j^A - q_i^A f_i, \forall s_i \in S$ $Q_j^B \geq 0, Q_j^A \geq 0, \quad \forall p_j \in P.$

**Fig. 2.** Input, output, and computation summary of two LP formulations for protein quantification via shared peptides. a) F<sub>1</sub>: A formulation that does not include peptide detectability, and; b) F<sub>2</sub>: using peptide detectabilities. We use  $f_i = 1/d_i$  as the reciprocal of detectability to maintain linear constraints.

where  $x$  is vector of dimension  $2m$ , given by  $x = [Q_1^B, \dots, Q_m^B, Q_1^A, \dots, Q_m^A]^T$ ,  $b$  is a  $(n+1)$ -dimensional vector described by  $b = [100, 0, \dots, 0]^T$ , and  $\mathcal{A}$  is a  $(n+1) \times 2m$  matrix. While this LP is not in standard form, it can easily be transformed into one.

The formulation of the linear program is natural in that the LP seeks for protein abundances that optimally fit the observed peptide ratios. Nevertheless, it raises questions about our confidence in the estimates of  $Q_j$ . Note first that a low value for the objective does not necessarily result in robust estimates of  $Q_j$ . Consider an under-determined system, where  $n+1 < 2m$ . By setting an arbitrary subset of  $2m - (n+1)$  variables to 0, and solving for the remaining, we obtain multiple solutions, each with 0 error. A simple illustration of this is found in the notion of *symmetric proteins*. Define proteins  $p_1 \in P$  and  $p_2 \in P$  as symmetric if and only if the set of incident peptides  $S_1 = \{s | (p_1, s) \in E\}$  and  $S_2 = \{s | (p_2, s) \in E\}$  are identical. Two symmetric proteins imply two identical columns in  $\mathcal{A}$ , which means that any linear combination of abundances for these 2 proteins will lead to an identical solution. Certainly, we can solve this problem as a special case: simply merge the two identical columns (proteins) into one, effectively reducing  $m$ . However, more complex dependencies might arise which are harder to detect.

Generalizing, when  $\text{rank}(\mathcal{A}) < 2m$ , we get multiple solutions with zero-error. If however,  $\mathcal{A}$  has full column rank, then by parsimony arguments, the LP solution is likely to provide accurate estimates of protein abundance values. Even if the system is full-rank, it might be ill-conditioned, resulting in poor estimates. We define a *rank-threshold* function to characterize the solvability, a quantity that is closely related to the condition number of the matrix. Start with the singular-value decomposition of  $\mathcal{A}$ . Using standard approaches, compute matrices  $U, V, \Sigma$  such that

$$\mathcal{A} = V\Sigma U^T$$

$\Sigma$  is a  $(n+1) \times 2m$  diagonal matrix with nonnegative real numbers  $\sigma_1, \dots, \sigma_p$  on the diagonal. These  $p$  diagonal entries describe the singular values of  $\mathcal{A}$  in decreasing order of magnitude, where  $p \leq \min\{n+1, 2m\}$ .  $U$  is orthonormal with dimensionality  $2m \times 2m$ , and  $V$  is orthonormal with dimensionality  $n+1 \times n+1$ , respectively. The rank of  $\mathcal{A}$  is given by the number of non-zero singular values. We define a related concept, *rank-threshold* of  $\mathcal{A}$  as

$$\mathcal{R}(\mathcal{A}) = \min\{t | \sigma_j > 10^{-t} \forall j\}$$

$\mathcal{R}(\mathcal{A}) = t$  being low implies that all its singular values are large ( $\geq 10^{-t}$ ), implying that estimates of protein abundance values should be robust. In our experiments, we will show that the rank-threshold is a good way to characterize the reliability of the final solution.

*Robust estimates for ill-conditioned systems:* This formalism allows us to distinguish high rank systems  $\mathcal{A}$  for which we can estimate protein abundance reliably, but it also provides a handle into under-determined systems. Using our notation, we can describe an under-determined system as one in which  $\mathcal{R}(\mathcal{A})$  is high. Specifically,  $\mathcal{R}(\mathcal{A}) = \infty$  implies the case when some of the singular values are 0. For a rank threshold  $t$ , define the  $\text{rank}_t$  of  $\mathcal{A}$  as

$$\text{rank}_t(\mathcal{A}) = \max\{j | \sigma_j > 10^{-t}\}$$

This ‘thresholded’ rank allows us to get the true dimensionality of a system for which we could get robust results. For all  $j$ , let  $U_j$  denote the  $2m \times j$  matrix formed by taking the first  $j$  columns of  $U$  (corresponding to the dominant singular values). Likewise, let  $V_j$  denote the matrix formed by the first  $j$  columns of  $V$ , and  $\Sigma_j = \text{diag}[\sigma_1, \dots, \sigma_j]$ . This implies that if  $\text{rank}_t(\mathcal{A}) = k$ , then

$$\mathcal{R}(\mathcal{A}U_k) = \mathcal{R}(V_k\Sigma_k) = t$$

We choose  $B = \mathcal{A}U_k$ , and solve the linear program for the  $k$  dimensional vector  $y$

$$\min \sum_i |\varepsilon_i| \text{ where } \varepsilon = \mathcal{B}y - b, U_k y \geq 0 \quad (2)$$

Note that  $\mathcal{R}(B) = t$  implying that the estimates of  $y$  are robust. The reason to keep  $y$  unconstrained, but impose  $U_k y \geq 0$  is the following: The values  $y$  cannot

be interpreted directly, but can be used to retrieve protein abundance values  $x$  by solving

$$x = U_k y$$

Our constraints ensure that the protein abundance values are non-negative.

## 2.1 Incorporating Peptide Detectability

Here we consider an alternative formulation that builds on different assumptions to improve robustness to measurement errors and potentially greatly increase the numbers of components that can be solved. Assuming one is able to estimate the *absolute* peptide abundances  $q_i^B$  and  $q_i^A$  (as previously described [2]), this formulation allows one to relate the absolute peptide abundance with the total abundance of its parent proteins and thus make inferences about peptide *detectabilities* in addition to relative protein abundances.

We define the detectability of a peptide  $s_i$  as a quantity  $d_i \in [0, 1]$  that relates peptide abundance to the abundances of its parent proteins. In the absence of experimental error, for each peptide  $s_i \in S$ ,

$$\begin{aligned} q_i^B &= d_i \times \sum_{(p_j, s_i) \in E} Q_j^B \\ q_i^A &= d_i \times \sum_{(p_j, s_i) \in E} Q_j^A. \end{aligned}$$

In dealing with errors, we use a linear programming formulation that is similar to  $F_1$ , but with  $2n + 1$  constraints and  $2m + n$  variables. (See Figure 2,  $F_2$  formulation.) We use  $f_i = \frac{1}{d_i}$  as the reciprocal of detectability to maintain linearity of equations. The previous discussion regarding reliable estimates of abundance values is unchanged from the previous section.

The iTRAQ data does not provide peptide abundance values that can be used directly for  $F_2$ . However, recent developments indicate that the absolute peptide abundances can be experimentally estimated [2]. Also, recent results have shown that peptide detectabilities can be reliably estimated with very little variability across mass spectrometry runs [18]. This observation is especially important in  $F_2$ . The knowledge of peptide detectabilities implies  $2m$  variables instead of  $2m + n$  and greatly increases the number of cases that can be solved.

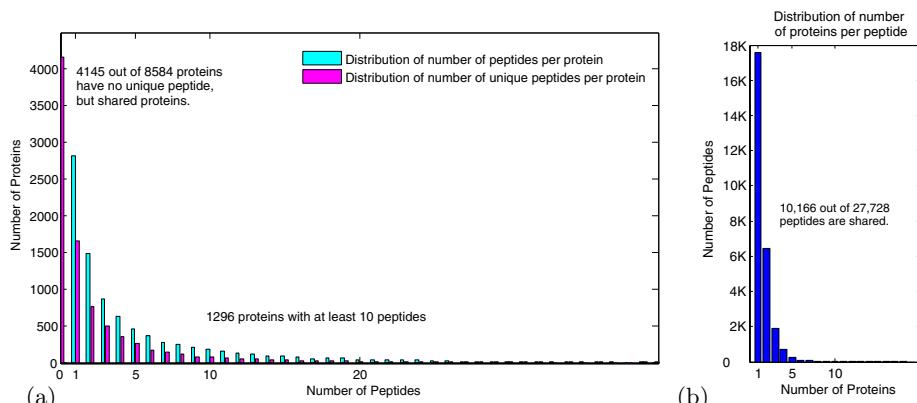
## 3 Results

*Data-set:* We choose an *Arabidopsis* model of root-knot nematode infection. The root-knot nematodes are worm-like, microscopic plant-parasites that infect a multitude of plants, including all major crops, turf, and many ornamental plants. The diversity and extent of infection makes it economically significant to explore. The typical mode of infection is via the root. The female nematode lays its eggs at the root tip. The juveniles infect via the root tip, and move up. Inside, they manipulate the cellular machinery to create specialized *feeding* cells, which grow and multi-nucleate, but do not divide, eventually forming giant cells that provide nutrients to the parasite [3,10]. As the nematodes exploit the

Arabidopsis cellular machinery to create the giant cell phenotype, an analysis of proteins that are differentially expressed in infected versus non-infected host cells can help elucidate the underlying mechanism [4]. As the Arabidopsis genome is sequenced, with extensive annotation on the known genes and pathways, it is an appropriate model for the host.

An iTRAQ method was used to collect protein abundance information. A brief overview of the method is given here (See [1] for details). The samples are enzymatically digested into short peptides. Peptides from different samples are *N*-terminally covalently labeled with tags of different mass, but then pooled and analyzed together using tandem mass spectrometry. Each spectrum contains both the fragment masses used to identify the peptide, and the intensities of the differential tags for abundance computation. In our terminology, for every peptide  $s_i$ , we read the intensities of the two tags as  $q_i^A$   $q_i^B$ , and compute the ratio  $r_i = q_i^A/q_i^B$ , which approximates the ratio of the peptide abundance values in the two samples.

Our data-set is a collection of 118,426 spectra, encoding 27,728 peptides mapping onto 8,584 protein sequences. Each protein is mapped to at least one peptide and vice versa. The number of peptides mapping to a protein sequence varies considerably, ranging from 1 to 59. The distribution of the number of peptides per protein, is shown in Figure 3. Close to half of the proteins (4,145 out of 8,584) do not have a unique peptide. The number of unique peptides per protein range from 0 to 51. The distribution of the number of unique peptides per protein is shown in Figure 3. Likewise, there is tremendous spectral redundancy among peptides, with the number of spectra encoding a peptide ranging from 1 to 975. Close to half of the peptides (10,166) are shared by multiple protein sequences. We reduce the data by merging *symmetric* peptides, or peptides that belonged to an identical subset of proteins. The redundancy helps understand the measurement error, and the merging removes artificial dimensionality,



**Fig. 3.** Mapping of peptides to proteins in *Arabidopsis* root-knot nematode infection iTRAQ data. (a) Distribution of the number of peptides and unique peptides per protein. (b) Distribution of the number of proteins per peptide.

giving a better measure of rank, and rank-threshold. Likewise, we also merge the symmetric proteins for reasons mentioned earlier.

After merging, we obtain a protein-peptide bi-partite graph  $G = (P \cup S, E)$ , where  $|P| = 6,998$ ,  $|S| = 8,069$ ,  $|E| = 13,055$ .  $G$  has 4119 connected components projecting onto 257 non-isomorphic topologies with size ranging from 2 to 127. In this study, we consider only the 1190 non-trivial components, with at least 2 proteins.

In addition to testing on this data, we also perform a series of controlled experiments by simulating data-sets based on the topologies of the Arabidopsis data-set.

*Generation of simulation data:* We start with 257 topologically distinct (non-isomorphic) components of the Arabidopsis data, and generated 100 data-sets from each topology with different values<sup>1</sup>. For each component, we do the following:

1. Sample protein amounts  $\mathbf{Q}^B = [Q_1^B, \dots, Q_m^B]$  at random from the collection of iTRAQ tag intensities.
2. Generate ratio  $R_j$  by sampling from a log-normal  $N(0, \sigma_R)$  distribution.  $\sigma_R$  is set to 0.7 which is the estimated standard deviation of the log peptide ratios in the Arabidopsis data. Compute  $Q_j^A = R_j Q_j^B$ .
3. For each peptide  $s_i$ , generate  $d_i$  uniformly from  $(0, 1]$ , and compute  $q_i^A = d_i \sum_{(p_j, s_i) \in E} Q_j^A$ ,  $q_i^B = d_i \sum_{(p_j, s_i) \in E} Q_j^B$ . When detectabilities are not incorporated, choose  $d_i = 1$ .
4. Compute peptide ratios  $r_i$ , and perturb according to a log-normal  $N(0, \sigma)$ , over a range of values  $\sigma$ . Denote  $\sigma$  as *perturbation level*.

We consider the *system* of constraints for each data-set.

Once the data is generated, only the peptide ratios  $r_i$  are used as inputs. The linear programs are solved for  $\mathbf{Q}^{B'} = [Q_1^{B'}, \dots, Q_m^{B'}]$ , and  $\mathbf{Q}^{A'} = [Q_1^{A'}, \dots, Q_m^{A'}]$  using ILOG OPL Development Studio 6.1.<sup>2</sup> The reliability of the estimates is tested using three measures.

*Validation statistics:* Recall that the value of the optimized objective is a weak indicator of the quality of results. For the simulations, as the protein abundances are known, we can compute the error in the estimate as the protein-abundance-distance, PAD:

$$\text{PAD}(\mathbf{Q}^{B'}, \mathbf{Q}^B) = \frac{\|\mathbf{R}^B\|_1}{m} \quad (3)$$

where  $\mathbf{R}^B = \left[ \ln \frac{Q_i^{B'}}{Q_i^B} \right]$ . While any norm can be used as a valid measure of distance, the choice of the 1-norm, averaged over the dimensions can be loosely interpreted as average fold difference between actual and estimated protein

---

<sup>1</sup> Simulation data - <http://www.cse.ucsd.edu/~bdost/downloads/SimData.zip>

<sup>2</sup> Source code (C#)-<http://www.cse.ucsd.edu/~bdost/downloads/PQPLinearProg.zip>

abundances. The true protein abundances are not available for the Arabidopsis iTRAQ data, so we compute an indirect measure LRD, defined as

$$\text{LRD}(\mathbf{r}, \mathbf{r}') = \frac{\|\mathbf{r} - \mathbf{r}'\|_1}{n} \quad (4)$$

where  $\mathbf{r} = [\ln(r_1), \dots, \ln(r_n)]$  is the vector of experimental peptide log-ratios for the  $n$  peptides, and  $\mathbf{r}'$  describe the peptide log ratios computed by using the estimated protein abundances. Intuitively, if the protein abundance estimates are accurate, the computed peptide log-ratios should match the experimental log-ratios. In a similar way, we compute the peptide log detectability distance LDD as the average 1-norm of the logs of detectabilities. We use PAD, LRD, and LDD to test performance on simulations.

### 3.1 Results of Simulation

As the reliability of the estimates depend upon rank of  $\mathcal{A}$ , we loosely group each of  $100 \times 257$  simulated systems into three categories according to  $\text{rank}(\mathcal{A})$  for a fixed rank-threshold  $t$ , as follows:

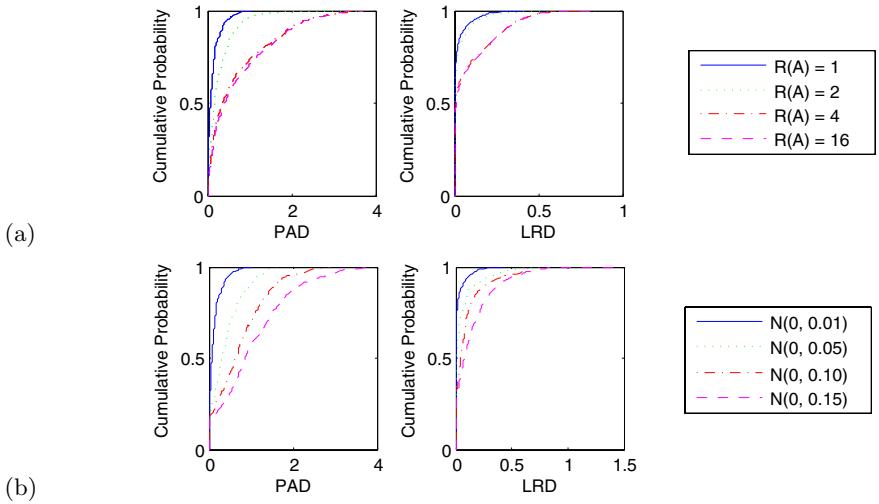
- Category I :  $\text{rank}_t(\mathcal{A}) = 2m \leq n + 1$  (Over-determined, full-rank systems).
- Category II:  $\text{rank}_t(\mathcal{A}) < 2m \leq n + 1$  (Ill-conditioned systems).
- Category III :  $\text{rank}_t(\mathcal{A}) \leq n + 1 < 2m$  (Under-determined systems).

At rank-threshold 1, we obtain 1074, 3926, and 20700 systems in Categories I, II, and III for the  $F_1$  simulation, and similar distributions for  $F_2$ . As the rank-threshold is increased, some of the Category II systems move into Category I (Table II). Additionally, the performance of under-determined systems is uniformly worse than the other two (data not shown). Therefore, we will focus on Category I evaluation using different rank-thresholds. For each category, and each validation statistic, we compute *cumulative-probability* as the fraction of systems in that category that achieve a certain distance or lower. The ideal case is when the cumulative probability is 1 at distance 0.

In the absence of noise, we achieve the ideal case, zero PAD and LRD, for all Category I systems at rank-threshold 4. As noise is introduced to data and less stringent rank-thresholds is used, we deviate from the ideal case. Figure 4a shows the cumulative probability distribution against PAD, and LRD at perturbation level 0.01. Out of 1074 Category I systems at rank-threshold 1, 75% have PAD error of less than 0.16, and an LRD error of less than 0.01. The performance degrades for higher rank-thresholds. Note that in all cases, the optimized objective

**Table 1.** Simulation Category I systems grouped according to their rank-thresholds

	$\mathcal{R}(\mathcal{A})$				
	1	2	4	8	16
$F_1$ - Category I	1074 (4.2%)	2514 (9.8%)	3044 (11.8%)	3980 (15.5%)	4388 (17.1%)
$F_2$ - Category I	663 (2.6%)	2251 (8.8%)	2955 (11.5%)	3104 (12.1%)	4989 (19.4%)

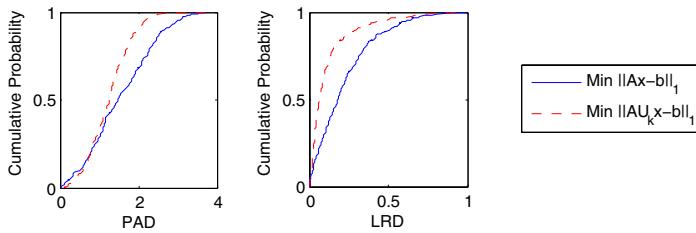


**Fig. 4.** Simulation results using  $F_1$  formulation. Cumulative probability of PAD and LRD for Category I systems (a) perturbation level 0.01, but different rank-thresholds (b) at rank-threshold 1, but different perturbation levels. In all cases, we measure the fraction of systems that were estimated within a certain distance.

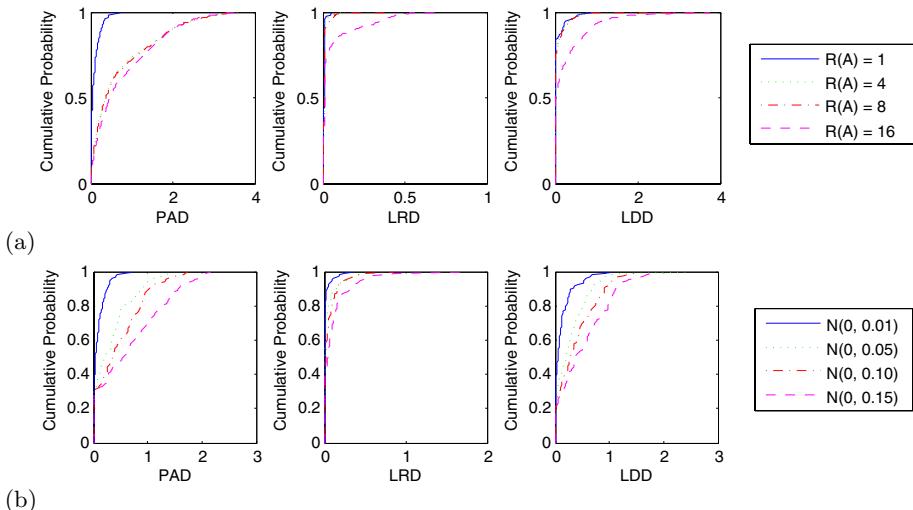
is very close to 0 ( $\sim 10^{-4}$ ). However, in the ill-conditioned and under-determined systems, multiple solutions will lead to a low-error solution, and an arbitrarily picked solution will have high PAD and LRD error. The performance also degrades with an increase in perturbation error. Figure 4b plots the cumulative ratio for Category I systems at rank-threshold 1 under increasing perturbation levels. Thus while 93% of systems show LRD of at most 0.1 at perturbation 0.01, the number falls to 55% at perturbation 0.15.

*Ill-conditioned systems.* We identified a number of Category II systems where the rank-threshold was poor, but only because a small number of singular values were close to 0. For example, in a simulation with perturbation level 0.05, we observed 339 systems for which fewer than 3 singular values were at most  $10^{-16}$ , and  $\text{rank}_1(\mathcal{A}) \geq 2m - 3$  (remaining s.v.  $\geq 10^{-1}$ ). Our results show that the revised LP, suggested for ill-conditioned systems in Section 2, indeed provides better estimates of protein abundance values of these systems. (See Figure 5.) For example, over 88% of the systems from the revised LP achieve an LRD of 0.25 or better, compared to 65% from the original formulation.

*Peptide detectabilities.* A similar behavior is observed during estimation of peptide detectabilities (Figure 6a,b). The performance of PAD, and LRD surprisingly does not change with the addition of an extra  $n$  unknowns (also,  $n$  new constraints get added). We also plot the performance of detectability estimates. As expected, the performance is acceptable for low rank-threshold systems and low perturbations, but degrades for higher rank-thresholds, and higher perturbation



**Fig. 5.** Improved estimates of protein abundance values using SVD based projection on Category II (over-determined but ill-conditioned systems)

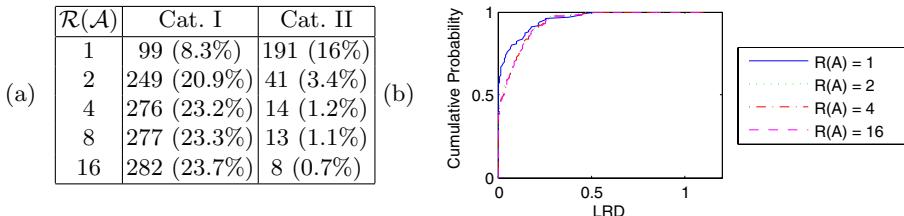


**Fig. 6.** Simulation results using  $F_2$  formulation. Cumulative probability of PAD, LRD and LDD for Category I systems (a) at different rank-thresholds (b) at rank-threshold 1, but different perturbation levels. In all cases, we measure the fraction of systems that were estimated within a certain distance.

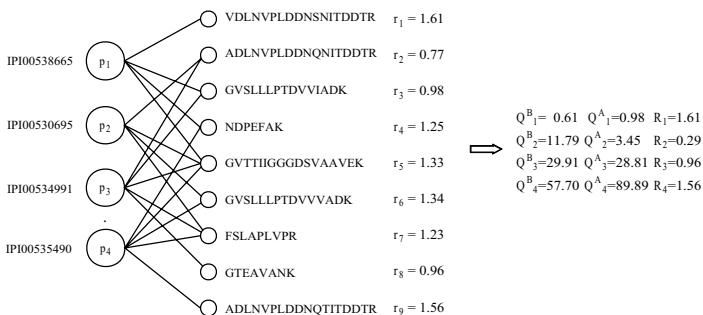
levels. The detectability estimates are robust as well, and degrade in a similar manner.

### 3.2 Arabidopsis iTRAQ Data

We focus on the 1190 non-trivial systems from the iTRAQ data comparing infected samples to non-infected ones. iTRAQ data is not appropriate to get peptide abundance values reliably, so we only use the  $F_1$  formulation on this data-set. The distribution of systems in different rank categories is described in Figure 7a. A total of 99 systems fall into Category I with the most stringent rank-threshold 1, covering 219 proteins and 357 peptides. In addition to relative protein abundances, we estimate abundance ratios across samples for 4 proteins which do not have a unique peptide.



**Fig. 7.** Arabidopsis root-knot nematode infection iTRAQ data. (a) Number of systems in Category I and II at different rank-thresholds. (b) Empirical cumulative probability distribution of LRD for Category I systems at different rank-thresholds.



**Fig. 8.** PhosphoGlycerate kinase family members sharing peptides

As actual protein abundances are not known, we use LRD to evaluate the estimates. The LRD statistic of different categories is as expected with this group performing better than the other groups. Within the 99 systems, 79 have LRD smaller than  $10^{-1}$ , and 55 have LRD smaller than  $10^{-4}$ . The list of systems, constituent peptides, and protein abundance values are shown in online supplemental data<sup>3</sup>. Here, we cherry-pick a few representative examples that point to the differential expression of individual sequences in response to the infection.

**Ca<sup>2+</sup> ATPases:** One of the systems is encoded by 3 proteins from the P-type Ca<sup>2+</sup> ATPase super-family involved in Ca<sup>2+</sup> transport. The three members are the plasma-membrane bound AT5G57110 (ATPase 8), AT4G2990 (ATPase 10), and AT3G21189(ATPase 9). Earlier reports have suggested that ATPase8,10 are co-expressed evenly over all vegetative tissues, while ATPase 9 is expressed almost exclusively in pollen [9]. In our data, the 3 form a confected component with 6 peptides, and our analysis showed the relative wild-type expression of the 3 to be 34.3%, 57.5%, 8.22% respectively, confirming this observation. Further, we find that ATPase 8 is 3× over-expressed in the infected state.

<sup>3</sup> Supplemental data -

<http://www.cse.ucsd.edu/~bdost/downloads/Recomb09SuppData.eps>

**Profilins:** The Profilin family encodes proteins that regulate actin cytoskeleton formation. Five profilins are known. The two that are identified in our data (PRF-1,2) are constitutively expressed in all vegetative organs, and a regulatory element in their first intron is suspected to mediate this expression, differentiating them from the other Profilins [5]. In our data-set, the two are in a component with 3 peptides, one shared. Our analysis shows that Profilin-2 has only (13%) of the total abundance, and is further reduced two-fold upon infection.

**Cinnamyl-alcohol dehydrogenases:** A number of genes in Arabidopsis have been annotated as part of the CAD family, an assertion which has subsequently been challenged, pointing instead to the central role of two members (AtCAD4, and AtCAD5) in the CAD metabolic network. These two molecules have expression patterns consistent with lignification at stem tissues. Interestingly, expression was also observed in various non-lignifying zones (e.g. root caps) indicative of a possible role in plant defense [7]. Our results have a single connected component with AtCAD4,5 and 3 peptides. The analysis shows that both proteins are equally abundant, with AtCAD5 (AT4G34230) at 56% in non-infected cells. However, AtCAD5 is significantly ( $1.5\times$ ) over-expressed, while AtCAD4 (AT3G19450) is  $2\times$  under-expressed.

**Phosphoglycerate kinases:** Phosphoglycerate kinases have been previously shown to be differentially expressed during defense response of Arabidopsis [6]. In our data, four proteins from this family are in a component with nine peptides as shown in Figure 8. In this example, along with the relative protein abundances, we also compute the abundance ratio across-sample for IPI00530695 even though it does not have a unique peptide. Our analysis suggests that both IPI00538665 and IPI00535490 are  $1.5\times$  over-expressed, but IPI00535490 is much more abundant in both samples. IPI00530695 is  $3\times$  under-expressed while the abundance of IPI00534991 does not change.

## 4 Discussion

The extent of peptide sharing in proteomics is under-estimated; consequently, shared peptides, and proteins with non-unique peptides are typically discarded, corresponding to as much as 50% of the data in our experience. As mass spectrometry based protein quantification becomes routine, shared peptide analysis will be increasingly important. Our results are the first to show that a careful analysis not only helps in recovering abundance values of some of these proteins, but also helps quantify the relative levels of different proteins. These across-protein relative abundance computations can help elucidate these differential regulation of the proteins from a family. We investigate topological and numerical considerations in estimating reliability of our computations. Nevertheless, the final quality of the results does depend upon the accuracy of the experimental abundance computations [28]. As the mass spectrometers become more accurate, experimental variation in relative abundance computations will decrease, increasing the power of our methods. In a similar fashion, the estimation

of peptide detectabilities is in an early stage of development. Our results attest to the viability of using shared peptides for detectability computation, but also point to the importance of detectability values in extending the scope of shared peptide analysis. The model's ability to automatically estimate peptide detectabilities may result in an ongoing cycle of self-refinement where different systems resulting from different experimental conditions may allow one to continuously expand the set of known detectabilities, which in turn would allow for the resolution of more complicated systems. In fact, we note that this progressive convergence towards an extensive database of peptide detectabilities may even allow one to learn more about systems that were previously not solvable in a given experiment by adding information from different or even additional targeted experiments aimed at estimating the necessary detectabilities.

A final contribution of this paper is the use of novel evaluation methods for shared peptide computations. Clearly, different algorithms can be used to optimize the error in estimation, including non-linear optimization and other machine learning approaches. We have experimented using simulated annealing approach with a non-linear cost function that minimizes the absolute sum of differences between observed and expected peptide ratios. While such approach is more time consuming, it provides better estimates for systems with unbalanced protein abundances as linear programming formulation is biased towards the error terms associated with the more abundant proteins. Details of that study will be discussed somewhere else. This paper describes a systematic simulation based framework to compare, and develop improved methods for shared peptide analysis.

## Acknowledgments

The research was supported by the National Center for Research Resources of NIH via grant P-41-RR24851.

## References

1. Alves, P., Arnold, R.J., Novotny, M.V., Radivojac, P., Reilly, J.P., Tang, H.: Advancement in protein inference from shotgun proteomics using peptide detectability. In: Pac. Symp. Biocomput., pp. 409–420 (2007)
2. Bantscheff, M., Schirle, M., Sweetman, G., Rick, J., Kuster, B.: Quantitative mass spectrometry in proteomics: a critical review. *Anal. Bioanal. Chem.* 389(4), 1017–1031 (2007)
3. An Advanced Treatise on Meloidogyne: Volume I. North Carolina State University Graphics (1985)
4. Jammes, F., Lecomte, P., de Almeida-Engler, J., Bitton, F., Martin-Magniette, M.L., Renou, J.P., Abad, P., Favery, B.: Genome-wide expression profiling of the host response to root-knot nematode infection in *Arabidopsis*. *The Plant Journal* 44, 447–458 (2005)
5. Jeong, Y.M., Mun, J.H., Lee, I., Woo, J.C., Hong, C.B., Kim, S.G.: Distinct roles of the first introns on the expression of *Arabidopsis* profilin gene family members. *Plant Physiol.* 140, 196–209 (2006)

6. Jones, A.M., Bennett, M.H., Mansfield, J.W., Grant, M.: Analysis of the defence phosphoproteome of *arabidopsis thaliana* using differential mass tagging. *Proteomics* 6(14), 4155–4165 (2006)
7. Kim, S.J., Kim, K.W., Cho, M.H., Franceschi, V.R., Davin, L.B., Lewis, N.G.: Expression of cinnamyl alcohol dehydrogenases and their putative homologues during *Arabidopsis thaliana* growth and development: lessons for database annotations?. *Phytochemistry* 68, 1957–1974 (2007)
8. Mallick, P., Schirle, M., Chen, S.S., Flory, M.R., Lee, H., Martin, D., Ranish, J., Raught, B., Schmitt, R., Werner, T., Kuster, B., Aebersold, R.: Computational prediction of proteotypic peptides for quantitative proteomics. *Nat. Biotechnol.* 25(1), 125–131 (2007)
9. Marmagne, A., Rouet, M.A., Ferro, M., Rolland, N., Alcon, C., Joyard, J., Garin, J., Barbier-Brygoo, H., Ephritikhine, G.: Identification of new intrinsic proteins in *Arabidopsis* plasma membrane proteome. *Mol. Cell Proteomics* 3, 675–691 (2004)
10. <http://www.nematology.umd.edu/rootknot.html>
11. Wiese, S., Reidegeld, K.A., Meyer, H.E., Warscheid, B.: Protein labeling by iTRAQ: a new tool for quantitative mass spectrometry in proteome research. *Proteomics* 7(3), 340–350 (2007)

# Evaluating Between-Pathway Models with Expression Data

Benjamin J. Hescott, Mark D.M. Leiserson, Lenore J. Cowen,  
and Donna K. Slonim

Tufts University Department of Computer Science

**Abstract.** Between-Pathway Models (BPMs) are network motifs consisting of pairs of putative redundant pathways. In this paper, we show how adding another source of high-throughput data, microarray gene expression data from knockout experiments, allows us to identify a compensatory functional relationship between genes from the two BPM pathways. We evaluate the quality of the BPMs from four different studies, and we describe how our methods might be extended to refine pathways.

## 1 Introduction

In this paper, we use microarray expression data to validate instances of an important class of network motif in the yeast interactome. These motifs, called “Between Pathway Models” or BPMs, consist of putative pairs of redundant pathways and were first described by Kelley and Ideker in a seminal paper in 2005 [4]. The basic idea behind the BPM is simple. Protein-protein interaction relationships may be divided into two types: physical interactions and genetic interactions. In particular, the subset of the genetic interactions that are “synthetic lethal” interactions are considered. Synthetic lethality between two yeast genes indicates that strains lacking either individual gene (called “knockout” or “deletion” strains) are viable on rich media, but if the two genes are deleted simultaneously, the yeast is inviable. Kelley and Ideker defined two sets of genes, G1 and G2, to represent a BPM motif if there are few synthetic lethal edges *within* G1 and *within* G2, but many synthetic lethal edges between G1 and G2, and if the opposite holds for the physical interaction edges, i.e., there are many physical interaction edges within G1 and G2, but few between G1 and G2. In their notation and ours, gene sets G1 and G2 are referred to as the “pathways” of this BPM.

The term BPM has been used to describe both purely mathematical substructure within the protein-protein interaction network, but also as a synonym for two sets of genes that function in redundant pathways. It is in the second fashion that we will use the terms “BPM” and “putative or candidate BPM” in this work; that is, independent of the exact definition of the graph theoretic criteria used in each of the papers we consider, the two sets of genes (and their connectivity edges) will be considered a candidate BPM for us, where we will try to validate the assertion that these two sets of genes come from redundant pathways using other high-throughput measures (in the present case, using microarray studies).

The motivation for this terminology, and indeed for the interest in the BPM network motif, is the following. Organisms often evolve a sort of fault-tolerance or redundancy, in that they may have multiple sets of proteins capable of performing certain essential functions. Suppose, for some particular essential function, there are two pathways, each of which can compensate for the other if only one of the pathways is disrupted. Then one would expect to see many physical interactions between the genes in each pathway. One would also expect to see synthetic lethal interactions whenever a gene essential to *each* pathway is simultaneously suppressed or deleted. Thus these pathways will organize into a BPM pattern, with many synthetic lethal edges between G1 and G2 and with many physical interaction edges tending to occur within G1 and within G2.

Compared to Kelley and Ideker [1], a subsequent paper of Ulitsky and Shamir [2] identified BPMs using somewhat different definitions of “many” and “few.” The graph theoretic structures of the Ulitsky and Shamir pathways, and thus of their BPMs, are somewhat different (Ulitsky and Shamir also included “synthetic sick” interactions in their data). Recent work of Ma, Tarone and Li [3] and of Brady et al. [4] redefined BPMs mainly in terms of the structure and placement of the synthetic lethality edges only, evaluating their putative BPMs after the fact by enrichment or the placement of physical interaction edges within the motifs. All four studies claim that their putative BPMs form possible examples of pairs of compensatory pathways in yeast. It is important to stress that regardless of the *algorithm* used to generate the putative BPM, or the exact graph-theoretic structure that is sought, all four studies generate pairs of sets of genes (G1,G2), where they claim that G1 and G2 make up compensatory pathways.

It is hard to validate the claim that two individual sets of genes (G1,G2) make up compensatory pathways given only the networks of protein-interaction and genetic-interaction data. One method might be to see if a study’s putative BPMs *correctly predict* which untested pairs of genes will physically interact or will be synthetically lethal. Note that Brady et al. [4] employ this approach: the method of [4] run on data from a prior BioGRID download was shown to be able to predict the location of “new” synthetic lethal edges in a more current version of BioGRID. However, the standard measure of “goodness of BPMs” agreed on by all four studies [1,2,3,4] has been to show that a substantial fraction of their BPMs exhibit functional coherence, in the form of GO enrichment for the pathways. While this sort of enrichment result provides evidence that the genes in each pathway have some common function, it does not directly demonstrate a relationship between the two pathways unless the enriched functions happen to be related.

In this paper, we show how adding another source of high-throughput data, microarray gene expression data, allows us to identify a compensatory functional relationship between genes from the two pathways. Expression data gives us a temporal snapshot of the cells’ RNA under specific conditions. We can use gene expression data from knockout experiments to help evaluate the quality of the BPMs from all four studies.

In particular, suppose that we had microarray expression profiles of all yeast single-gene deletion strains. Then suppose  $(G1, G2)$  is a putative BPM with pathways  $G1$  and  $G2$ , and let  $g$  be a gene from  $G1$ . If the loss of  $g$  disables pathway  $G1$ , and if pathway  $G2$  is truly compensating for  $G1$ 's loss as the BPM suggests, then we might see the genes in  $G2$  show a coherent change in gene expression in the  $g$ -deletion strain compared to wild-type. We furthermore expect this change to be stronger than what we would see for a random set of genes.

This signal is subtle, but detectable. Our method is similar to that used by the Gene Set Enrichment Analysis (GSEA) method [5] to test for coherent expression of gene sets. We compute a cluster-rank score, analogous to their Enrichment Score, that measures the coherence of expression changes of the genes in the pathway. Because the pathway being validated may include genes involved in different but related roles in the same functional process, we use the absolute value of the log expression change to rank our genes. Thus, a pathway that has strong up-regulation of some genes and strong down-regulation of others in response to a compensatory deletion will still score well.

In practice, we do not currently have expression profiles for all the yeast deletion strains. However, the Rosetta Compendium [6] includes expression profiles of 276 deletion mutants. This is enough data for an initial assessment of many of the BPMs, and certainly enough for method development. In all four studies we find example BPMs that we can validate, though the percentage of validated BPMs is highest for the BPMs of Brady et al. [4].

We note that there have been several successful previous approaches for integrating microarray measurements with protein-protein interaction knowledge. Ideker et al. [7] searched for connected sets of genes with unexpectedly high levels of differential expression. Segal, Wang and Koller [8] and Ulitsky and Shamir [9] used correlated expression data together with protein-protein interaction data to identify sets of genes that putatively act in similar pathways. Liu et al. [10] showed how such analysis could be done in reference to genes deregulated in type 2 diabetes; and Ulitsky, Karp and Shamir [11] most recently did a high-throughput analysis identifying connected gene subnetworks enriched for genes deregulated in different diseases.

## 2 Methods

### 2.1 Computing the ClusterRank Score

Our BPM evaluation method relies on measuring the coherence of changes in the complementary pathway's genes' expression levels. We do this by computing what we call the *ClusterRank Score*. Note that this score is similar to the GSEA Enrichment Score [5], except that the weights are the raw ranks of the *absolute value* of the “log 10” ratio. This allows us to validate pathways whose genes respond strongly, but in different ways, to knockouts in a compensatory pathway.

Let  $(X, Y)$  be the two pathways of a candidate BPM. Without loss of generality, let  $\bar{x} \in X$  be a gene for which there is expression data for the deletion strain. Furthermore, we require that  $Y$  contain at least three genes. For every

gene  $g$ , we define the “log-10-ratio” (as in the Rosetta Compendium data) to be  $Q_{\bar{x}}(g) = \log_{10} \frac{\varepsilon_{\bar{x}}(g)}{\varepsilon^*(g)}$  where,  $\varepsilon_{\bar{x}}(g)$  is the expression of gene  $g$  in the deletion strain  $\bar{x}$  and  $\varepsilon^*(g)$  is the expression of gene  $g$  in wild-type. Let the yeast genes,  $g_1 \dots g_N$  be enumerated such that  $|Q_{\bar{x}}(g_1)| \leq |Q_{\bar{x}}(g_2)| \leq \dots \leq |Q_{\bar{x}}(g_N)|$ . Similar to the GSEA method define:

$$\begin{aligned} hit(i) &= \frac{\sum_{j=i}^N j I_j}{\sum_{j=1}^N j I_j} \\ miss(i) &= \frac{\sum_{j=i}^N (1 - I_j)}{\sum_{j=1}^N (1 - I_j)} \end{aligned}$$

where

$$I_j = \begin{cases} 1 & \text{if } g_j \in \text{pathway } Y \\ 0 & \text{otherwise} \end{cases}$$

We now define the *ClusterRankScore* as:

$$ClusterRankScore(Y_{\bar{x}}) = \max_{i=N to 1} hit(i) - miss(i).$$

Note that genes with the same “log 10” ratio have the same rank.

## 2.2 Estimating Statistical Significance

How can we tell whether the observed score for a particular pathway is good or not? It is hard to know what a good *ClusterRankScore* is because, like the GSEA Enrichment Score, it depends in part on the number of genes in the pathway. Therefore, we generate an appropriate null distribution each time, and estimate significance from that.

Specifically, the *ClusterRankScore* of a pathway  $Y$  is converted to a p-value using a permutation test as follows. We generate 99 random subsets of genes the same size as the complement pathway,  $Y$ ,  $G_1, \dots, G_{99}$ , with  $|G_1| = \dots = |G_{99}| = |Y|$ . For each subset,  $G_i$ , we calculate the *ClusterRankScore*( $G_{i\bar{x}}$ ). The set of all 100 tests are sorted and the p-value is the percentile of *ClusterRankScore*( $Y_{\bar{x}}$ ).

We say a BPM pathway is *validated* if its p-value is below 0.10. This value was chosen as an arbitrary cutoff. In fact, we can simply use these p-values to rank the pathways from most- to least-supported by the given knockout data available. In Figure 5 we look instead at cumulative rates of validation as this cutoff value ranges from zero to one.

We also performed a second test we call the *random-knockout validation test* on a subset of the validated pathways. In this case the *ClusterRankScore* of Pathway  $Y$  with knockout gene  $\bar{x}$ , *ClusterRankScore*( $Y_{\bar{x}}$ ), is compared against the scores of random knockout strains of genes not present in either pathway. Specifically, we find 99 random deletion mutants  $\bar{z}$ ,  $\bar{z} \notin X, \bar{z} \notin Y$  and compute *ClusterRankScore*( $Y_{\bar{z}}$ ). All 100 scores are sorted, and the reported p-value is the percentile of *ClusterRankScore*( $Y_{\bar{x}}$ ) in the list. If the p-value of pathway  $Y$

is less than 0.10 we say that the BPM passes the random-knockout validation test. More about the relative merits of the two validation methods is said in the Discussion.

## 2.3 Data

The Rosetta Compendium [6] includes genome-wide expression profiles for 276 yeast deletion mutants. Each mutant is missing exactly one gene, and for each mutant the expression levels of all yeast genes are measured, compared to wild-type. The  $\log_{10}$  of the mutant-to-wild-type expression ratio is reported.

We consider between pathway models where at least one gene from the 276 knockout strains in the Rosetta data appears somewhere in the BPM and the number of genes in the compensatory pathway is greater than two. We find 160 of the 404 BPMs reported by Kelley and Ideker [1] satisfy these criteria, as do 36 BPMs from Ulitsky and Shamir [2], 54 from Ma et al. [3], and 959 from Brady et al. [4].

Note, that Ma et al. describe producing over 2,000 possible BPMs, which they believe contain many false positives. Since they do not supply this large set of BPMs, when we refer to the dataset of the Ma et al. study, we are referring to the subset of the BPMs that they make available; namely those of their BPMs that were found to be enriched for the same function in both pathways.

**Website.** All data and *ClusterRank* code are available online at <http://bcb.cs.tufts.edu/validatedBPM>

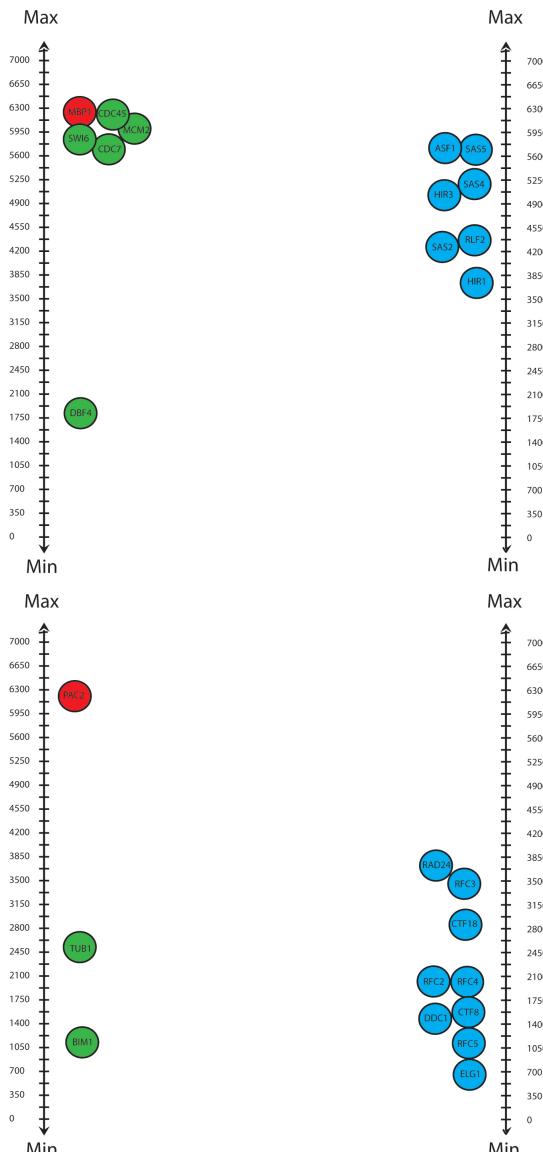
## 3 Results

### 3.1 BPM Validation

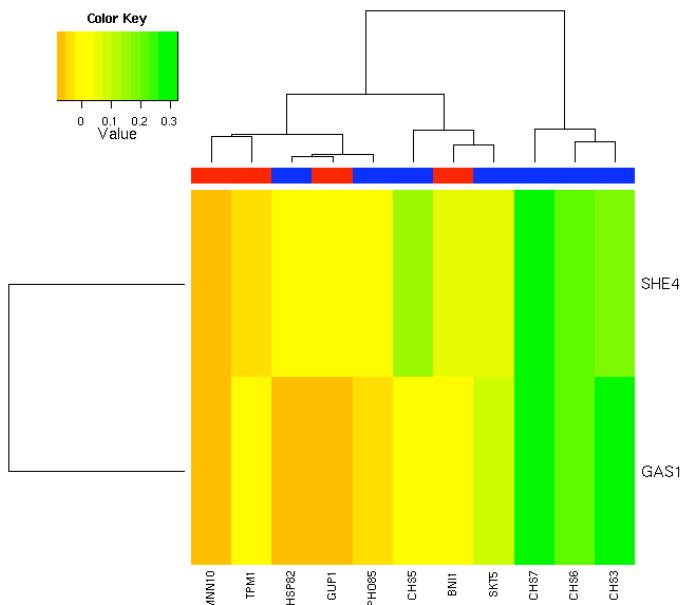
In Figure 1 we show both a validated BPM from Kelley and Ideker and a second BPM that we do not validate. For the BPM on the top, pathway 1 (by convention, shown on the left-hand side) contains the deleted gene MBP1. Note that MBP1 has a high rank, as we take the absolute value of its expression ratio. Notice how the genes in pathway 2 are all differentially regulated when MBP1 is deleted, with ranks ranging from 3742 to 5749, yielding a *ClusterRankScore* of .6 with a *p*-value < .06. In the second BPM in the Figure, PAC2 in pathway 1 is the deleted gene. Note how the differential expression in pathway 2 in response to the loss of gene PAC2 clusters towards the low end, corresponding to an absolute expression change close to zero. The highest ranked expression change, 3718, is below the lowest rank we saw with the previous BPM. These data give the second BPM a *ClusterRankScore* of -0.39, corresponding to a *p*-value of 0.99.

In the validated BPM from this figure, both pathways display functional enrichment: Pathway 1 for the GO Biological Process term “DNA replication initiation,” and pathway 2 for “chromatin assembly.”

When we have data for two or more deletion mutants in the same pathway, we can do more than simply look at the data from each individually. Figure 2 shows



**Fig. 1.** Two BPMs with validation data. The genes in each pathway are represented by a set of dots. The deleted strain's gene is shown in red, other genes in that pathway are green, and genes in the compensatory pathway are shown in blue. Each is plotted next to a number line indicating the ranks of the genes when sorted by their differential expression between the wild-type and mutant strains. The figure on the top shows the data for a validated BPM from Kelley and Ideker. In contrast, the figure on the bottom shows the genes in a BPM we do not validate. Higher ranks correspond to larger changes in expression value (in either direction) in the deletion strains as compared to wild-type.



**Fig. 2.** A heat map of the differential expression of yeast genes in pathway 2 (individual genes in the pathway are shown with red bars at the top) in response to the deletion of two different genes (SHE4 and GAS1) from pathway 1 (shown with blue bars at the top) in a validated BPM of Ma et. al. Notice that genes across both pathways are expressed similarly under the GAS1 and SHE4 deletion mutants.

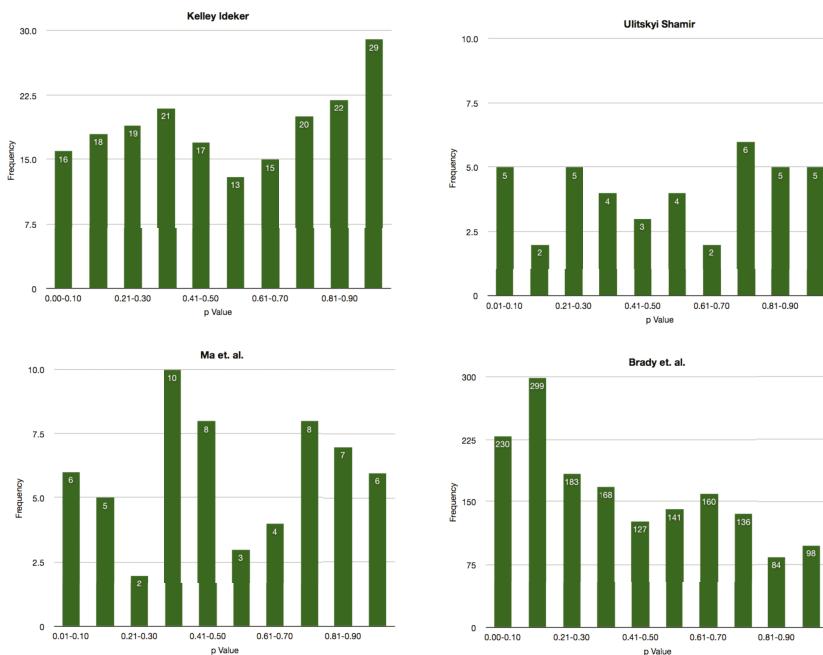
a heat map of a BPM from Ma et al. [3]. Here we have two deletion mutants from pathway 1, SHE4 and GAS1. Both validate the opposite pathway with p-values of 0.02 and 0.09 and ClusterRankScores of 0.64 and 0.56 respectively. With both strains we can compare both pathways and order the genes in terms of up regulation and down regulation. It is clear to see that CHS3, CHS6, and CHS7 are over-expressed within both mutants and that MNN10 is highly down-regulated in both.

In Figure 3 we summarize the number of pathways in each of the four studies whose intersection with the knockout data was non-empty and contained at least three genes. A total of 1209 pathways across all four studies met these criteria, of which we found 257 pathways that we were able to validate (ClusterRank test described above with  $p < .1$ .) The Kelley-Ideker data had 160 pathways meet this criteria with 16 of these pathways validated. Ulitsky-Shamir had 36 pathways tested and 5 of these pathways were validated. Ma, et. al. had 54 pathways meet the criteria with 6 validated pathways. Brady, et. al. had 959 pathways meet the criteria and had 230 of these validate.

To compare the various techniques we consider the percentage of tested pathways and their *ClusterRankScore* with regard to their lowest p-value. For each method we plot the p-value against the percentage of pathways that have a smaller or equal p-value. We saw that 44% of the Brady pathways had values

BPM Data Set	# 3+ BPMs that Intersect Knockout Data	# Validated Pathways	% Validated
Kelley-Ideker (2005)	160	16	10%
Ulitsky-Shamir (2007)	36	5	14%
Ma, et al. (2008)	54	6	11%
Brady, et al. (2008)	959	230	24%

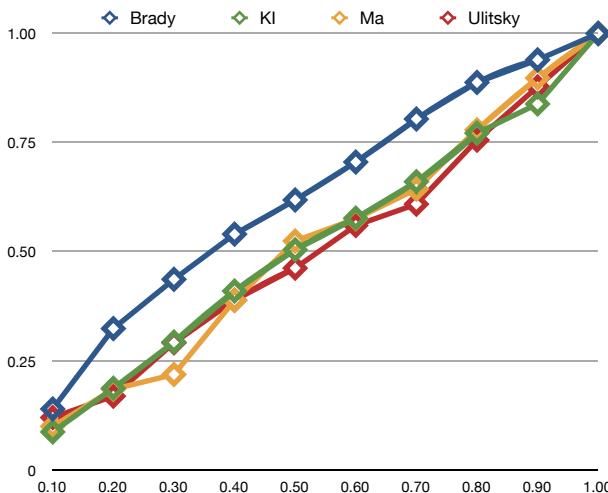
**Fig. 3.** For each method, we consider all BPMs for which we can compute a *ClusterRankScore* based on data currently available, i.e. for BPMs where at least one gene intersects the Rosetta Compendium data and its opposite pathway contains at least three genes. Such a pathway is considered validated if its *ClusterRankScore* has a p-value  $\leq 0.01$ .



**Fig. 4.** Histograms of the distribution of *ClusterRankScores* by p-value for each of the four methods

less than 0.30. Similarly we saw 29% for Kelley and Ideker, 29% for Ulitsky and Shamir, and 22% for Ma et. al., see Figures 4 and 5.

The “validation” described above can be described as a measure for how likely a pathway in a BPM is likely to demonstrate coherent expression changes when a gene in the second pathway is deleted. However, it is possible that a validated pathway would show coherent expression changes when

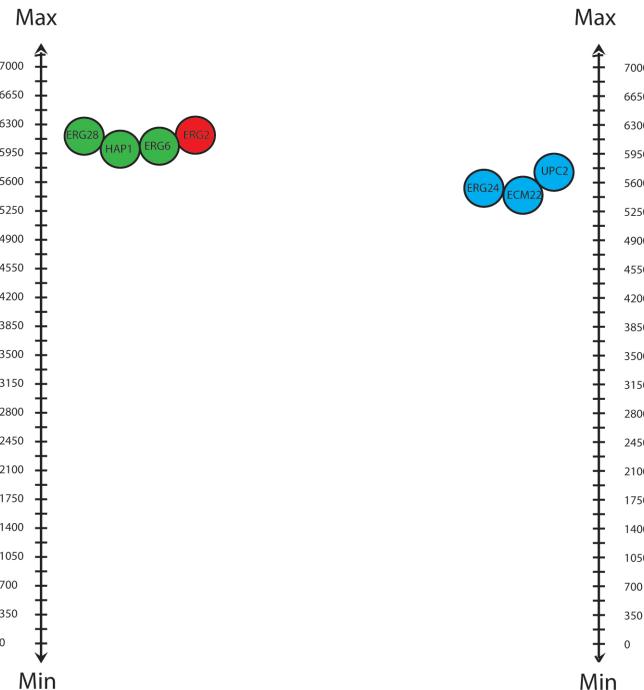


**Fig. 5.** For each p-value, the percentage of pathways tested that have a smaller or equal p-value

many genes (both on and off the second pathway) were deleted, or even just change coherently in the wild-type strain in response to different conditions. If the validated pathway shows *more* coherent expression changes when a gene on its other BPM pathway is deleted than when a random gene is deleted, we say it passes the random-knockout validation test. This more stringent test is again, something that we don't have enough data to apply in a high-throughput fashion; but this stronger test in fact shows that the two pathways are connected to each other, not just individually coherent.

In Figure 6 we see a validated BPM from Brady. Pathway 1 contains ERG8, HAP1, ERG6, with knockout ERG2; the complementary pathway contains ERG24, ECM2, and UPC2. Here we have validation that pathway 2 clusters nicely in the ERG2 deletion strain against random gene sets of size 3: we observe a p-value of 0.01 for its *ClusterRankScore*. We also know that this pathway clusters best with this particular deletion mutant as compared to random deletion mutants in the Hughes data set: the BPM passes the random-knockout validation test with a p-value of < 0.01.

In fact, this turns out to involve interesting and important genes. In particular, these genes are all involved in the ergosterol biosynthesis pathway, required for generation of a major constituent of the fungal plasma membrane, ergosterol [12][13]. This pathway has been extensively studied as a target of antifungal drugs [14]. The synthesis of lanosterol is the first step in the pathway dedicated to yeast biosynthesis in wild-type yeast, and ERG24 is required to complete C-14 demethylation of lanosterol. [15]. Later in the pathway, ERG6 converts zymosterol to fecosterol, which is then converted by ERG2 to episterol [16]. ERG28 has recently been shown to be a scaffolding protein that physically interacts with ERG6. HAP1, ECM22 and UPC2 are all transcription factors that are involved



**Fig. 6.** A BPM from Brady et al. that passes both the random-gene validation and the random-knockout validation tests

in the regulation of the ergosterol biosynthesis pathway. Many deletion strains with single mutants of these genes (or double mutants of genes on the same side of the partition) result in accumulation of atypical sterol intermediates, viable, at least in certain genetic conditions and on rich media; for example, single mutants of ERG24 causes the accumulation of the aberrant sterol ignosterol [16]. Based on the BPM structure and the expression data, we postulate that there is some mechanism of compensation involving the transcription factors ECM22 and UPC2 (perhaps to remove toxic sterol intermediates) when the late stages of the ergosterol biosynthetic pathway is disrupted. On the other hand, when ERG24 is deleted, perhaps HAP1 can upregulate other genes to synthesize aberrant sterols that let the yeast hang on to viability in certain favorable conditions. If the two subportions of the pathway are thus compensating, stronger antifungal drugs might result from targeting genes in both subsets simultaneously.

## 4 Discussion

Understanding the functional relationships between proteins is essential for the interpretation of genomic data sets. Manual, experimental construction of pathway models is a slow and painstaking process [17][18]. Thus, there is a need for computational methods to predict pathway models and functions. Between-pathway

models are an especially advantageous method of pathway-mining, in that the compensatory nature of the two complementary pathways may provide additional clues to function.

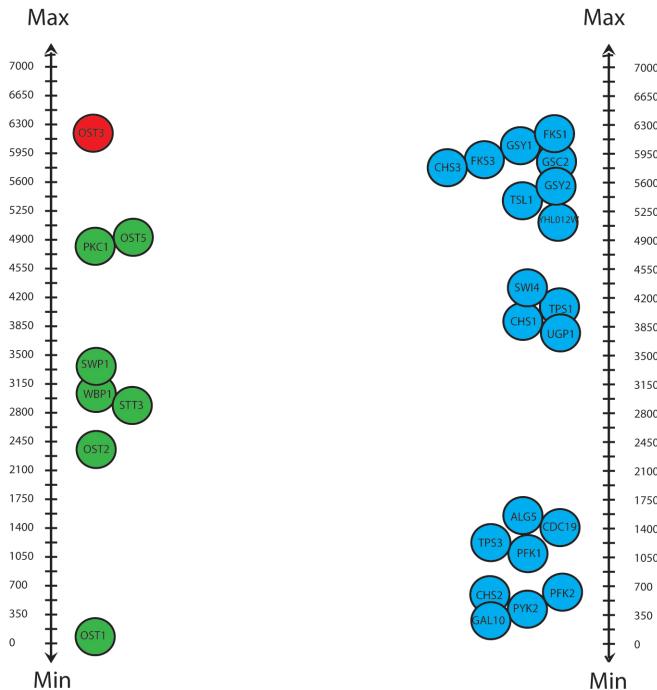
Here, we suggest a method for evaluating the results of BPM discovery projects. However, our method is still only a pragmatic approximation of our ultimate goal, which would be reachable if we had gene expression data for all yeast single-gene deletion strains. In that case, we could reasonably expect to find knockouts of most genes on both sides of a BPM. We would then be able to look for consistent ClusterRank scores across most genes in each pathway from a BPM, and for a relationship between the scores of the knockouts on both sides. Even those with only middling p-values might be validated if knocking out every gene in one pathway produces the same set of coherent changes in the other, and vice-versa.

We have instead worked with the Rosetta compendium - the best source of such data available, but still containing knockouts of only about 5% of the yeast genome. Thus, finding BPMs with more than a few deleted genes is difficult, and we have needed to adjust our methods accordingly. We note that, because few BPM pathways contain more than just one deleted gene, we may fail to validate some pathways that are simply noisy rather than incorrect. For example, suppose that a pathway contains 8 pathway-related genes and 2 unrelated ones. If the one deletion strain we have is one of the unrelated genes, then the pathway will not have been validated in our study. However, the addition of more data would address this problem.

Similarly, this lack of data has had an effect on our methods to evaluate statistical significance. If we had the full set of knockout data, then the random-knockout validation method would be the preferred approach. This would be a much stronger way of assessing the significance of coherent expression changes in *the specific gene set* than comparison to a random set of genes. (This is because we know that random sets of genes have low expression correlation, while it is possible that a particular gene set is co-expressed, wildly variable, and completely unrelated to the knockout from the complementary pathway.) However, in this paper, we still prefer random-gene-set permutations over random-mutant-strain permutations, simply because the set of mutants considered in our data set is so small (276), and it is biased heavily towards those genes whose deletion mutants were considered most interesting to investigators. Thus, selecting 100 random mutant strains from this set is almost certainly heavily biased towards specific functions, many of which may well be intentionally correlated with those of the pathways being evaluated.

We point out that the BPMs of Brady et al. represent the work of our own group (LC) or colleagues. We worked independently to design a validation method that made the most sense, and we at first applied it only to the other three data sets (whose combined sizes made their evaluation much more efficient). Only once the methods were fixed did we attempt to evaluate the final Brady data set using our method.

Finally, we note that for specific BPMs, it may be becoming financially feasible for interested investigators to obtain expression profiles of all BPM genes in all or nearly all BPM knockout strains (perhaps by taking advantage of new custom array technologies), and thus having complete data sets may ultimately be a



**Fig. 7.** A BPM from Kelley and Ideker not validated by our method. This BPM illustrates clustering of subsets of genes within the compensatory pathway.

reality. Once such data are available, our method might be applied to help refine the specific pathway models further. Imagine what Figure 2 would look like if it contained data for not just two deletion strains, but deletions for all the genes in one pathway from the BPM. Clustering the rows of the matrix might begin to allow a potential ordering of the genes in the pathway itself.

In fact, even with the data we have now, some pathway refinement is possible. Figure 7 shows a BPM from Kelley and Ideker and, on the right-hand side, the ranks of a pathway's gene expression changes in deletion mutant OST3. Among the 20 genes on the right-hand pathway, we see three different clusters of coherent expression changes. The gene set FKS1, CHS3, FKS3, GSY1, GSC2, GSY2, TSL1, YHL012W all have high ranks, another set of four genes have moderate ranks, and a third set show little change in their expression within this deletion mutant as compared to wild-type. One interesting possibility is that this BPM should perhaps be refined to include just the high-ranking genes.

## Acknowledgments

BH and LC were supported in part by NIH grant 1R01GM080330-01A1, ML was supported in part by NSF grant (ASE+NHS)(dms)0428715 and DS was supported in part by NIH grant R21 LM004911.

## References

- Kelley, R., Ideker, T.: Systematic interpretation of genetic interactions using protein networks. *Nature Biotechnology* 23(5), 561–566 (2005) doi: 10.1038/nbt1096 PMID: 15877074
- Ulitsky, I., Shamir, R.: Pathway redundancy and protein essentiality revealed in the *S. cerevisiae* interaction networks. *Molecular Systems Biology* 3(104) (2007) PMCID: PMC1865586
- Ma, X., Tarone, A., Li, W.: Mapping genetically compensatory pathways from synthetic lethal interactions in yeast. *PLoS One* 3(4), 1922 (2008) doi:10.1371/journal.pone.0001922 PMCID: PMC2275788
- Brady, A., Maxwell, K., Daniels, N., Cowen, L.: Fault tolerance in protein interaction networks: Stable bipartite subgraphs and redundant pathways. *PLoS ONE* (to appear)
- Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Ladner, E.S., Mesirov, J.P.: Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America* 102(43), 15545–15550 (2005) PMCID: PMC1239896
- Hughes, T.R., Marton, M.J., Jones, A.R., Roberts, C.J., Stoughton, R., Armour, C.D., Bennett, H.A., Coffey, E., Dai, H., He, Y.D., Kidd, M.J., King, A.M., Meyer, M.R., Slade, D., Lum, P.Y., Stepaniants, S.B., Shoemaker, D.D., Gachotte, D., Chakraburty, K., Simon, J., Bard, M., Friend, S.H.: Functional discovery via a compendium of expression profiles. *Cell* 102(1), 109–126 (2000) doi: 10.1016/S0092-8674(00)00015-5 PMID: 10929718
- Ideker, T., Ozier, O., Schwikowski, B., Siegel, A.: Discovering regulatory and signaling circuits in molecular interaction networks. *Bioinformatics* 18, S233 (2002) PMID: 12169552
- Segal, E., Wang, H., Koller, D.: Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics* 19(suppl. 1), 264–271 (2003) PMID: 12855469
- Ulitsky, I., Shamir, R.: Identification of functional modules using network topology and high-throughput data. *BMC Systems Biology* 1(8) (2007) PMCID: PMC1839897
- Liu, M., Liberzon, A., Kong, S., Lai, W., Park, P.J., Kohane, I.S., Kasif, S.: Network-based analysis of affected biological processes in type 2 diabetes models. *PLoS Genetics* 3(6), e96 (2007) PMCID: PMC1904360
- Ulitsky, I., Karp, R., Shamir, R.: Detecting disease-specific dysregulated pathways via analysis of clinical expression profiles. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 347–359. Springer, Heidelberg (2008)
- Paultauf, F., Kohlwein, S., Henry, S.: Regulation and compartmentalization of lipid synthesis in yeast. *The Molecular and Cellular Biology of the yeast Saccharomyces: Gene Expression* 2, 415–500 (1992)
- Parks, L., Casey, W.: Physiological implications of sterol biosynthesis in yeast. *Annual Review of Microbiology* 49(1), 95–116 (1995) doi: 10.1146/annurev.mi.49.100195.000523 PMID: 8561481
- Lupetti, A., Danesi, R., Campa, M., Tacca, M.D., Kelly, S.: Molecular basis of resistance to azole antifungals. *Trends in Molecular Medicine* 8(2), 76–81 (2002) doi: 10.1016/S1471-4914(02)02280-3 PMID: 11815273

15. Bhuiyan, M.S.A., Eckstein, J., Barbuch, R., Bard, M.: Synthetically lethal interactions involving loss of the yeast erg24: the sterol c-14 reductase gene. *Lipids* 42(1), 69–76 (2007) PMCID: PMC1847747
16. Valachovic, M., Bareither, B., Bhuiyan, M.S.A., Eckstein, J., Barbuch, R., Balderes, D., Wilcox, L., Sturley, S., Dickson, R., Bard, M.: Cumulative mutations affecting sterol biosynthesis in the yeast *Saccharomyces cerevisiae* result in synthetic lethality that is suppressed by alterations in sphingolipid profiles. *Genetics* 173(4), 1893–1908 (2006) PMCID: PMC1569731
17. Goto, S., Bono, H., Ogata, H., Fujibuchi, W., Nishioka, T., Sato, K., Kanehisa, M.: Organizing and computing metabolic pathway data in terms of binary relations. In: *Pac. Sympos. Biocomp.* (1997) PMID: 9390290
18. Davidson, E., Rast, J., Oliveri, P., Ransick, A., Calestani, C., Yuh, C., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., Otim, O., Brown, C., Livi, C., Lee, P., Revilla, R., Rust, A., Pan, Z., Schilstra, M., Clarke, P., Arnone, M., Rowen, L., Cameron, R., McClay, D., Hood, L., Bolouri, H.: A genomic regulatory network for development. *Science* 295(5560), 1669–1678 (2002) doi:10.1126/science.1069983 PMID: 11872831

# Sorting Signed Permutations by Inversions in $O(n \log n)$ Time

Krister M. Swenson, Vaibhav Rajan, Yu Lin, and Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics  
EPFL (École Polytechnique Fédérale de Lausanne), Switzerland  
`{krister.swenson,vaibhav.rajan,yu.lin,bernard.moret}@epfl.ch`

**Abstract.** The study of genomic inversions (or reversals) has been a mainstay of computational genomics for nearly 20 years. After the initial breakthrough of Hannenhalli and Pevzner, who gave the first polynomial-time algorithm for sorting signed permutations by inversions, improved algorithms have been designed, culminating with an optimal linear-time algorithm for computing the inversion distance and a subquadratic algorithm for providing a shortest sequence of inversions—also known as sorting by inversions. Remaining open was the question of whether sorting by inversions could be done in  $O(n \log n)$  time.

In this paper, we present a qualified answer to this question, by providing two new sorting algorithms, a simple and fast randomized algorithm and a deterministic refinement. The deterministic algorithm runs in time  $O(n \log n + kn)$ , where  $k$  is a data-dependent parameter. We provide the results of extensive experiments showing that both the average and the standard deviation for  $k$  are small constants, independent of the size of the permutation. We conclude (but do not prove) that almost all signed permutations can be sorted by inversions in  $O(n \log n)$  time.

## 1 Introduction

Genomic rearrangements have been the subject of intense research over the last 10 years. Initially identified in the 1920s in the fly genome through genetic studies [14][15], then studied in detail in chloroplast organelles in the 1980s (for instance in a series of papers from Palmer’s lab, beginning with [9][10]), they were brought to the attention of the computational community in the early 1990s [11]. A large number of papers have since been published on the combinatorics and algorithmics of genomic rearrangements (see [8] for a survey). Starting at the beginning of this century, genomic rearrangements have assumed much more importance with the advent of whole-genome sequencing and the emergence of comparative genomics as a major discipline in biocomputing.

Of the various genomic rearrangements studied, perhaps the simplest and best documented is the *inversion* (also called reversal in much of the Computer Science literature), through which a segment of a chromosome is reversed in place. In 1987, Day and Sankoff [5] formalized a model of genomic inversions in which a chromosome is represented as a permutation of signed gene indices, the sign indicating the direction of transcription of the gene; in this framework, an inversion acts on an interval of the permutation by reversing the order in which the indices appear within the interval and by flipping the sign of each index. Sankoff later provided a probabilistic model [12]

and posed two fundamental questions about inversions in this framework: given two signed permutations on the same index set, what is the smallest number of inversions required to transform one permutation into the other and what is a sequence of inversions implementing this transformation [1]. The first problem is thus to compute an edit distance, where the edit operation is the inversion; the second is to return an edit sequence—a problem usually known as “sorting,” since a simple re-indexing can turn one of the permutations into the identity. Many years of work were needed to ascertain the complexity of each of these problems. The breakthrough came in 1995, when Hannenhalli and Pevzner provided a polynomial-time algorithm to solve both problems. (In contrast, in 1997, Caprara [4] showed that both problems were NP-hard if phrased in terms of unsigned permutations.) The running time for both problems has been steadily reduced over the years. In 2001, our group gave an optimal linear-time algorithm to compute the edit distance [1]; and in 2004, Tannier and Sagot, building on the work of Kaplan and Verbin [7], gave a  $O(n\sqrt{n \log n})$  algorithm to produce a sorting sequence. Remaining open was the question of whether signed permutations can be sorted by inversions in  $O(n \log n)$  time, just like sorting plain numbers.

In this paper, we give a qualified positive answer to this question by describing two new algorithms for sorting signed permutations by inversions. The first is a randomized algorithm that runs in guaranteed  $O(n \log n)$  time, but may fail; successive restarts reduce the probability of failure, but we cannot guarantee that every permutation will be sorted with high probability with a finite number of restarts, so that it is not a true Las Vegas algorithm. (Indeed, we give a family of permutations that cannot be sorted by this algorithm regardless of the number of restarts.) The other is a deterministic algorithm that always sorts the permutation and runs in  $O(n \log n + kn)$  time, where  $k$  is the number of successive “corrections” (detailed in Section 5) that must be applied—a value, incidentally, that is not related to the edit distance  $d$ , although it is bounded by it. We give a family of permutations for which  $k$  is  $\Theta(n)$  (the worst-case value for  $k$ ) and thus for which our sorting algorithm will run in quadratic time. However, we present the results of very extensive experimentation showing that the expected value and the standard deviation of  $k$  are small constants (less than 1), independent of  $n$ , so that the running time of the algorithm is, with high probability,  $O(n \log n)$ . Thus we conclude (but do not prove) that almost all permutations can be sorted in optimal  $O(n \log n)$  time.

## 2 Preliminaries

A permutation  $\pi$  is written as  $(\pi_1 \pi_2 \dots \pi_n)$ , where each element  $\pi_i$  is a signed integer and the absolute values of these elements are all distinct and form the set  $\{1, 2, \dots, n\}$ . The absolute value of  $\pi_i$  is denoted by  $|\pi_i|$ . An *inversion*  $\rho(i, j)$  on a permutation  $\pi = (\pi_1 \dots \pi_i \dots \pi_j \dots \pi_n)$  reverses all elements between  $\pi_i$  and  $\pi_j$  while changing their signs giving  $(\pi_1 \dots \pi_{i-1} - \pi_j \dots - \pi_i \pi_{j+1} \dots \pi_n)$ . We assume that every permutation of  $n$  elements is framed by elements 0 and  $n+1$ . In this way we consider each permutation to be linear, noting that each linear permutation corresponds to  $n+1$  circular permutations (of length  $n+1$ ), which are equivalent in terms of the sequences of inversions used to sort them.

Two adjacent elements,  $\pi_i$  and  $\pi_{i+1}$  for  $0 \leq i \leq n + 1$ , form an *adjacency*. An adjacency is a *non-breakpoint* if and only if we have  $\pi_{i+1} - \pi_i = 1$ , otherwise it is a *breakpoint*. An *oriented pair*,  $(\pi_i, \pi_j)$ , in a permutation is a pair of integers with opposite signs such that  $\pi_i + \pi_j = \pm 1$ . The inversion induced by an oriented pair  $(\pi_i, \pi_j)$ , called an *oriented inversion*, is  $p(i, j-1)$  for  $\pi_i + \pi_j = +1$ , and  $p(i+1, j)$  for  $\pi_i + \pi_j = -1$ . An oriented inversion always creates a non-breakpoint; we say that it *heals* the breakpoint (or breakpoints—there could be two) to which the elements of the oriented pair belonged before the inversion.

A *framed common interval* (FCI) [2] of a length  $n$  permutation is a substring of the permutation,  $(as_1s_2\dots s_kb)$  or  $(-bs_1s_2\dots s_k-a)$  (with  $s_1s_2\dots s_k$  possibly empty) such that

- for each  $i$ ,  $1 \leq i \leq k$ ,  $|a| < |s_i| < |b|$ ,
- for each  $l$ ,  $|a| < l < |b|$ , there exists a  $j$ ,  $1 \leq j \leq k$ , with  $|s_j| = l$ , and
- the FCI is not a union of shorter intervals with the above properties.

The substring  $s_1s_2\dots s_k$  is thus a (possibly empty) signed permutation of the integers greater than  $a$  and less than  $b$ ; elements  $a$  and  $b$  are called the *frame* elements. The framed interval is said to be common in that it also exists, in its canonical form  $(+a^+(a+1)^+(a+2)\dots^+b)$ , in the identity permutation. FCI  $B$  is *nested* inside FCI  $A$  if and only if the left and right frame elements of  $A$  occur, respectively, before and after the frame elements of  $B$ .

A *component* is comprised of the frame elements from an FCI along with all elements inside the FCI that are not used for a nested subinterval. A *bad component* is a component where all elements have the same sign. Two components can only overlap at the frame elements [3]. An inversion is said to be *unsafe* if it creates a bad component, otherwise it is *safe*. A permutation is *positive* if it is not the identity permutation and every element is positive. A positive permutation indicates the existence of at least one bad component. Any permutation containing bad components can be transformed to another permutation that does not contain any bad component in linear time [1]. Thus, in the algorithms we describe, we assume that the input permutation does not contain any bad components.

### 3 Background: Data Structures for Permutations

To implement an algorithm for sorting by inversions, we need a data structure for handling permutations that supports two basic operations: (i) choose an oriented inversion, and (ii) perform an inversion.

We now describe the data structure of Kaplan and Verbin [7] that stores a permutation in linear space and allows us to perform an inversion in logarithmic time. The structure is a splay tree, in which the nodes are ordered by the indices of the permutation, with one additional flag maintained at each node.

To perform an inversion  $p(i, j)$  between (and including) indices  $i$  and  $j$ , index  $i-1$  is splayed and the right subtree of the root is split from the root yielding subtrees  $T_{<i}$  and  $T_{\geq i}$  where  $T_{<i}$  ( $T_{\geq i}$ ) contains all elements with indices less than (greater than or equal to)  $i$ . Next, index  $j$  is splayed in  $T_{\geq i}$  and again the right subtree is split from its root yielding subtrees  $T_{rev}$  and  $T_{>j}$  where  $T_{>j}$  contains all elements with indices greater than  $j$  and

$T_{rev}$  contains the elements of the permutation that have to be reversed. Finally, there are three subtrees:  $T_{<i}$ ,  $T_{rev}$  and  $T_{>j}$ . Now, actually reversing the elements in  $T_{rev}$  can take  $\Theta(n)$  time since  $\Theta(n)$  elements could be reversed in a single inversion. To achieve logarithmic time complexity a lazy approach is taken: a *reversed* flag is maintained in each node, which if turned on indicates that the subtree rooted at the node is reversed. Now instead of immediately reversing a subtree, we just set its reversed flag. During an inversion the reversed flag of the root of  $T_{rev}$  is flipped and  $T_{<i}$  is joined to  $T_{rev}$  to get  $T_{\leq j}$ . This is achieved by making  $T_{rev}$  the right child of the root of  $T_{<i}$ , which still contains the element at index  $i - 1$ , yielding the tree  $T_{\leq j}$ .  $T_{\leq j}$  is then joined to  $T_{>j}$  by splaying  $j$  in  $T_{\leq j}$ , after which  $T_{>j}$  is made the right child of the root of  $T_{\leq j}$ , yielding the final tree which represents the permutation after the inversion. Since the only operation that takes more than constant time is the splay and since splaying takes amortized logarithmic time [13], each inversion takes amortized logarithmic time.

A tree could have several reversed flags, but the invariant maintained is that an in-order traversal modified by the reversed flags yields the permutation. So to read the permutation one would traverse a reversed subtree in reverse order while flipping signs of elements read. Nested reversed flags cancel in the sense that a reversed flag on a node within a reversed subtree, implies that the inner subtree (rooted at that node) is not reversed. Thus, a subtree rooted at a node is reversed if and only if there is an odd number of reversed flags in the path from the root to the node (including the node).

When a sequence of inversions is performed, reversed flags can get nested to arbitrarily deep levels. We can push the flag down a traversed path in the tree, by flipping the sign of the element in the node, exchanging the left and right subtrees, and flipping the reversed flags in both children. The reversed flag of a leaf is cleared by just flipping its sign. Pushing down a flag takes constant time per node so the logarithmic time complexity of splaying is maintained. By pushing down the flags in the splay path we ensure that the three subtrees created ( $T_{<i}$ ,  $T_{rev}$  and  $T_{>j}$ ) reflect the changes made in all the previous inversions.

This is exactly the data structure described in [7]; it can handle a sequence of  $d$  inversions in  $O(d \log n)$  time. The data structure maintains only the state of the permutation at each step (in a lazy way). However it does not maintain information about oriented pairs, nor could it do so efficiently, as a single inversion could change the orientation of  $\Theta(n)$  pairs. Indeed, using this data structure to maintain the information necessary to choose an oriented inversion at each step would increase the running time by a factor of  $n$ .

To overcome this problem both Kaplan and Verbin [7] and Tannier *et al.* [16] used a two-level version of the data structure in which a permutation is stored in linear blocks of size  $O(\sqrt{n \log n})$  each. Corresponding to each block is a splay tree that maintains information about all oriented pairs  $(\pi_i, \pi_j)$  such that either  $\pi_i$  or  $\pi_j$  is in the block. Performing an inversion while maintaining information about all oriented pairs takes  $O(\sqrt{n \log n})$  time and choosing an inversion at each sorting step takes  $O(\log n)$  time, so that the total time complexity of their algorithms is  $O(n\sqrt{n \log n})$ .

In order to run in  $O(n \log n)$  time, these algorithms need to be able to choose an oriented inversion in logarithmic time and thus information to identify such inversions must also be maintained in logarithmic time through an inversion.

## 4 Our Algorithm

Instead of addressing the data structure (by designing a new data structure that can somehow process  $O(n)$  new pair orientations in logarithmic time), we address the root question of identifying an oriented inversion. Our key contribution is that we need not maintain information about *all* oriented inversions for every permutation at each sorting step—a few suffice in most cases.

### 4.1 MAX Inversions

**Definition 1.** Let  $(\pi_i, \pi_j)$  be an oriented pair in a permutation and let  $\pi_j$  be the negative element in the pair. The oriented inversion corresponding to  $(\pi_i, \pi_j)$  is a MAX inversion if  $\pi_j$  has the maximum value of all negative elements in the permutation. The pair  $(\pi_i, \pi_j)$  is called the MAX pair of the permutation.

For example the MAX inversion in the permutation (4 5 -3 1 -6 2 -7) is  $\rho(4, 6)$ , corresponding to the oriented pair (2, -3), and the MAX inversion in the permutation (2 3 -1 -4) is  $\rho(1, 3)$ , corresponding to the oriented pair (0, -1). We maintain information about only the MAX inversions in the data structure and correspondingly perform a MAX inversion in each sorting step. The result is algorithm MAX.

---

#### Algorithm 1. MAX

---

- 1: **while** there exists a negative element in the permutation **do**
  - 2:   Find index of maximum negative element  $\pi_j$ .
  - 3:   Find index of  $\pi_i = |\pi_j| - 1$ .
  - 4:   Perform inversion corresponding to oriented pair  $(\pi_i, \pi_j)$ .
  - 5: **end while**
- 

Because any permutation that contains a negative element contains a MAX inversion and because any sequence of oriented safe inversions is optimal [6], we can conclude as follows.

**Lemma 1.** *In the absence of unsafe MAX inversions at any sorting step, algorithm MAX produces an optimal sorting sequence.*

Algorithm MAX fails to sort only when it is “stuck” at an all-positive permutation that is not the identity, which happens when a MAX inversion was unsafe. (We deal with unsafe inversions in the next section.) The same arguments hold *mutatis mutandis* if we choose an oriented pair with the minimum negative element, yielding another algorithm, algorithm MIN. Combining the two strategies and picking one at random at each step gives us a randomized algorithm: algorithm RAND.

### 4.2 Maintaining Information through an Inversion

We now show how to maintain information about the maximum negative element of a permutation through an inversion using the splay tree data structure. We describe the process for MAX, but the obvious analog works for MIN.

**Algorithm 2.** RAND

---

```

while there exists a negative element in the permutation do
    randomly select either MAX or MIN
    if MAX then
        Find index of maximum negative element  $\pi_j$ .
        Find index of  $\pi_i = |\pi_j| - 1$ .
        Perform inversion corresponding to oriented pair  $(\pi_i, \pi_j)$ .
    else if MIN then
        Find index of minimum negative element  $\pi_k$ .
        Find index of  $\pi_l = |\pi_k| + 1$ .
        Perform inversion corresponding to oriented pair  $(\pi_k, \pi_l)$ .
    end if
end while

```

---

Let the maximum negative element of a subtree,  $MAX_{neg}$ , be the element in the subtree that has the maximum value among all negative elements in the subtree. The minimum positive element,  $MIN_{pos}$ , of a subtree is defined similarly. These values are stored in each node of the splay tree. Note that the  $MAX_{neg}$  of the root node is the maximum negative element of the permutation, that is, the negative element of the MAX pair of the permutation. The  $MAX_{neg}$  of a node is the maximum of the following three: the  $MAX_{neg}$  of the left subtree, the  $MAX_{neg}$  of the right subtree, and the element in the node if the element is negative. Also notice that whenever the reversed flag of a node is turned on,  $MAX_{neg}$  and  $MIN_{pos}$  are swapped. Therefore pushing down a reversed flag applies this swap to the children, unless there is a cancellation of flags.

A splay operation performs a series of rotations based on the structure of the tree and the index being queried. Each rotation changes at most three edges of a connected subtree while maintaining the binary search tree property.  $MAX_{neg}$  can be recalculated for only the subtree that is affected. Recall that to perform an inversion  $\rho(i, j)$  the splay tree is split into three subtrees which are rejoined after the reversed flag has been set for one of the trees. The value of  $MAX_{neg}$  can be kept for each of the subtrees in the process by simply checking the children of the root after each operation.

By maintaining the  $MAX_{neg}$  values in this fashion, one can maintain the invariant that the  $MAX_{neg}$  of the root node is the maximum negative element of the permutation through any sequence of inversions. Since calculating  $MAX_{neg}$  takes  $O(1)$  time per node, these modifications do not alter the time complexity of the data structure.

**Lemma 2.** *For any (signed) permutation of size  $n$ , there exists a data structure that handles an inversion in  $O(\log n)$  time while maintaining information about the maximum negative element of the permutation.*

### 4.3 Finding the MAX Pair

We now describe how to obtain the elements of the MAX pair in a permutation using the modified data structure described above.

First the maximum negative element of the permutation is located. If the element in a node is not equal to the  $MAX_{neg}$  of the node then  $MAX_{neg}$  of the node lies in either

the left subtree or the right subtree of the node. Therefore starting at the root one can go down the tree looking for the maximum negative element. Reversed flags must be pushed down along the path to ensure that  $MAX_{neg}$  values are updated and the correct path is followed.

To find the second element of the MAX pair, a lookup vector of pointers (of  $n$  elements) maps each element to the node that contains the element. These pointers do not change throughout the computation and enable constant-time lookup of the node containing the second element of the MAX pair.

#### 4.4 Finding the Indices of the MAX Inversion

In absence of reversed flags, the indices of the MAX inversion can be obtained directly from the current location of the nodes corresponding to the MAX pair. However, the presence of a reversed flag indicates nodes that have outdated indices, forcing additional work to retrieve the correct indices.

The index of a node (with respect to the current state of the permutation) can be calculated using the index of the parent node and the sizes of the left and right subtrees. Thus the current index of a node can be calculated whenever the reversed flag is pushed down from it. The size of the subtree rooted at a node is easily maintained. If the node is a right child, then its index is one more than the sum of its parent's index and the size of the left subtree. If the node is a left child, then its index is one less than the difference of its parent's index and the size of the right subtree. The index of the root is just the size of its left subtree. Thus starting at the root, as the reversed flags are pushed down along any path in the tree, the current indices can be calculated.

As one traverses the tree from the root searching for the maximum negative element, the indices are recalculated. After the node corresponding to the second element in the MAX pair is found using the lookup vector, its updated index can be retrieved by traversing up to the root (using parent pointers) and returning down the same path, pushing down the reversed flags and recalculating indices at each node.

#### 4.5 Putting It All Together

The previous subsections detail all the steps for performing a MAX inversion. The time complexity of each of these steps is easy to analyze. Pushing down the reversed flag takes  $O(1)$  time per node. Thus, finding the maximum negative element and its updated index takes  $O(\log n)$  time. Finding the other element of the MAX pair takes  $O(1)$  time and obtaining its updated index takes  $O(\log n)$  time. Therefore the complexity of finding the two indices (steps 2 and 3 in algorithm MAX) is  $O(\log n)$ . For each inversion, maintaining  $MAX_{neg}$ ,  $MIN_{pos}$ ,  $MIN_{neg}$ , and  $MAX_{pos}$  in the nodes takes  $O(1)$  time during split and join operations, and  $O(1)$  time for each rotation in the two splays. Therefore performing the inversion in step 4 of algorithm MAX takes  $O(\log n)$  time. So we have proved:

**Theorem 1.** *For any signed permutation of size  $n$ , a data structure exists that*

- *allows checking whether there exists an oriented inversion in  $O(1)$  time,*

- allows performing a MAX (or MIN) inversion, while maintaining the permutation, in  $O(\log n)$  time,
- and is of size  $O(n)$ .

**Theorem 2.** *In the absence of unsafe inversions at any sorting step, algorithm MAX produces an optimal sorting sequence in  $O(n \log n)$  time.*

## 5 Bypassing Bad Components

We saw that algorithms MAX and RAND can get stuck at a positive permutation by choosing an unsafe inversion. We offer two strategies for recovery.

### 5.1 Randomized Restarts

For algorithm RAND we can simply restart the computation hoping that a better outcome is met in the next run. Indeed, the experiments from Section 6 show that, for most permutations, this simple approach suffices. However, this approach cannot always sort a permutation as there exists a family of permutations that it cannot handle. For instance, take the permutation (3 1-4-2): both MAX and MIN inversions are unsafe because they yield the same positive permutation (3 1 2 4); this small example can be extended to any length by appending the requisite number of positive elements.

### 5.2 Recovering from an Unsafe Inversion: Tannier and Sagot's Approach

Tannier and Sagot [17] introduced a powerful approach for finding and replacing unsafe inversions. They noticed that it is computationally difficult to detect an unsafe inversion as it is used; but it is of course trivial to find out that the process is stuck at a positive permutation. Their approach is thus *postmortem*: their algorithm traces the sorting process back to the most recent unsafe inversion and replaces it with a safe one without invalidating other sorting inversions that have been applied. They used an *overlap graph* to keep track of the remaining breakpoints (and whether or not they are oriented). Using the overlap graph they can find the most recent unsafe inversion, replace it with a safe inversion, and continue sorting without invalidating those sorting inversions that have been applied after the most recent unsafe inversion [17]. However, the process may have to be repeated, as, even after replacing an unsafe inversion with a safe one, their algorithm may again get stuck at a positive permutation.

### 5.3 Recovering from an Unsafe Inversion: Our Approach

We use the same general idea, but do not maintain the full overlap graph, as it is too expensive to maintain. Denote by  $p_1$  the first positive permutation at which the algorithm gets stuck and by  $p_i$  the  $i$ th such positive permutation. Recovering from a positive permutation  $p_i$  involves three steps: finding the most recent unsafe inversion  $\mu_i$ , replacing  $\mu_i$  with a safe inversion, and appending inversions without invalidating those sorting inversions that had been applied after  $\mu_i$ . We describe each of these steps in turn.

**Finding the most recent unsafe inversion:** In the trace-back phase, we undo the inversions that have been done so far in order to find the most recent unsafe inversion  $\mu_i$ . Note that an unsafe inversion is an inversion that, when undone, creates a good component from bad components. Denote by  $\pi \cdot S$  and  $\pi \cdot \rho$  the result of applying the inversions from the sequence of inversions  $S$  and the single inversion  $\rho$  to the permutation  $\pi$ , respectively. Let  $U(\pi)$  be the set of unsafe inversions on a permutation  $\pi$  and let  $B(\pi)$  be the set of bad components in  $\pi \cdot \mu$  for  $\mu \in U(\pi)$ . *Undoing* the inversion  $\rho$  in  $\pi \cdot \rho$  refers to performing  $\rho$  on  $\pi \cdot \rho$  which yields  $\pi$ , and undoing the inversions  $S = \rho_1, \rho_2, \dots, \rho_n$  in  $\pi \cdot S$  refers to performing the inversions of  $S$  in the reverse order which yields  $\pi \cdot S \cdot \rho_n \dots \rho_2 \cdot \rho_1 = \pi$ . The sequence of inversions on input permutation  $\pi^0$  that results in the positive permutation  $p_i$  is denoted by  $S_i$ , so  $p_i = \pi^0 \cdot S_i$ .

*Remark 1.* When undoing inversions from  $S_i$ , the most recent unsafe inversion  $\mu_i$  is the first inversion met that turns an element in  $B(p_i)$  from bad to good.

Finding  $\mu_i$  is not trivial because framed intervals can be nested. For example the positive permutation (2 3 6 7 4 5 8 9 10 1) has two components: the one framed by the implicit frame elements 0 and 11, and the nested component framed by the elements 3 and 8. Undoing the inversion  $\rho(2, 7)$  will leave both bad components intact despite the fact that it occurs within the frame elements of the larger component. Thus, in the trace-back phase,  $\rho(2, 7)$  cannot be an unsafe inversion. However, undoing the inversions  $\mu(5, 7)$  and  $\mu(4, 5)$  will make the inner component good and so these two inversions, had they have been performed, would have been unsafe. The following remark characterizes undoing an unsafe inversion in terms of the components in  $B(p_i)$ .

*Remark 2.* An inversion is the most recent unsafe inversion  $\mu_i$  if and only if it is the most recent inversion to change the indices of a proper nonempty subset of the elements from some component  $b \in B(p_i)$ .

The trace-back algorithm is thus as follows: start undoing the inversion sequence  $S_i$ , checking after each inversion whether there exist components in  $B(p_i)$  with both changed and unchanged indices and stop undoing when an unsafe inversion is found. This can be done by keeping an ancillary splay tree where nodes represent adjacencies in the permutation rather than permutation elements.

If every adjacency in  $p_i$  were a breakpoint, the most recent inversion would be unsafe; the heart of the problem, then, is with non-breakpoints and how they interact with the undoing of unsafe inversions. We present a labeling of the ancillary tree so that the safety check can be carried out by a constant-time comparison on the two adjacencies broken by an inversion. Each adjacency has a label indicating the innermost overlying component along with a label that is set only for non-breakpoints. For a given component, each group of consecutive non-breakpoints (ignoring nested components) gets a unique second label. Thus an inversion displaces only a fraction of the elements of a component if and only if both broken adjacencies are labeled as non-breakpoints with the same component and non-breakpoint labels.

In the example, the permutation (2 3 6 7 4 5 8 9 10 1) has component label X for adjacencies (0,2), (2,3), (8,9), (9,10), (10,1), and (1,11), and component label Y for the others. The non-breakpoint labels are the same for (2,3), (8,9), and (9,10), but different between (6,7) and (4,5). Inversion  $\rho(2, 7)$  acts upon non-breakpoints with the same pair

of labels while inversion  $\mu(5, 7)$  acts upon non-breakpoints with different component labels and  $\mu(4, 5)$  acts upon non-breakpoints with different non-breakpoint labels.

We can list the endpoints of the components of a permutation in linear time [112]. A simple traversal of the permutation, keeping one stack for each label, can perform the node labeling described above. Thus the setup of the ancillary tree can be done in  $O(n)$  time. Let  $S_i^1$  be the sequence of inversions applied before  $\mu_i$  in  $S_i$  and  $S_i^2$  be the sequence of inversions applied after  $\mu_i$  in  $S_i$ . Each safe inversion in  $S_i^2$  that is undone before encountering  $\mu_i$  will cost  $O(\log n)$  time so the total cost for finding a most recent unsafe inversion is  $O(n + |S_i^2| \log n)$ .

### Finding a safe inversion given an unsafe inversion

**Lemma 3.** *Given an unsafe oriented inversion  $\mu_i$  and the bad component  $b$  created by  $\mu_i$ , one can find at least one safe inversion  $v_i$  to replace  $\mu_i$  in  $O(n)$  time.*

*Proof.* A bad component could have been created in one of three ways when  $\mu_i$  was applied. Without loss of generality we ignore the symmetric counterpart to the first case below (both cannot happen at once), leaving us with two cases to consider.

$$- (\pm \pi_0 \dots {}^+ l {}^+ x_1 \dots {}^+ x_s \underbrace{\pm \pi_x \dots {}^- r {}^- x_{k-1} \dots {}^- x_{s+1}}_{\text{where the braced inversion creates the bad component}} {}^+ \pi_{x+1} \dots {}^+ \pi_n)$$

where the braced inversion creates the bad component

$$b = {}^+ l {}^+ x_1 \dots {}^+ x_s {}^+ x_{s+1} \dots {}^+ x_{k-1} {}^+ r, \text{ and}$$

$$- (\pm \pi_0 \dots {}^+ l {}^+ x_1 \dots {}^+ x_l \underbrace{{}^- x_{r-1} \dots {}^- x_{l+1}}_{\text{where the braced inversion creates the bad component}} {}^+ \pi_r \dots {}^+ x_{k-1} {}^+ r \dots {}^+ \pi_n)$$

where the braced inversion creates the bad component

$$b = {}^+ l {}^+ x_1 \dots {}^+ x_l {}^+ x_{l+1} \dots {}^+ x_{r-1} {}^+ x_r \dots {}^+ x_{k-1} {}^+ r.$$

For the first case we examine the substrings  ${}^+ l {}^+ x_1 \dots {}^+ x_s$  and  ${}^- r {}^- x_{k-1} \dots {}^- x_{s+1}$ , one of which must contain at least two elements because any bad component contains at least 4 elements. Since the component  $(l, \dots, r)$  is an FCI, if  $L = {}^+ l {}^+ x_1 \dots {}^+ x_s$  contains two or more elements then there must exist a  $w \geq l$  such that  $w+1$  is not in  $L$ . Likewise, if  $R = {}^- r {}^- x_{k-1} \dots {}^- x_{s+1}$  contains two or more elements then there must exist a  $-v$  with absolute value at most  $r$  such that  $v-1$  is not in  $R$ . Thus,  $(w, -(w+1))$  or  $((v-1), -v)$  define oriented pairs, and consequently, oriented inversions. These oriented inversions must be different from  $\mu$ . After applying  $(w, -(w+1))$  or  $((v-1), -v)$  we are left with some elements from the interval  $[l, r]$  grouped on the left with positive signs and some elements from the interval  $(l, r]$  grouped on the right with negative signs, so that any component created by this inversion must be good, and the inversion must be safe.

For the second case, there exists a smallest  $-x_y$  in the subsequence  ${}^- x_{r-1} \dots {}^- x_{l+1}$  and, consequently,  ${}^+ (x_y - 1)$  in either  ${}^+ l {}^+ x_1 \dots {}^+ x_l$  or  ${}^+ x_r \dots {}^+ x_{k-1} {}^+ r$ . Again, this oriented pair implies an oriented inversion. If this inversion does not create a bad component, we are done. If it does create such a bad component, it produces an instance of the first case. If we have  $-x_y = {}^- x_{l+1}$ , the symmetric argument applies to the largest element.

For the first case we find this inversion by a linear scan of the elements in  $R$  and  $L$ , each time checking whether the index of its potential counterpart in an oriented pair is on  $L$  and  $R$ , respectively. Since a lookup table of indices can be initialized in linear time, the whole process will take linear time. For the second case we do a linear scan of the negative elements within the interval to find the minimum. Thus, the safe inversion  $v_i$  to replace  $\mu_i$  can be found in  $O(n)$  time.

**Appending inversions to the sorting sequence:** After we get the permutation  $q_i = \pi \cdot S_i^1$ , we apply the safe inversion  $v_i$  on  $q_i$ . Note that the inversions in  $S_i^2$  remain valid because  $v_i$  only acts within the elements in  $B(p_i)$ . Now we would like to ensure that the sequence of inversions  $S'_i$  we append after  $v_i$  does not create the same adjacencies as those in  $S_i^2$ . We achieve this by renaming the permutation  $q_i$  in the following way.

By definition,  $q_i \cdot \mu_i$  has at least one bad component created by  $\mu_i$  along with a possibly nonempty set  $G(q_i \cdot \mu_i)$  of good components. The inversions that sort the components of  $G(q_i \cdot \mu_i)$  correspond exactly to the sequence  $S_i^2$ . Thus, our desired sequence  $S'_i$  of inversions should only displace (if at all) such components without affecting their structure.

Say there is a component  $c$  of length  $m$  with left frame element  $l$ . The *canonical form*  $\hat{c}$  of  $c$  is a permutation of length  $m$  with  $\hat{c}[i] = c[i] - l + 1$ ,  $1 \leq i \leq m$ , where  $p[i]$  denotes the  $i$ th element of a permutation  $p$ . Components  $c$  and  $d$  are said to be *structurally equivalent* if and only if we have  $\hat{c} = \hat{d}$ .

**Lemma 4.** *Let  $q_i$  be a permutation without a bad component and  $\mu_i$  be an inversion such that  $q_i \cdot \mu_i$  has at least one bad component and a set of good components  $G(q_i \cdot \mu_i)$ . There exists a  $q'_i$  where any sequence  $S'_i$  that sorts  $q'_i$  to the identity, when applied to  $q_i$ , will result in a permutation whose only components are those in  $G(q_i \cdot \mu_i)$ .*

*Proof.* Rename the permutation  $q_i \cdot \mu_i$  such that all breakpoints from components in  $G(q_i \cdot \mu_i)$  become non-breakpoints and then undo  $\mu_i$  to get  $q'_i$ . Note that this renaming leaves one structurally equivalent bad component in place of each bad component, so that the renaming is unique. A inversion sequence that sorts  $q'_i$  to the identity heals all breakpoints from the bad components in  $q_i \cdot \mu_i$ ; moreover, it does not heal any breakpoint from components of  $q_i$  that are in  $G(q_i \cdot \mu_i)$  due to the nesting property of FCIs.

For example, take  $q_i = (2\ 3\ 6\ 7\ 4\ -8\ -5\ -9\ 10\ -1)$  and  $\mu_i = \mu(6, 7)$ . Now  $q_i \cdot \mu_i$  is  $(2\ 3\ 6\ 7\ 4\ 5\ 8\ -9\ 10\ -1)$ , so that  $G(q_i \cdot \mu_i)$  is comprised of the components framed by the pair (of frame elements)  $(0, 11)$  and the pair  $(8, 10)$ .  $q_i \cdot \mu_i$  is renamed to  $q'_i \cdot \mu_i = (1\ 2\ 5\ 6\ 3\ 4\ 7\ 8\ 9\ 10)$ , yielding  $q'_i = (1\ 2\ 5\ 6\ 3\ -7\ -4\ 8\ 9\ 10)$ . The sorting sequence  $S'_i = (\rho(3, 6), \rho(3, 4), \rho(4, 7))$  for  $q'_i$  can be applied to  $q_i$  to get  $(2\ 3\ 4\ 5\ 6\ 7\ 8\ -9\ 10\ -1)$ .

**Lemma 5.** *Given a permutation  $p$  with a set of bad components  $B(p)$ , permutation  $p'$ , that has one structurally equivalent bad component in place of each  $b \in B(p)$  and only non-breakpoints everywhere else, can be constructed in linear time.*

*Proof.* If an adjacency is not part of a bad component then label it with a null value; otherwise label it by the bad component of which it is part of. Also label adjacencies with the left and right endpoints of each component, which can be done in linear time [10]. We use a stack  $R$ , the top of which we denote by  $t(R)$ . Perform the following steps until the end of the permutation is reached, i.e., until we have  $i = n$ .

1. Label each element  $p'[i]$  with the value  $p'[i - 1] + 1$  until an adjacency corresponding to a bad component is encountered.
2. If the adjacency is a left endpoint, then push onto  $R$  the value  $p[i - 1] - p'[i - 1]$  and go to step 3. If it is a right endpoint, then pop the top element. If the next breakpoint is labeled with a bad component, then go to step 3 otherwise go to step 1.

3. Label each element  $p'[i] = p[i] - t(R)$  until an adjacency with a different component label is reached, then go to step 2.

The renaming procedure takes linear time and works correctly because every bad component is renamed to a structurally equivalent component in step 2.

**Overall running time analysis:** We call this algorithm, with the recovery phase included, MAX-RECOVER or RAND-RECOVER, depending on whether algorithm MAX or algorithm RAND is used in the forward-sorting phase. If algorithm MAX or RAND gets stuck at a positive permutation  $p_i$ , we proceed by undoing inversions until a permutation  $q_i$  is found such that  $q_i \cdot \mu_i$  has fewer bad components than  $q_i$ . Finding such a  $q_i$  and  $\mu_i$  alone takes  $O(n + |S_i^2| \log n)$  time. The inversions undone in this step are not discarded as they can be applied after replacing the unsafe inversion  $\mu_i$  with a sequence of at least two safe inversions. Notice that each inversion undone in the trace-back must be done or undone on a splay tree at most three times and that  $S_i^2$  and  $S_j^2$  for any two  $p_i$  and  $p_j$ ,  $i \neq j$ , must be disjoint. Thus the  $O(n \log n)$  term describes the amount of time spent for undoing inversions over the entire course of the algorithm and just a linear amount of work beyond that must be done in each recovery phase.

**Theorem 3.** *The running time of MAX-RECOVER or RAND-RECOVER is  $O(n \log n + kn)$  where  $k$  is the total number of unsafe inversions performed in the algorithm.*

In Section 6 we show strong empirical evidence that, on random permutations of length  $n$ , the average value and standard deviation of  $k$  remain constant (about  $\frac{1}{2}$ ) even as  $n$  grows very large, leading us to conjecture that these algorithms sort almost all permutations in  $O(n \log n)$  time. In the worst case, however, RAND-RECOVER and MAX-RECOVER can use  $\Theta(n^2)$  time, as in the following family of permutations: build a permutation of length  $n$  by starting with the identity permutation of length  $n \bmod 5$  as the first block, followed by  $n/5$  copies of the block  $i(i+3)(i+1)-(i+4)-(i+2)(i+5)$ , each of which shares its first element with the last element of the preceding block.

## 6 Experimental Results

We present experimental results for algorithms MAX, RAND, MAX-RECOVER and RAND-RECOVER. All of the experiments are on random permutations of length 100, 200, 500, 1000, 2000, 5000, 10,000 and 20,000. For each length, we tested our algorithms on 1,000,000 permutations.

**Table 1.** The failure rates for MAX, RAND and RAND+RESTART

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
MAX	39.5%	38.9%	39.0%	39.1%	39.3%	39.3%	39.3%	39.2%
RAND	39.0%	39.2%	39.5%	39.5%	39.6%	39.5%	39.6%	39.5%
RAND-RESTART	17.2 %	17.1 %	16.8 %	16.4 %	16.3 %	16.2 %	16.0 %	16.0 %

**Table 2.** Number of recovery steps ( $k$ ) for MAX-RECOVER: Average and Standard Deviation

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
AVE( $k$ )	0.513	0.518	0.522	0.524	0.524	0.525	0.524	0.525
SD( $k$ )	0.765	0.770	0.772	0.774	0.773	0.775	0.774	0.777

**Table 3.** Number of recovery steps ( $k$ ) for RAND-RECOVER: Average and Standard Deviation

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
AVE( $k$ )	0.485	0.489	0.492	0.493	0.495	0.495	0.495	0.499
SD( $k$ )	0.690	0.694	0.697	0.697	0.698	0.698	0.698	0.699

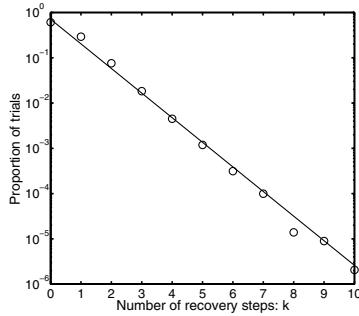
**Fig. 1.** The distribution of  $k$  for MAX-RECOVER on random permutations of length 10,000

Table II lists the failure rates for algorithm MAX and algorithm RAND. Algorithm MAX and algorithm RAND produce an optimal sorting sequence with frequency 61%. We also include the failure rates for RAND-RESTART: the simple heuristic that runs RAND on the input permutation a second time if it fails to sort at the first attempt. The failure rate for RAND-RESTART reduces to 16% ( $\approx 0.39 \times 0.39$ ), which suggests that the two runs are independent with respect to the failure rate.

Tables 2 and 3 summarize the details of the number of recovery steps,  $k$ , that we observe in algorithms MAX-RECOVER and RAND-RECOVER. The average value and the standard deviation of  $k$  remain constant as  $n$  grows. Figure II shows the distribution of  $k$  for MAX-RECOVER on random permutations of length 10,000. This figure is representative of the observed distribution for the other lengths as well. The similarity to the inverse exponential function suggests that the upper bound for the average value of  $k$  is a constant.

## 7 Conclusions

We have given two new algorithms for sorting signed permutations by inversions, one a fast heuristic that works on most permutations, the other a deterministic algorithm that sorts all permutations and takes  $O(n \log n)$  time on almost all of them. We have given the results of very extensive experimentation to confirm these claims. We have thus taken a major step towards a final resolution of the sorting problem. Future work

includes a formal proof that our deterministic algorithm sorts almost all permutations in  $O(n \log n)$  time and designing an algorithm to deal with the few remaining permutations where our algorithm takes more time.

## References

1. Bader, D.A., Moret, B.M.E., Yan, M.: A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.* 8(5), 483–491 (2001)
2. Bergeron, A., Heber, S., Stoye, J.: Common intervals and sorting by reversals: a marriage of necessity. In: Proc. 2nd European Conf. Comput. Biol. ECCB 2002, pp. 54–63 (2002)
3. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. In: Warnow, T.J., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 68–79. Springer, Heidelberg (2003)
4. Caprara, A.: Sorting by reversals is difficult. In: Proc. 1st Int'l Conf. Comput. Mol. Biol. (RECOMB 1997), pp. 75–83 (1997)
5. Day, W.H.E., Sankoff, D.: The computational complexity of inferring phylogenies from chromosome inversion data. *J. Theor. Biol.* 127, 213–218 (1987)
6. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In: Proc. 27th Ann. ACM Symp. Theory of Comput. (STOC 1995), pp. 178–189. ACM Press, New York (1995)
7. Kaplan, H., Verbin, E.: Efficient data structures and a new randomized approach for sorting signed permutations by reversals. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 170–185. Springer, Heidelberg (2003)
8. Moret, B.M.E., Warnow, T.: Advances in phylogeny reconstruction from gene order and content data. In: Zimmer, E.A., Roalson, E.H. (eds.) Molecular Evolution: Producing the Biochemical Data, Part B, Methods in Enzymology, vol. 395, pp. 673–700. Elsevier, Amsterdam (2005)
9. Palmer, J.D.: Chloroplast and mitochondrial genome evolution in land plants. In: Herrmann, R. (ed.) Cell Organelles, pp. 99–133. Springer, Heidelberg (1992)
10. Palmer, J.D., Thompson, W.F.: Rearrangements in the chloroplast genomes of mung bean and pea. *Proc. Nat'l Acad. Sci., USA* 78, 5533–5537 (1981)
11. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Apostolico, A., Galil, Z., Manber, U., Crochemore, M. (eds.) CPM 1992. LNCS, vol. 644, pp. 121–135. Springer, Heidelberg (1992)
12. Sankoff, D., Goldstein, M.: Probabilistic models for genome shuffling. *Bull. Math. Biol.* 51, 117–124 (1989)
13. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. *J. ACM* 32(3), 652–686 (1985)
14. 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)
15. Sturtevant, A.H., Dobzhansky, T.: Inversions in the third chromosome of wild races of *drosophila pseudoobscura* and their use in the study of the history of the species. *Proc. Nat'l Acad. Sci., USA* 22, 448–450 (1936)
16. Tannier, E., Bergeron, A., Sagot, M.-F.: Advances on sorting by reversals. *Disc. Appl. Math.* 155(6–7), 881–888 (2007)
17. Tannier, E., Sagot, M.-F.: Sorting by reversals in subquadratic time. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 1–13. Springer, Heidelberg (2004)

# Finding Biologically Accurate Clusterings in Hierarchical Tree Decompositions Using the Variation of Information

Saket Navlakha<sup>1,2</sup>, James White<sup>2</sup>, Niranjan Nagarajan<sup>1,2</sup>,  
Mihai Pop<sup>1,2</sup>, and Carl Kingsford<sup>1,2</sup>

<sup>1</sup> Department of Computer Science  
`{saket,mpop,carlk}@cs.umd.edu`

<sup>2</sup> Center for Bioinformatics and Computational Biology,  
Institute for Advanced Computer Studies  
University of Maryland, College Park  
`whitej@umd.edu, niranjan@umiacs.umd.edu`

**Abstract.** Hierarchical clustering is a popular method for grouping together similar elements based on a distance measure between them. In many cases, annotations for some elements are known beforehand, which can aid the clustering process. We present a novel approach for decomposing a hierarchical clustering into the clusters that optimally match a set of known annotations, as measured by the variation of information metric. Our approach is general and does not require the user to enter the number of clusters desired. We apply it to two biological domains: finding protein complexes within protein interaction networks and identifying species within metagenomic DNA samples. For these two applications, we test the quality of our clusters by using them to predict complex and species membership, respectively. We find that our approach generally outperforms the commonly used heuristic methods.

**Keywords:** Hierarchical Tree Decompositions, Variation of Information, Clustering, Protein Interaction Networks, Metagenomics, OTUs.

## 1 Introduction

Hierarchical clustering is an important tool in many applications. One application where it has been particularly useful is predicting protein membership in complexes using protein-protein interaction (PPI) networks. High-throughput experimental protocols are producing information on thousands of PPIs [52]. Embedded within these networks are protein complexes, i.e. stable groups of interacting proteins that perform some biological function in the cell. Complex membership is known for some proteins, but even for well-studied species like *S. cerevisiae*, 70-80% of proteins have no complex annotation according to MIPS [19]. Consequently, computational methods for determining to which complexes each protein belongs have recently been developed (e.g. [3, 29, 34, 36]). A common approach to this problem is to identify clusters in the network [2, 5, 33, 34, 35, 48]. Often these clusters are

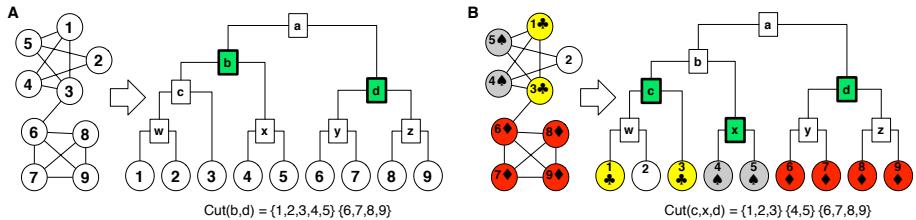
detected by hierarchically clustering the graph [1, 6, 39, 40] based on a topological distance measure such as the Czekanowski-Dice [6] or Jaccard distances. Complex memberships are then transferred to unannotated proteins by considering common known annotations within their clusters [2, 28, 34]. This leads to the following computational problem:

*Problem 1 (Predicting Protein Complexes).* Given a hierarchical clustering of a PPI network for which protein complex annotations are known for some of the proteins, predict complex membership for the unannotated proteins.

A second application of hierarchical clustering is determining bacterial species for uncharacterized DNA sequences obtained from environmental samples [44, 50, 41]. In the expanding field of metagenomics, the composition of microbial communities is examined by sampling DNA from the environment. A typical diversity study involves targeted 16S rRNA gene sequencing using universal primers, a method that has successfully been used to describe bacterial communities in environments ranging from the ocean to soil to the human gut [13, 16, 42]. The standard methodology for 16S sequence analysis begins with a multiple sequence alignment containing both the environmental samples and several sequences of known origin. An evolutionary distance is computed between every pair of sequences using a distance measure such as Jukes-Cantor [23], Kimura 2-parameter [27], or Felsenstein-84 [15]. A hierarchical clustering is then created from these distances, which is analyzed to identify which operational taxonomic units (OTUs; the more precise analog of “species” in the bacterial world) are in the sample. Thus, the approach to this problem is similar to that for complex prediction from PPI networks: uncharacterized sequences are clustered (along with some sequences from known species), and are then assigned to species based on annotated sequences in the same cluster. By estimating the composition of a microbial community, comparisons can be made of the wealth of organisms present in different environments, leading to estimations of the overall diversity. The accuracy of this analysis is vital for researchers examining environments with unknown composition. This leads to the following computational problem:

*Problem 2 (Predicting Species for Uncharacterized DNA).* Given a hierarchical clustering of DNA sequences, some of which are derived from known species, predict the species to which the uncharacterized sequences belong and estimate the number of OTUs in the sample.

In this paper, we give improved methods for applying hierarchical clustering to both of these applications. In general, hierarchical clustering algorithms are based on one or two types of operations: top-down splitting or bottom-up merging. In the network clustering setting, for example, clusters may be split based on network modularity [35] or minimum cuts [11]. Clusters to merge may be chosen based on distances such as the Dice coefficient [6], the Jaccard index [21], or correlation of shortest-path profiles [39], among others. The clustering process produces a tree ranging from the root (all nodes in one cluster) to the leaves (the nodes being clustered, each in its own cluster).



**Fig. 1.** Example PPI network where use of known annotations can produce a better clustering. (A) The network consists of two dense subgraphs that in most approaches would result in the hierarchical decomposition shown. By looking at the topology of the graph, it is reasonable to place proteins  $\{1, 2, 3, 4, 5\}$  into one cluster and proteins  $\{6, 7, 8, 9\}$  into a separate cluster by choosing cut  $\{b, d\}$ . (B) If some annotations are known (indicated in the figure by ♦, ♦, ♦), we want to choose a cut that not only abides by the topology, but also matches the known annotations as closely as possible. Here, cut  $\{b, d\}$  is not ideal because it places proteins  $\{1, 3\}$  and  $\{4, 5\}$  together, which have different known annotations (node 2 has no known annotation). The better cut is  $\{c, x, d\}$ , which induces clusters  $\{1, 2, 3\}$ ,  $\{4, 5\}$ , and  $\{6, 7, 8, 9\}$ . A method that only considers topology will be unable to reconstruct this clustering.

In order to apply most methods for predicting new annotations (either a complex for a protein or a species for a sequence), the hierarchical clustering must be converted into a flat grouping of the elements. Typically, this is done by choosing a set of nodes in the tree (called a *node-cut*) such that the path from each leaf to the root of the tree passes through exactly one chosen tree node. Each chosen tree node yields a cluster consisting of all the leaves in the subtree rooted at that node. We refer to such a flat, non-overlapping grouping of elements simply as a *clustering*. To avoid confusion, we refer to hierarchical clusterings as “hierarchical decompositions.” Some hierarchical decomposition algorithms provide a natural stopping point that can be used to choose a clustering. Newman’s spectral partitioning [35], for example, is a top-down approach for hierarchically decomposing nodes in a network that stops splitting clusters when any split would decrease the modularity of the clustering. Graph summarization [33], a bottom-up approach, stops merging clusters when a particular cost function is minimized. However, many algorithms do not have natural stopping points [1, 11, 24]. Instead, they require the user to estimate the number of clusters beforehand, or they require a threshold and stop when no split or merge satisfying the threshold can be found. In general, it is not clear how to choose the number of clusters or an appropriate distance threshold. Therefore, choosing an appropriate clustering implied by the hierarchy is generally a stumbling block. In many applications annotations are known for some of the elements being clustered, and these partial annotations can help determine which clustering compatible with the hierarchical decomposition is the most biologically reasonable. For example, Figure 1 shows a small PPI network and its natural hierarchical decomposition. The network topology alone suggests a different clustering than the one that makes the most sense when the known annotations are taken into account.

**Our contributions.** In this paper, we propose a novel method, VI-Cut, to choose a clustering from a hierarchical tree decomposition based on how well the clusters induced by a cut in the tree match known annotations, as measured by the variation of information (VI, [31]) metric. The cut is chosen such that each node is placed in a cluster. Hence, nodes with unknown annotations can be placed together in a cluster with nodes with known annotations. We can thus test the quality of a clustering based on how well we can use each cluster to predict annotations for nodes with unknown annotations (e.g. node 2 in Figure 1B). The VI-Cut method is a very natural approach which gives a principled, mathematically sound way to convert a hierarchical decomposition to a flat clustering. To prove its generality, we show that it can be successfully applied in two very different biological problems, and we expect the approach will be applicable to other domains besides the two considered here.

**Improvement in predicting protein complexes.** We apply our VI-Cut method to two different hierarchical decompositions of a PPI network for the yeast *S. cerevisiae*. The first hierarchical decomposition is created using the Czekanowski-Dice distance between network nodes and applying a neighbor-joining algorithm, following the approach of Brun et al. [6]. The second hierarchical decomposition is created using graph summarization [33], which was recently shown [34] to outperform other graph clustering approaches such as MCL [48], MCODE [2], and Newman's spectral partitioning [35] at the task of predicting membership in protein complexes. For both types of hierarchical decompositions, we compare against the methods proposed by Brun et al. [6], Dotan-Cohen et al. [12] and also against an approach that chooses statistically enriched clusters. We also compare against the clustering induced by the natural stopping point of the graph summarization algorithm. Unlike any other method, the VI-Cut produces clusters which perform well in terms of accuracy and coverage of predicted annotations on both trees.

**Improvement in predicting species.** We also applied VI-Cut to predict species annotations for a simulated metagenomic sample created from 1677 real 16S rRNA gene sequences. The sample contains 49 species in various proportions. DOTUR [41] is the most common software for dividing input sequences into OTUs. DOTUR takes as input a distance matrix (derived from a multiple sequence alignment and distance correction) and a distance threshold that defines when to stop merging clusters. We replicated six different methodologies for creating input to DOTUR that have been used in recent 16S rRNA studies [16,42,25,9,44,50]. Each methodology uses a different multiple sequence alignment algorithm, distance correction, and distance threshold. None of these methods, however, take known OTU annotations into account. We test the quality of the VI-Cut clusters and the clusters produced by each of these six methodologies by using them to predict OTUs. In each case, the clusters created by VI-Cut produce predictions with about the same accuracy as the previous methodologies, but with large increases in coverage. Further, the VI-Cut clusters provide a much better estimate of the true number of species embedded within the data set.

### 1.1 Related Work: Semi-supervised Clustering

Several previous attempts have been made to apply semi-supervised clustering to gene expression data. To produce a flat clustering from a hierarchical tree decomposition derived from expression data, several methods assign an *enrichment* score to each internal tree node based on the partial, known annotations, signifying the functional coherence of the cluster [7, 45, 47]. Clusters are then chosen by iteratively choosing high-scoring subtrees, subtrees with uniquely enriched annotations, or other similar heuristics. Recently, Dotan-Cohen et al. [12] proposed a semi-supervised approach based on choosing a subset of edges in the tree decomposition. Each chosen edge induces a connected component in the tree which corresponds to a cluster. Their goal is to choose the minimum number of edges such that each cluster consists of genes which all share at least one annotation, allowing genes that are unannotated to take on any annotation.

All of the above approaches differ from VI-Cut in the objective function used to produce a clustering from the tree and are only applied to clusterings derived from gene expression. No previous studies have predicted OTU annotations using a semi-supervised approach. Brun et al. [6] use PPI network data to build a hierarchical tree decomposition and extract clusters which have a majority annotation (computed using the known annotations). Other heuristics have been proposed to choose a clustering from a network decomposition [39, 11, 40], however, they either rely on manual inspection of the hierarchical decomposition [39], or require a similarity threshold to be input by the user [1, 40, 4].

## 2 Methods

### 2.1 Finding the Clustering That Best Matches Known Annotations (VI-Cut)

**Criteria for choosing a clustering.** A hierarchical decomposition is specified by a tree  $T$  where the leaves correspond to the elements being clustered. A *node-cut* is a subset  $K$  of tree nodes such that the path from every leaf of  $T$  to  $\text{Root}(T)$  passes through some node in  $K$  and such that there is no pair of nodes  $x, y \in K$  where  $x$  is an ancestor of  $y$ . Every node-cut  $K$  of the tree induces a clustering  $C_K$ : each node  $x \in K$  yields one cluster that contains the elements corresponding to the leaves in the subtree rooted at  $x$ . Despite the simple structure, there are an enormous number of possible node-cuts even for short, binary trees. A complete binary tree of height 7, for example, induces exactly 44,127,887,745,906,175,987,802 (i.e.  $4 \times 10^{22}$ ) possible clusterings.

We assume that some (but not all) of the elements that we are interested in clustering are already annotated. Let  $D$  be the partial clustering defined by these known annotations by grouping those with the same annotation together. Among all the possible choices for a node-cut  $K$ , we desire the one that induces a clustering  $C_K$  that best matches the known partial information  $D$ . A natural measure for how well  $C_K$  agrees with  $D$  is given by the variation of information

(VI, [31]) distance metric between the two clusterings:

$$VI(C_K, D) \doteq H(C_K) + H(D) - 2I(C_K, D). \quad (1)$$

Other methods have been used to measure the distance between clusterings, including pair-counting methods, such as the Rand [38], Mirkin [32], and Jaccard [21] indices. VI is attractive because it is a metric, information-theoretic, and, crucially, can be rewritten such that the total distance between clusterings is the sum of each cluster's contribution. Drawbacks associated with other measures are discussed by Meila [31].

In the definition of VI, the clusterings  $C_K$  and  $D$  are represented as discrete random variables taking on  $|C_K|$  and  $|D|$  values, respectively (one value for each cluster in the clustering). Each value corresponds to the probability that a random element chosen belongs to that cluster. This probability is computed by dividing the number of elements in the cluster by the total number of elements. In both clusterings, we ignore unannotated proteins.  $H(X)$  denotes the entropy of random variable  $X$ . Intuitively, the entropy of a clustering tells us how uncertain we are about which cluster a randomly chosen element lies in.  $I(X, Y)$  denotes the mutual information between the random variables  $X$  and  $Y$ . Intuitively, the mutual information gives the reduction in uncertainty regarding the cluster assignment of an element in  $D$  if its assignment in  $C_K$  is given, summed over all elements. In the following, we exploit the decomposability property of VI. Other properties of VI are explored by Meila [31].

Because  $I(X, Y) = H(X) + H(Y) - H(X, Y)$ , where  $H(X, Y)$  is the joint entropy, we can rewrite  $VI(C_K, D)$  to be  $2H(C_K, D) - H(C_K) - H(D)$ . Over possible choices of  $K$ ,  $H(D)$  remains constant. Therefore,  $\min_K VI(C_K, D)$  is achieved for the same  $K$  that minimizes

$$\min_K 2H(C_K, D) - H(C_K). \quad (2)$$

To find a node-cut that minimizes this value, we assign a *quality score*  $q(x)$  to each node  $x$  in the hierarchical decomposition  $T$ . The function  $q(x)$  will be chosen so that the sum of the quality scores for nodes in a node-cut  $K$  will equal  $2H(C_K, D) - H(C_K)$ . Define  $L(x)$  to be the set of leaves in the subtree rooted at node  $x$  that are annotated with some known annotation, and  $A(d)$  to be the set of leaves (from the whole tree) that are known to have annotation  $d$ . Define  $n = |L(\text{Root}(T))|$ , the number of elements that have a known annotation. We then set  $q(x)$  to be

$$q(x) \doteq p(x) \log p(x) - 2 \sum_{d \in D} p(x, d) \log p(x, d), \quad (3)$$

where the probabilities are defined as

$$p(x) = |L(x)|/n, \quad (4)$$

$$p(x, d) = |L(x) \cap A(d)|/n. \quad (5)$$

The value  $p(x)$  is the probability that an element with a known annotation would fall into the cluster induced by  $x$ . The joint probability  $p(x, d)$  is the probability that a random annotated element falls into cluster  $x$  and has annotation  $d$ . Note that  $H(C_K) = -\sum_{x \in C_K} p(x) \log p(x)$  and  $H(C_K, D) = -\sum_{x,d} p(x, d) \log p(x, d)$  ( $x \in C_K, d \in D$ ) so that (3) implies that  $\sum_{x \in K} q(x) = 2H(C_K, D) - H(C_K)$ , which is the value we are attempting to minimize in (2). Therefore, the node-cut whose quality scores sum to the smallest number corresponds to the clustering that best matches the known annotations according to the VI distance.

**Algorithm to find the best cut in a hierarchical tree decomposition.** We can find a node-cut  $K$  in a tree so that  $\sum_{x \in K} q(x)$  is minimized (a “min-node-cut”) using dynamic programming. Let  $\text{Children}(x)$  denote the children of a tree node  $x$ . We can compute the minimum-weight node-cut recursively:

$$\text{CutDist}(x) = \min \begin{cases} q(x) & \text{case I (default if } x \text{ is a leaf)} \\ \sum_{y \in \text{Children}(x)} \text{CutDist}(y) & \text{case II} \end{cases} \quad (6)$$

The min-node-cut of a subtree  $S$  either chooses the root  $x$  of  $S$  with a weight of  $q(x)$  (case I) or it does not choose the root and chooses instead the min-node-cut of each of the subtrees rooted at the children of  $x$  (case II). If  $x$  is a leaf node, the min-node-cut defaults to  $q(x)$ . Therefore, the value of  $\text{CutDist}(\text{Root}(T))$  is weight of the smallest weight node-cut. To find the actual choice of nodes corresponding to the node-cut of this weight we can backtrack through which cases occurred during the recursive calls. We have flexibility in how we break ties when the value of case I equals the value of case II. If we always break ties in favor of case I, we will choose the highest min-node-cut in the tree. Alternatively, if we always choose case II, we choose the lowest min-node-cut in the tree. This algorithm does not require the user to enter the number of clusters to return.

## 2.2 Handling Multiple Annotations on Some Elements

Up to this point, we have assumed that each element has at most one known annotation. This is true by definition in the OTU clustering problem and, of all yeast proteins annotated with some MIPS complex, only 11% are annotated with more than one complex. Hence, for the applications we consider in this paper, the assumption of a single annotation on each element is mostly justified. On the other hand, multiple annotations are present in other applications. They can be used to model either uncertainty in the truth or genuine membership in multiple clusters. A natural way to handle multiple annotations on each element is to look for the node-cut  $K$  that induces a clustering  $C_K$  that minimizes the VI distance between  $C_K$  and the closest clustering compatible with a choice of a single annotation for each element. Unfortunately, even computing the minimum distance between a given clustering  $C$  and a clustering compatible with a set of annotations is NP-complete.

**Definition 1 (annotation collection).** Given a set of elements  $E$  and a set of annotations  $L$ , an annotation collection is a collection of subsets  $A_\ell \subseteq E$  for each  $\ell \in L$  such that every  $e \in E$  is in at least one  $A_\ell$ .

An annotation collection defines which annotations apply to each of the elements of  $E$ . Each  $A_\ell$  consists of the elements that are annotated with  $\ell$ . An annotation collection implicitly specifies many possible clusterings for  $E$ : a choice of a single annotation  $\ell(e)$  for every  $e \in E$  such that  $e \in A_{\ell(e)}$  induces a clustering that groups all elements with the same annotation together. Let  $\text{Compatible}(\mathcal{L})$  be the set of clusterings induced in this way by an annotation collection  $\mathcal{L}$ . The natural measure of how well a given clustering  $C$  matches an annotation collection  $\mathcal{L}$  is to compute the minimum VI distance between  $C$  and some clustering in  $\text{Compatible}(\mathcal{L})$ . Formally, we define:

*Problem 3 (MIN-VI ANNOTATION CHOICE).* Given a set of elements  $E$ , a clustering  $C$  of  $E$ , an annotation collection  $\{A_\ell \subseteq E : \ell \in L\}$  over a set of annotations  $L$ , compute  $\min_{D \in \text{Compatible}(\mathcal{L})} VI(C, D)$ .

**Theorem 1.** The decision version of MIN-VI ANNOTATION CHOICE is NP-complete.

*Proof.* We reduce from EXACT COVER BY 3-SETS (X3C) [17]. Let  $I$  be an instance of X3C specified by a set  $X_I$  and a collection of 3-tuples  $R_I = \{(x, y, z) : x, y, z \in X_I\}$ . An  $I$  is a “yes” instance if there is a subcollection  $M$  of  $R_I$  such that every element in  $X_I$  belongs to exactly one set in  $M$ . We construct an instance of MIN-VI ANNOTATION CHOICE as follows. Take  $E = X_I$ , and let  $C = \{E\}$  be the clustering consisting of a single cluster. For every  $(x, y, z) \in R_I$ , we create an annotation  $A_\ell = \{x, y, z\}$  containing only those 3 elements. The annotation collection  $\mathcal{L}_I$  consists of these  $A_\ell$  sets. We show that there is a clustering  $D \in \text{Compatible}(\mathcal{L}_I)$  with  $VI(C, D) \leq \log(|E|/3)$  if and only if  $I$  belongs to X3C. Because  $C = \{E\}$ , we have  $H(C) = 0$ , and  $VI(C, D) = 2H(C, D) - H(C) - H(D) = H(D)$ . If there is an exact cover  $D$ , it consists of a set of  $|E|/3$  clusters of size 3, yielding  $H(D) = -(|E|/3)[(3/|E|)\log(3/|E|)] = \log(|E|/3)$ . If there is no exact cover, then any clustering  $D$  induced by  $\mathcal{L}$  must contain some clusters of size  $\leq 2$ . Because  $-(3/n)\log(3/n) < -(2/n)\log(2/n) - (1/n)\log(1/n) < -(3/n)\log(1/n)$  for all  $n$ , the presence of clusters of size 2 or 1 yields a larger entropy than grouping those elements into sets of size 3. Hence, if there is no exact cover,  $H(D) > \log(|E|/3)$  for all  $D$  induced by  $\mathcal{L}$ . In fact, it can be shown that the difference between the minimum VI distance for an instance with an exact cover and an instance without an exact cover is at least  $1/|X_I|$ , so this difference can be encoded using a polynomial number of bits.  $\square$

*Problem 4 (MIN-VI TREE CUT WITH ANNOTATION CHOICE).* Given a set of elements  $E$ , a hierarchical decomposition  $T$  of  $E$ , and an annotation collection  $\mathcal{L} = \{A_\ell \subseteq E : \ell \in L\}$  over a set of annotations  $L$ , compute  $\min_{C_K, D} VI(C_K, D)$ , where  $K \in \text{Cut}(T)$  and  $D \in \text{Compatible}(\mathcal{L})$ .

**Theorem 2.** The decision version of MIN-VI TREE CUT WITH ANNOTATION CHOICE is NP-complete.

*Proof.* As above, we reduce from EXACT COVER BY 3-SETS (X3C) [17] (using the same notation). We construct an instance of MIN-VI TREE CUT WITH ANNOTATION CHOICE as follows. Take  $E = X_I \cup Y$  where  $Y$  is a set of new elements such that  $|Y| = 2|X_I|$  and let the hierarchical decomposition  $T$  have a star topology (all leaves connected to the root) with the elements of  $E$  as leaves. For every  $(x, y, z) \in R_I$ , we create an annotation  $A_\ell = \{x, y, z\}$  containing only those 3 elements. The annotation collection  $\mathcal{L}_I$  consists of these  $A_\ell$  sets and  $Y$ . We show that there is a clustering  $D \in \text{Compatible}(\mathcal{L}_I)$  and node-cut  $K$  for  $T$  which induces a clustering  $C_K$ , with  $VI(C_K, D) \leq 1/3 \log(|E|/3) + 2/3 \log 3/2$  if and only if  $I$  belongs to X3C. It is easy to verify that if there is an exact cover  $D'$  then with  $D = D' \cup \{Y\}$  and  $C_K = \{E\}$  we get  $VI(C_K, D) = 1/3 \log(|E|/3) + 2/3 \log 3/2$ . Conversely, if there is no exact cover, then any clustering  $D$  induced by  $\mathcal{L}$  must contain some clusters of size  $\leq 2$ . Using a similar argument as before, we can show that  $VI(D \cup \{Y\}, \{E\}) > 1/3 \log(|E|/3) + 2/3 \log 3/2$ . The only other node-cut possible is the one which puts every node in  $E$  in a separate cluster and the corresponding optimal annotation choice gives a VI distance  $\geq 2/3 \log |E| - 2/3 \log 3/2 > 1/3 \log(|E|/3) + 2/3 \log 3/2$  (in the ideal case every element in  $R_I$  will have its own annotation), for  $|E| > 2$ . Note that the difference between the minimum VI distance for an instance with an exact cover and an instance without an exact cover is still  $\geq 1/|X_I|$  and hence can be encoded using a polynomial number of bits.  $\square$

Given these hardness results, we are forced to consider heuristics to handle the few proteins that belong to multiple MIPS complexes. We cannot use equation (5) directly to compute  $p(x, d)$  because it will not yield a probability distribution. Instead, if protein  $i$  has  $k_i$  annotations, we count each of its annotations as  $1/k_i$ . In other words,  $p(x, d) = (1/n) \sum_{i \in L(x) \cap A(d)} 1/k_i$ . This way  $p(x, d)$  defines a probability distribution even if proteins belong to multiple complexes, and we can use the method of the previous section as a heuristic to find a clustering that matches the given annotations well. This is the approach we follow for the complex membership prediction experiments below.

### 2.3 Predicting New Annotations

We can test the quality of our clusters by using them to make new predictions for protein or sequence membership within complexes or OTUs. A common approach, here called “majority,” transfers an annotation  $A$  to every unannotated element in a cluster if more than 50% of the annotated elements in the cluster are annotated with  $A$ . If no annotation exists on more than 50% of the annotated elements, no predictions are made. Clusters consisting of a single annotated element are ignored.

To test the efficacy of the various clustering methods, we omit the known annotations from a fraction of the elements. The omitted annotations are the “test set,” and the remaining annotations are the “training set.” Each method finds its clusters based only on the annotations in the training set. We vary the size of the training set from 10% to 90% of the total number of elements with known annotations, chosen randomly. For each element  $x$  in the test set, the majority annotation

is computed and then transferred to  $x$  as a predicted annotation. If multiple annotations are transferred, each transferred annotation is counted as one prediction. A prediction is correct if the protein or sequence is known to belong to that complex or OTU, and incorrect if it is only known to belong to other complexes or OTUs. Naturally, given the incomplete state of knowledge, some “incorrect” predictions may in fact be correct. For each size of the training set, we measure performance by the accuracy and coverage of the predictions made over 500 random samplings. (For the Snip approach we only took 10 samplings). Accuracy is the probability that a predicted annotation is correct. Coverage is the average number of elements in the test set for which a correct annotation was made divided by the total number of elements in the test set.

## 2.4 Application to Predicting Protein Complex Annotations

**Protein networks.** We constructed a protein interaction network for *S. cerevisiae* using all edges in the IntAct [26] database. This network contains 5,492 proteins with 40,332 interactions. For the hierarchical decomposition, we consider only the largest connected component of the network (which we refer to as  $Y_{\text{ppi}}$ ), which contains 5,462 proteins and 40,311 interactions. Most of these interactions were determined using yeast two-hybrid or TAP assays, while a smaller number were derived from traditional, low-throughput experiments. Interactions obtained from high-throughput assays, however, are typically very noisy with potentially a 90% false positive rate [20]. Hence, we created a high-confidence yeast interaction network from IntAct that only includes edges supported by at least two experiments. The high-confidence network contains 2,604 proteins and 8,341 interactions, fewer than half the proteins of the  $Y_{\text{ppi}}$  network. Its largest connected component, which we call  $Y_{\text{high-conf}}$ , contains 2,378 proteins and 8,189 interactions.

**Protein complexes.** Annotations for yeast complexes are from MIPS [19], ignoring the “550” section of the catalog, which represent computationally inferred complexes. This set of complexes has been widely used to assess computational methods [22, 51, 37]. To make the most specific predictions possible we use the lowest-level complexes in the catalog. Of the 5,462 and 2,378 proteins in  $Y_{\text{ppi}}$  and  $Y_{\text{high-conf}}$ , 1191 and 930 proteins, respectively, have some known complex annotation. Of the 267 complexes, 266 and 230 are represented by at least one protein in the  $Y_{\text{ppi}}$  and  $Y_{\text{high-conf}}$  network, respectively. The average number of proteins per complex in  $Y_{\text{ppi}}$  is 5.2 (min = 1, max = 78), and in  $Y_{\text{high-conf}}$  is 4.7 (min = 1, max = 67).

**Hierarchical decomposition of the PPI network.** We use two approaches to generate two different hierarchical tree decompositions of a PPI network. The first tree, called  $T_{\text{Dice}}$ , is built by applying the neighbor-joining algorithm BIONJ [18] to distances between proteins computed by the Czekanowski-Dice [6] distance. Self-loops were added to each protein to decrease the distance between proteins that interact. This is the approach followed by Brun et al. [6] for

predicting the cellular function of proteins. The second tree, called  $T_{GS}$ , is built using the greedy graph summarization algorithm (GS, [33,34]). The GS process has a natural stopping point (when there is no longer any compression benefit to merging two nodes). We modified the algorithm so that it continues to merge the pair of nodes that give the least negative benefit until all nodes are placed in a single cluster.

**Comparison methods.** For the  $T_{Dice}$  tree, we compare the VI-Cut approach against three other methods. Brun et al. [6] filter false edges from their PPI network by removing proteins which take part in fewer than 3 interactions. In our setting, we simply use the high-confidence network,  $Y_{\text{high-conf}}$ . Brun et al. [6] extract clusters from their hierarchical network decomposition by selecting the largest subtrees that contain at least 3 proteins that all share the same annotation and that make up the majority annotation in the subtree. Dotan-Cohen et al. [12] choose the minimum number of edges in the tree to “snip” such that each cluster induced by the snip contains proteins that all share at least one annotation. Another popular approach involves using the hypergeometric P-value to assign an enrichment score to each internal node in the tree. We then do a breadth-first walk down the tree from the root, choosing clusters if they are enriched past a pre-defined threshold ( $P \leq 0.01$ ). The computed P-values are Bonferroni corrected to account for multiple-testing. We refer to these methods by Brun, Snip, and Enrich, respectively. For Brun and Enrich, if a protein is not assigned to any chosen subtree, it is placed in a cluster by itself. When considering  $T_{GS}$ , we also compare with the clustering induced by the natural stopping point of the unmodified greedy GS process. For the VI-Cut on both trees we select the lowest min-node-cut.

## 2.5 Application to Predicting Operational Taxonomic Unit (OTUs)

**Creation of simulated 16S sample.** We obtained 1860 partial 16S rRNA gene sequences from the Ribosomal Database Project II (release 9.57 [8]) with complete taxonomic identification. These sequences were then screened for conflicting annotation information using the RDP Bayesian classifier [49], and selected for length and quality, resulting in a final set of 1677 sequences. This dataset is designed to simulate a microbial environment of moderate complexity spanning seven phyla with several dominant and rare species. Nine species are only observed once in the data, while eight species have more than 90 observations. Though no single species represents more than 6% of the sample, 66% of the sample is Proteobacteria with roughly equally distributions of Alpha-, Beta-, and Gammaproteobacteria. The other 34% of the sample comes from the following six phyla: Actinobacteria, Bacteroidetes, Chlamydiae, Fibrobacteres, Firmicutes, and Spirochaetes. By using real 16S rRNA sequences, we accurately model the nucleotide divergence we expect to see within any species. This approach has been successfully used to provide high-quality benchmarks for metagenomic assembly and gene-finding [30].

**Hierarchical decomposition of OTU sequences.** Sequences were oriented and subsequently aligned using a multiple-sequence alignment (MSA) algorithm (such as ClustalW [46], NAST [10], or MUSCLE [14]). MSAs were trimmed so that each sequence spanned the entire alignment. From the alignment, we then used DNADIST with default parameters from the PHYLIP package [15] to compute distance matrices using the Felsenstein-84 [15] or Jukes-Cantor [23] distances. The distance matrices were then fed into DOTUR [41], an OTU clustering algorithm, which assigns sequences to OTUs using the furthest-neighbor algorithm. The clusters returned by DOTUR depend on a user-defined distance threshold. If the threshold is set to 0.03, for example, an OTU cluster is defined as a set of sequences which are each no more than 3% different from each other. We modified DOTUR to output the full hierarchical tree decomposition, which we use to find the VI-Cut clusters or OTUs based on partial, known annotations.

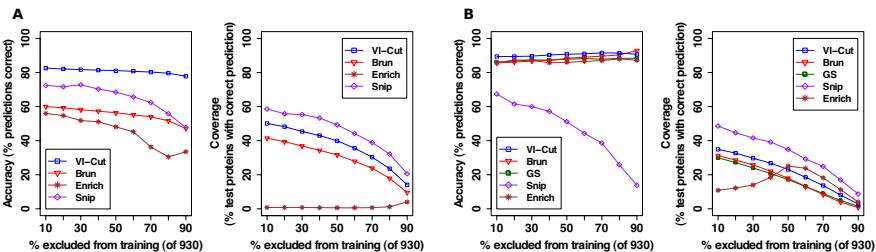
**Comparison methods.** We consider six recently published methods for identifying OTUs that illustrate the current range of OTU-analysis used in the field of metagenomics. These methods differ in the MSA, distance correction, and distance threshold used to define OTUs. The six methods we consider are: Kennedy et al. [25], Fulthorpe et al. [16], Schloss et al. [42], Corby-Harris et al. [9], Sogin et al. [44], and Warnecke et al. [50]. We refer to each by their first author. See Table 1 for their parameters. The Corby-Harris approach yielded nearly identical results as the Kennedy method, and is therefore omitted from the table. We compare the VI-Cut clusters, obtained using the highest min-node-cut, with the threshold-derived clusters of these six methodologies based on their predictive ability and estimation of the number of OTUs present in the sample.

## 3 Results and Discussion

### 3.1 VI-Cut Yields Better Predictions for Protein Complexes

We created a hierarchical decomposition  $T_{\text{Dice}}$  based on the Czekanowski-Dice distance between proteins in  $Y_{\text{high-conf}}$ , following the same procedure described by Brun et al. [6] (see Section 2.4). From  $T_{\text{Dice}}$ , for various sizes of training sets, we compute four clusterings derived from the methods of Brun et al, Dotan-Cohen et al., the Enrich approach described in Section 2.4, and the VI-Cut approach described in Section 2.1. Using these clusterings, we predict membership in MIPS protein complexes using the “majority” annotation transfer rule. The accuracy and coverage of these predictions are shown in Figure 2A. The x-axis of these plots gives the percentage of annotations that were excluded from the annotation set when choosing a clustering; larger values indicate tests where there are fewer known annotations. The y-axis shows the accuracy and coverage of the predictions — in both cases, larger numbers are preferred.

While both Brun et al. and VI-Cut take into account the known annotations when defining their clusters from the tree, the cut chosen by minimizing the VI distance is able to make considerably more accurate predictions than the clusters created by the Brun et al. heuristic. Over all tested sizes of training

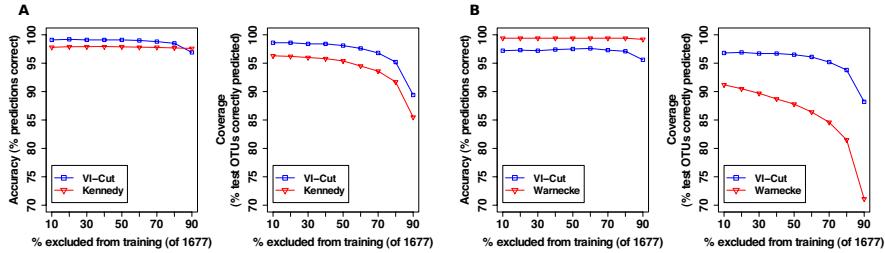


**Fig. 2.** Accuracy and coverage for protein complex predictions for various sizes of training sets on (A) the  $T_{\text{Dice}}$  tree, and (B) the  $T_{\text{GS}}$  tree

set, the predictions made by the VI-Cut approach are more accurate by at least 22 percentage points. Further, when the number of known annotations is very small, the improvement of the VI-Cut method is even greater. At the fewest number of known annotations (90% annotations excluded) the VI-Cut method is almost 30% more accurate in its predictions. The VI-Cut method also makes more correct annotations (larger coverage) over the entire range of sizes for the training set. The Enrich approach is even less accurate than Brun et al. and with a significantly lower coverage. This is largely because the enrichment approach returns a few number of large modules for which very few predictions can be made. The Snip approach yields a higher coverage than VI-Cut but with a greater loss in accuracy.

The robustness of the VI-Cut approach is not limited to hierarchical decompositions that are derived from the Czekanowski-Dice distance. We repeated the prediction experiments using the tree  $T_{\text{GS}}$  built by the greedy graph summarization (GS) technique. Figure 2B shows the accuracy and coverage achieved by the four previously mentioned methods, and the clustering induced by the natural stopping point of the GS procedure. The clusters produced by the natural stopping point are the same regardless of the training set because annotations are not considered when the GS algorithm is applied. Accuracy and coverage can still vary, however, as predictions in majority annotations change within each cluster. As shown in Figure 2B, the predictions made by the VI-Cut are almost always more accurate than every other method. The Snip method has a larger coverage, but this is negated by its poorer accuracy. Interestingly, the Enrich approach initially has an increase in recall with less training data before decreasing as one would expect. This is probably because Enrich returns substantially fewer modules as the training set size increases. As a result, the most reasonably-sized clusters are found in the middle ranges.

In general, the predictions made on  $T_{\text{GS}}$  are much more accurate than those made on  $T_{\text{Dice}}$ . This suggests that the hierarchical decomposition defined by GS better represents the protein complexes within the PPI. Interestingly, for  $T_{\text{GS}}$ , the accuracy of all approaches except for Snip slightly increases as less training data is available. This may imply that with smaller training sets, only easy predictions are made. As the size of the training set increases, however, more difficult predictions are attempted, for which accuracy is generally lower.



**Fig. 3.** Accuracy and coverage comparison of the (A) Kennedy and (B) Warnecke OTU clustering methods. Although Kennedy and Warnecke produce the same clusters regardless of the training set, the predictions they make vary due to differences in the majority annotation within each cluster.

Further, for  $T_{GS}$ , Enrich especially benefits by choosing smaller, more reasonable clusters. Overall, VI-Cut makes accurate predictions covering many proteins on both trees, unlike any other method.

**Variations.** Results on  $Y_{ppi}$  echo the results obtained on the  $Y_{high\text{-}conf}$  network. Further, VI-Cut continued to outperform the other methods when using two other annotation transfer rules: plurality (transferring the most common annotation), and hypergeometric enrichment [23]. However, the performance of the plurality and hypergeometric rules was generally worse than the majority rule.

### 3.2 VI-Cut Yields Better Prediction of OTUs

We apply the same tests to predict OTU annotations for 16S DNA sequences. Predictions were made in the same way as with protein complexes, but instead of complex-membership annotations, we use known OTU annotations and transfer them to sequences with no OTU annotation. We again use the majority annotation transfer rule. We compare the predictive ability of the VI-Cut method for clustering metagenomic samples with previously published methods, including Kennedy [25], Fulthorpe [16], Schloss [42], Corby-Harris [9], Sogin [44], and Warnecke [50], described in Section 2.5. We also compare each method's ability to estimate the true number of OTUs present in the sample. The Corby-Harris approach resulted in nearly identical predictions and estimations as the Kennedy method. We therefore omit discussion of those results.

The VI-Cut generally outperforms each of these methods. Of the methods we compared against, Kennedy and Warnecke had the best overall coverage and accuracy, respectively. Figure 3 compares these methods with the VI-Cut. Compared to Kennedy, the VI-Cut mostly makes more accurate predictions, and covers a larger number of OTUs. Although Warnecke makes slightly more accurate predictions (average gain of 2%), the VI-Cut has significantly greater coverage. For example, with 80% of the sequences in the test set, the VI-Cut makes correct predictions for 1256 sequences, compared to just 1093 by Warnecke.

**Table 1.** Comparison of VI-Cut with other OTU clustering approaches applied to trees constructed from DOTUR with various parameters and distance thresholds, shown in parentheses. Performance is presented for 90% annotations excluded, average over 100 trials. **# OTUs** shows the average number of OTUs predicted by each method. The correct number of OTUs is 49. **Acc.** and **Coverage** show the accuracy and coverage for each approach. **Avg. VI** shows the VI distance of the clustering to the actual OTUs.

Method	# OTUs	Acc.	Coverage	Avg. VI
Tree 1: ClustalW, Felsenstein				
Kennedy (0.03)	70	97.6	85.5	0.087
VI-Cut	42	96.9	89.4	0.050
Tree 2: NAST, Felsenstein				
Fulthorpe (0.00)	386	98.9	49.6	0.646
VI-Cut	45	95.6	87.9	0.073
Tree 3: NAST, Jukes-Cantor				
Schloss (0.03)	99	97.5	80.5	0.157
VI-Cut	42	95.6	88.2	0.073
Tree 4: NAST, Jukes-Cantor				
Warnecke (0.01)	185	99.2	71.1	0.320
VI-Cut	42	95.6	88.2	0.073
Tree 5: MUSCLE, Jukes-Cantor				
Sogin (0.03)	96	97.5	78.2	0.190
VI-Cut	43	96.1	88.2	0.046

For all six trees, we find that the VI-Cut yields not only a closer VI distance to the true clustering, but also a much closer approximation to the true number of OTUs. There are 49 true OTUs in the sample and the VI-Cut estimates between 42 and 45, depending on which tree is used. This is a far better and more robust estimate of the true diversity of the population than the estimates of the other methods, which range between 70 and 386. The number of OTUs predicted are shown for test set size equal to 90% in Table 1. While it is true that our method starts with known annotations that hint at the number of true OTUs present in the sample beforehand, the average number of unique OTUs in the training set was only 35. Yet, VI-Cut was still able to identify that other OTUs exist, based on their topological non-compatibility with known annotations in the tree.

## 4 Conclusion

We presented a framework for finding cut-induced clusters in hierarchical tree decompositions that optimally match a partial set of known annotations, as measured by the variation of information [31]. Our VI-Cut method makes improved predictions of proteins’ membership in complexes and species annotations for metagenomic samples. While we showed that a generalization that allows multiple annotations per element is NP-hard, several open problems exist, such as providing an approximation guarantee on our heuristic that handles multiple annotations, and extensions that allow clusters to overlap. Nonetheless, the success of VI-Cut in two very different domains is evidence of the technique’s generality.

**Acknowledgements.** M.P. and C.K. thank NSF for grant IIS-0812111.

## References

1. Arnau, V., Mars, S., Marín, I.: Iterative cluster analysis of protein interaction data. *Bioinformatics* 21(3), 364–378 (2005)
2. Bader, G.D., Hogue, C.W.V.: An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* 4, 2 (2003)
3. Bernard, A., Vaughn, D.S., Hartemink, A.J.: Reconstructing the topology of protein complexes. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 32–46. Springer, Heidelberg (2007)
4. Böhm, C., Plant, C.: HISSCLU: a hierarchical density-based method for semi-supervised clustering. In: Proceedings of the 2008 International Conference on Extending Database Technology, pp. 440–451. ACM Press, New York (2008)
5. Brohee, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics* 7, 488+ (2006)
6. Brun, C., Chevenet, F., Martin, D., Wojcik, J., Guenoche, A., Jacq, B.: Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome Biol.* 5(1), R6 (2003)
7. Buehler, E.C., Sachs, J.R., Shao, K., Bagchi, A., Ungar, L.H.: The CRASSS plug-in for integrating annotation data with hierarchical clustering results. *Bioinformatics* 20(17), 3266–3269 (2004)
8. Cole, J.R., Chai, B., Farris, R.J., Wang, Q., Kulam, S.A., McGarrell, D.M., Garrity, G.M., Tiedje, J.M.: The ribosomal database project (RDP-II): sequences and tools for high-throughput rRNA analysis. *Nucleic Acids Res.* 33, 294–296 (2005)
9. Corby-Harris, V., et al.: Geographical distribution and diversity of bacteria associated with natural populations of *Drosophila melanogaster*. *Appl. Environ. Microbiol.* 73, 3470–3479 (2007)
10. DeSantis, T.Z., Hugenholtz, P., Keller, K., Brodie, E.L., Larsen, N., Piceno, Y.M., Phan, R., Andersen, G.L.: NAST: a multiple sequence alignment server for comparative analysis of 16s rRNA genes. *Nucleic Acids Res.* 34(Web Server issue), W394–W399 (2006)
11. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(11), 1944–1957 (2007)
12. Dotan-Cohen, D., Melkman, A.A., Kasif, S.: Hierarchical tree snipping: Clustering guided by prior knowledge. *Bioinformatics* 23(24), 3335–3342 (2007)
13. Eckburg, P.B., Bik, E.M., Bernstein, C.N., Purdom, E., Dethlefsen, L., Sargent, M., Gill, S.R., Nelson, K.E., Relman, D.A.: Diversity of the human intestinal microbial flora. *Science* 308(5728), 1635–1638 (2005)
14. Edgar, R.C.: MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* 32(5), 1792–1797 (2004)
15. Felsenstein, J.: PHYLIP: Phylogeny inference package (version 3.2). *Cladistics* 5, 164–166 (1989)
16. Fulthorpe, R.R., Roesch, L.F.W., Riva, A., Triplett, E.W.: Distantly sampled soils carry few species in common. *ISME J.* 2, 901–910 (2008)
17. Garey, M.R., Johnson, D.S.: Comptuers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
18. 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)

19. Guldener, U., Munsterkotter, M., Kastenmuller, G., Strack, N., van Helden, J., Lemer, C., Richelles, J., Wodak, S.J., Garcia-Martinez, J., Perez-Ortin, J.E., Michael, H., Kaps, A., Talla, E., Dujon, B., Andre, B., Souciet, J.L., De Montigny, J., Bon, E., Gaillardin, C., Mewes, H.W.: CYGD: the comprehensive yeast genome database. *Nucleic Acids Res.* 33(suppl. 1), D364+ (2005)
20. Hart, T.G., Ramani, A.K., Marcotte, E.M.: How complete are current yeast and human protein-interaction networks? *Genome Biol.* 7, 120+ (2006)
21. Jaccard, P.: Nouvelles recherches sur la distribution florale. *Bulletin de la Socit Vaudoise des Sciences Naturelles*, 223–270 (1908)
22. Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N.J., Chung, S., Emili, A., Snyder, M., Greenblatt, J.F., Gerstein, M.: A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science* 302(5644), 449–453 (2003)
23. Jukes, T.H., Cantor, C.R.: Evolution of Protein Molecules. Academic Press, London (1969)
24. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* 20(1), 359–392 (1998)
25. Kennedy, J., et al.: Diversity of microbes associated with the marine sponge, *Haliclona simulans*, isolated from Irish waters and identification of polyketide synthase genes from the sponge metagenome. *Environ. Microbiol.* 10, 1888–1902 (2008)
26. Kerrien, S., Alam-Faruque, Y., Aranda, B., Bancarz, I., Bridge, A., Derow, C., Dimmer, E., Feuermann, M., Friedrichsen, A., Huntley, R., Kohler, C., Khadake, J., Leroy, C., Liban, A., Liefertink, C., Montecchi-Palazzi, L., Orchard, S., Risso, J., Robbe, K., Roechert, B., Thorneycroft, D., Zhang, Y., Apweiler, R., Hermjakob, H.: IntAct—open source resource for molecular interaction data. *Nucleic Acids Res.* 35(Database issue), D561–D565 (2007)
27. Kimura, M.: A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* 16, 111–120 (1980)
28. King, A.D., Przulj, N., Jurisica, I.: Protein complex prediction via cost-based clustering. *Bioinformatics* 20(17), 3013–3020 (2004)
29. Li, X.L., Foo, C.S., Ng, S.K.: Discovering protein complexes in dense reliable neighborhoods of protein interaction networks. In: Comp. Syst. Bioinformatics Conference, vol. 6, pp. 157–168 (2007)
30. Mavromatis, K., Ivanova, N., Barry, K., Shapiro, H., Goltsman, E., McHardy, A.C.C., Rigoutsos, I., Salamov, A., Korzeniewski, F., Land, M., Lapidus, A., Grigoriev, I., Richardson, P., Hugenholtz, P., Kyropides, N.C.C.: Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nat. Methods*, 495–500 (2007)
31. Meila, M.: Comparing clusterings—an information based distance. *J. Multivariate Anal.* 98(5), 873–895 (2007)
32. Mirkin, B.: Mathematical classification and clustering. *J. Global Optim.* 12(1), 105–108 (1998)
33. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD Conference, pp. 419–432 (2008)
34. Navlakha, S., Schatz, M.C., Kingsford, C.: Revealing biological modules via graph summarization. *J. Comp. Biol.* 16(2), 253–264 (2009)
35. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* 103(23), 8577–8582 (2006)
36. Pei, P., Zhang, A.: A “seed-refine” algorithm for detecting protein complexes from protein interaction data. *IEEE T. Nanobiosci.* 6(1), 43–50 (2007)

37. Qiu, J., Noble, W.S.: Predicting co-complexed protein pairs from heterogeneous data. *PLoS Comp. Biol.* 4(4) (2008)
38. Rand, W.M.: Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* 66(336), 846–850 (1971)
39. Rives, A.W., Galitski, T.: Modular organization of cellular networks. *Proc. Natl. Acad. Sci. USA* 100(3), 1128–1133 (2003)
40. Samanta, M.P., Liang, S.: Predicting protein functions from redundancies in large-scale protein interaction networks. *Proc. Natl. Acad. Sci. USA* 100(22), 12579–12583 (2003)
41. Schloss, P.D., Handelsman, J.: Introducing DOTUR, a computer program for defining operational taxonomic units and estimating species richness. *Appl. Environ. Microbiol.* 71(3), 1501–1506 (2005)
42. Schloss, P.D., Handelsman, J.: Toward a census of bacteria in soil. *PLoS Comp. Biol.* 2(7), e92 (2006)
43. Sharan, R., Ulitsky, I., Shamir, R.: Network-based prediction of protein function. *Nat. Mol. Syst. Biol.* 3, 88 (2007)
44. Sogin, M.L.L., Morrison, H.G.G., Huber, J.A.A., Welch, D.M.M., Huse, S.M.M., Neal, P.R.R., Arrieta, J.M.M., Herndl, G.J.J.: Microbial diversity in the deep sea and the underexplored “rare biosphere”. *Proc. Natl. Acad. Sci. USA* 103(32), 12115–12120 (2006)
45. Tan, M., Smith, E., Broach, J., Floudas, C.: Microarray data mining: A novel optimization-based approach to uncover biologically coherent structures. *BMC Bioinformatics* 9(1), 268 (2008)
46. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22(22), 4673–4680 (1994)
47. Toronen, P.: Selection of informative clusters from hierarchical cluster tree with gene classes. *BMC Bioinformatics* 5, 32 (2004)
48. van Dongen, S.: A cluster algorithm for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam (2000)
49. Wang, Q., Garrity, G.M., Tiedje, J.M., Cole, J.R.: Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl. Environ. Microbiol.* 73(16), 5261–5267 (2007)
50. Warnecke, F., Luginbühl, P., Ivanova, N., Ghaseemian, M., Richardson, T.H., Stege, J.T., Cayouette, M., McHardy, A.C., Djordjevic, G., Aboushadi, N., Sorek, R., Tringe, S.G., Podar, M., Martin, H.G., Kunin, V., Dalevi, D., Madejska, J., Kirton, E., Platt, D., Szeto, E., Salamov, A., Barry, K., Mikhailova, N., Kyripides, N.C., Matson, E.G., Ottesen, E.A., Zhang, X., Hernández, M., Murillo, C., Acosta, L.G., Rigoutsos, I., Tamayo, G., Green, B.D., Chang, C., Rubin, E.M., Mathur, E.J., Robertson, D.E., Hugenholz, P., Leadbetter, J.R.: Metagenomic and functional analysis of hindgut microbiota of a wood-feeding higher termite. *Nature* 450(7169), 560–565 (2007)
51. Yu, H., Paccanaro, A., Trifonov, V., Gerstein, M.: Predicting interactions in protein networks by completing defective cliques. *Bioinformatics* 22(7), 823–829 (2006)
52. Zhu, X., Gerstein, M., Snyder, M.: Getting connected: analysis and principles of biological networks. *Genes Dev.* 21(9), 1010–1024 (2007)

# Identification and Frequency Estimation of Inversion Polymorphisms from Haplotype Data

Suzanne S. Sindi<sup>1,3</sup> and Benjamin J. Raphael<sup>2,3</sup>

<sup>1</sup> Division of Applied Mathematics

<sup>2</sup> Department of Computer Science

<sup>3</sup> Center for Computational Molecular Biology  
Brown University, Providence RI 02912, USA

[raphael@brown.edu](mailto:raphael@brown.edu)

**Abstract.** Structural rearrangements, including copy-number alterations and inversions, are increasingly recognized as an important contributor to human genetic variation. Copy number variants are readily measured via array-based techniques like comparative genomic hybridization, but copy-neutral variants such as inversion polymorphisms remain difficult to identify without whole genome sequencing. We introduce a method to identify inversion polymorphisms and estimate their frequency in a population using readily available single nucleotide polymorphism (SNP) data. Our method uses a probabilistic model to describe a population as a mixture of forward and inverted chromosomes and identifies putative inversions by characteristic differences in haplotype frequencies around inversion breakpoints. On simulated data, our method accurately predicts inversions with frequencies as low as 25% in the population and reliably estimates inversion frequencies over a wide range. On the human HapMap Phase 2 data, we predict between 88 and 142 inversion polymorphisms with frequency ranging from 20 to 92 percent. Many of these correspond to known inversions or have other evidence supporting them, and the predicted inversion frequencies largely agree with the limited information presently available.

## 1 Introduction

Characterization of variation in human populations has primarily focused on single nucleotide polymorphisms (or SNPs); large genotyping projects like HapMap [1] and others [2] provide a catalog of human SNPs useful for disease association and population genetics studies. There is now increasing appreciation that SNPs are not the only source of genetic variation in humans and that structural rearrangements including copy number variants and inversion polymorphisms are common [3]. Moreover, these structural variants have also been associated with disease [4] and have undergone selection in human lineages [5,6].

Measurement of copy number variants is becoming routine with microarray based techniques like SNP chips [7,8] or array comparative genomic hybridization [9]. In addition several methods have been introduced to identify copy-number variants from SNP data [10,11,12]. However, measurement of inversions is difficult because they do not lead to changes in DNA copy number and thus cannot be measured via microarray

hybridization. With the advent of whole genome sequencing, inversions and inversion polymorphisms have been found to be common in mammalian genomes, particularly microinversions less than a megabase in length that were largely invisible to earlier cytogenetic techniques. Paired-end sequencing studies in the human genome [13][14][15] and whole-genome resequencing of two individuals [16][17] have collectively identified over 400 inversion polymorphisms, as reported in the Database of Genomic Variants [18], but this is likely to be an underestimate due to the fact that few individuals have been assayed. Indeed, a comparison of human and chimpanzee genome sequences reported 1,576 putative inversions that distinguished these two species and also found that some of these were polymorphic in one lineage [19]. A later study [20] showed that while some of these inversions were artifacts, microinversions are frequent enough in mammalian genomes to reconstruct mammalian phylogeny using only microinversions in a 3Mb locus as phylogenetic characters. Finally, little is known about the frequencies of inversion polymorphisms in different human populations, with a few notable exceptions such as a common inversion under positive selection in Europeans that was linked to increased fertility [5].

While inversion polymorphisms in the human genome have only recently received attention, inversions have been studied in *Drosophila* since the pioneering work of Sturtevant and Dobzhansky in the 1930's. Indeed much of the knowledge of the population genetics of inversions comes from studies in *Drosophila*, where inversion polymorphisms are large enough to be observed with microscopes. Inversions segregating in a population have been shown to leave distinct population genetic signatures including reduced recombination rates and nucleotide variation [21] that result from suppression of recombination between standard and inverted arrangements in heterozygotes [22]. These signatures are extremely subtle, yet Bansal et al. [23] cleverly designed a method to predict inversion polymorphisms from SNP data. Their method relies on a statistic derived from patterns of long-range linkage disequilibrium between SNPs on either side of inversion breakpoints. However, by relying on linkage disequilibrium, a quantity computed over the whole population, their method has almost no power to identify inversions appearing in a population at frequencies less than 50%, and has limited power to detect inversions even at frequencies as high as 75%. Moreover, their method does not estimate the frequency of an inversion polymorphism or distinguish individuals containing an inversion.

We introduce a novel method to identify inversion polymorphisms and estimate their frequencies from SNP data. Our method employs a generative probabilistic model to identify characteristic differences in haplotype frequencies around inversion breakpoints. In our model, we consider whether the observed haplotypes surrounding a pair of genomic loci can be explained as arising from a mixture of *forward* chromosomes, with SNPs in the reference genome order, and *inverted* chromosomes, with SNPs in the reverse order. We compare this mixture model to the null model that describes the observed haplotypes assuming no inversion, and determine whether a mixture with a specific frequency is a better description of the observed haplotypes. We applied our method to both simulated data and data from the HapMap Phase 2 [1]. On simulated data, we accurately predict inversions with frequencies as low as 25% in the population and we reliably estimate the frequency of inversions over a range of frequencies from

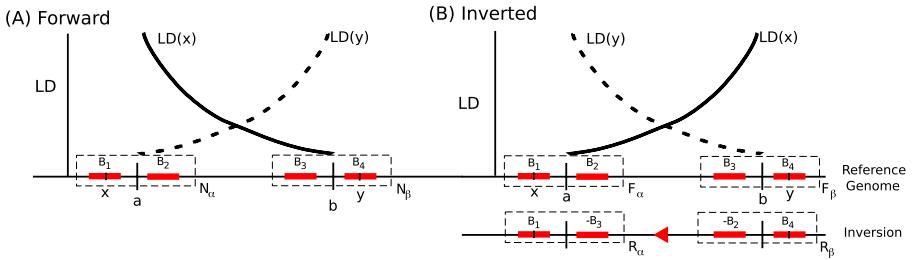
20–80%. Simulations also suggest that our method has a modest false positive rate. On the HapMap data, we predict 125 inversions in the CEU population, 142 inversions in the YRI population, and 88 in the combined JPT+CHB population. Some of these overlap known inversions, and others have evidence supporting their validity. Moreover, our predicted inversion frequencies largely agree with the limited information presently available. Detection of inversions from SNP data is a cost-effective strategy for identification of new variants, and for estimating frequencies of existing inversion polymorphisms. Our method is the first to predict inversion frequencies and to reliably detect inversions that are the minor allele in the population. Thus our method will find increasing use as genotypes of ever larger number of individuals become available.

## 2 Methods

### 2.1 Overview of Inversion Detection

The motivating idea of our approach is that an inversion polymorphism will disrupt linkage between nearby SNPs on either side of an inversion breakpoint in a characteristic pattern. Typically, the correlation between a pair of SNPs in the genome decays as a function of distance due to genetic recombination (Figure 1A). However, if there is an inversion in the population, and one computes the correlation between pairs of SNPs in the subset of individuals with the inversion, then there will be high correlation between pairs of SNPs at *opposite* ends of the inversion, but low correlation between pairs of SNPs on either end of the *same* inversion breakpoint (Figure 1B). In population genetic terms, we expect long-range linkage disequilibrium (LD) between pairs of SNPs near opposite inversion breakpoints. Bansal et al. [23] cleverly exploited the signal of apparent long-range LD to predict inversion polymorphisms in the human genome. However, there are two problems with the use of LD for inversion detection. First, long-range LD can be present in the genome for a variety of reasons such as selective sweeps or reduced rates of recombination, and thus long-range LD alone is not specific enough to identify inversion polymorphisms. Second, because LD is a quantity computed over the whole population, the signal of long-range LD will be reduced as the frequency of inverted chromosomes in the population decreases. Bansal et al. attempt to overcome the first limitation using a permutation test that compares their inversion score with the scores obtained by permuting the SNPs within a putative inversion. This test avoids problems with long haplotypes that might result from selective sweeps. However, they are unable to overcome the second problem, and consequently their method only has power to detect inversion polymorphisms that are the *major* allele (frequency greater than 50%) in the population. In their reported tests, the power decreases sharply for inversion polymorphisms with frequency less than 75%.

We overcome the limitations of LD for inversion prediction by explicitly modeling the haplotype block structure of the genomic regions near the two breakpoints of a putative inversion (Figure 1). *Haplotype blocks*, groups of adjacent SNPs that have undergone limited recombination in the human lineage were noted in early human genotyping studies [24][25]. Our model is based on the premise that *forward* chromosomes, with SNPs in the reference genome order, and *inverted* chromosomes, with SNPs in the



**Fig. 1.** (A) Typically, correlation between nearby SNPs decays with distance, as indicated by the linkage disequilibrium  $LD(x)$  between a SNP  $x$  and neighboring SNPs. (B) An inversion with breakpoints  $a$  and  $b$  produces apparent long-range LD coupled with absence of short-range LD. Groups of adjacent SNPs, in blocks  $B_1, B_2, B_3$ , and  $B_4$ , describe the correlation structure near the inversion breakpoints. An inversion polymorphism yields a mixture of forward chromosomes – with haplotype blocks  $B_1 \circ B_2$  and  $B_3 \circ B_4$  – and inverted chromosomes with haplotype blocks  $B_1 \circ B_3$  and  $B_2 \circ B_4$ .

reverse order will exhibit different haplotype block structure. We derive a probabilistic model that describes the population of chromosomes as a mixture of both types of chromosomes, and the forward and inverted subpopulations are in turn described by the frequencies of the haplotypes that span the inversion breakpoints.

The idea of partitioning the observed chromosomes into forward and inverted subpopulations using haplotype frequencies is reminiscent of population substructure methods like STRUCTURE [26], EIGENSTRAT [27], or [28], which define subpopulations by differences in SNP frequencies. However, our model is fundamentally different. Population substructure methods use information from *many* uncorrelated SNPs to define subpopulations, while for inversion detection, there are relatively few such SNPs available, because the subpopulations of interest are defined by local and not global (i.e. genome-wide) information. On the other hand, an inversion has a very specific dependency structure, which we exploit in our model.

## 2.2 Haplotype Mixture Model

Suppose that  $m$  chromosomes are assayed for SNPs at fixed locations. Consider two locations  $a$  and  $b$  on the genome, and let  $B_1, B_2, B_3$ , and  $B_4$  be the  $L$  SNPs immediately to the left and right of  $a$  and  $b$ . We refer to the four blocks  $B_1, B_2, B_3$  and  $B_4$  collectively as a *block configuration*. We assume that each SNP is biallelic and denote the two alleles as 0 and 1. For a chromosome  $\mathbf{x}$ , let  $B_i(\mathbf{x}) \in \{0, 1\}^L$  denote the alleles in chromosome  $\mathbf{x}$  for SNPs in block  $i$ , and let  $B_{i,j}(\mathbf{x}) = B_i(\mathbf{x}) \circ B_j(\mathbf{x})$  denote the concatenation of alleles in blocks  $i$  and  $j$ , for  $1 \leq i, j \leq 4$ .

In the case of no inversion, the SNPs in the concatenation  $B_1 \circ B_2$  are highly correlated, and thus  $B_1 \circ B_2$  forms a haplotype block. Similarly, the concatenation  $B_3 \circ B_4$  forms a haplotype block (Figure 1). Generally, correlation between SNPs decreases with distance in the genome. Thus, if  $a$  and  $b$  are sufficiently far apart, then these two haplotype blocks are independent. That is, the probability of the observed block configuration

$B_1(\mathbf{x})$ ,  $B_2(\mathbf{x})$ ,  $B_3(\mathbf{x})$ , and  $B_4(\mathbf{x})$  is

$$P(\mathbf{x}|N_\alpha, N_\beta) = P_{N_\alpha}(B_{1,2}(\mathbf{x})) \times P_{N_\beta}(B_{3,4}(\mathbf{x})), \quad (1)$$

where  $N_\alpha$  and  $N_\beta$  are discrete probability distributions on  $\{0, 1\}^{2L}$  giving the frequencies of haplotypes  $B_1 \circ B_2$  and  $B_3 \circ B_4$ , respectively. Similarly, if  $\mathbf{x}$  is a chromosome with an inversion with breakpoints  $a$  and  $b$ ,

$$P_I(\mathbf{x}|R_\alpha, R_\beta) = P_{R_\alpha}(B_{1,3}(\mathbf{x})) \times P_{R_\beta}(B_{2,4}(\mathbf{x})), \quad (2)$$

where  $R_\alpha$  and  $R_\beta$  are the probability distributions of inverted haplotypes  $B_1 \circ B_3$  and  $B_2 \circ B_4$ , respectively.<sup>1</sup>

If the population is a mixture of forward and inverted chromosomes and the frequency of inverted chromosomes in the population is  $\pi$ , then the probability of the observed block configuration for  $\mathbf{x}$  is

$$P_M(\mathbf{x}|F_\alpha, F_\beta, R_\alpha, R_\beta, \pi) = (1 - \pi)P(\mathbf{x}|F_\alpha, F_\beta) + \pi P_I(\mathbf{x}|R_\alpha, R_\beta). \quad (3)$$

Here, we use  $F_\alpha$  and  $F_\beta$  to denote the discrete probability distributions for the forward haplotypes,  $B_{1,2}$  and  $B_{3,4}$ , in order to differentiate these parameters in the mixture model from the parameters  $N_\alpha$  and  $N_\beta$  in the model with no inversion.

Given a set  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$  of chromosomes and fixed breakpoints  $a$  and  $b$  we wish to test the hypothesis that  $\pi = 0$  (null model) vs.  $\pi > 0$  (alternative model). The likelihood of observing the data according to the null model is

$$L_0(\mathbf{X}|N_\alpha, N_\beta) = \prod_{k=1}^m P(\mathbf{x}_k|N_\alpha, N_\beta). \quad (4)$$

Similarly, the likelihood of observing data according to the alternate model is

$$L_M(\mathbf{X}|F_\alpha, F_\beta, R_\alpha, R_\beta, \pi) = \prod_{k=1}^m P_M(\mathbf{x}_k|F_\alpha, F_\beta, R_\alpha, R_\beta, \pi). \quad (5)$$

Each of the probability distributions  $N$ ,  $F$  and  $R$  require at most  $2^{2L} - 1$  parameters to specify the probability of  $2^{2L}$  haplotypes of length  $2L$ . On real data, many of the haplotypes are not observed, and thus we restrict the probability distributions to have mass on the subset of observed haplotypes. Let  $\Omega_0$  be the parameter space of the null model and  $\Omega$  be the parameter space of the alternative model. Note that  $\Omega_0 \subseteq \Omega$ . Let  $\Lambda$  be the likelihood ratio,

$$\Lambda = \frac{\max_{\omega \in \Omega_0} L_0(X|\omega)}{\max_{\omega \in \Omega} L_M(X|\omega)}. \quad (6)$$

The value in the numerator is obtained from the maximum likelihood estimates of  $N_\alpha$  and  $N_\beta$ , which are equal to the empirical frequencies of observed haplotypes in  $\mathbf{X}$ .

---

<sup>1</sup> Strictly speaking, the inverted haplotypes are  $B_1 \circ (-B_3)$  and  $B_2 \circ (-B_4)$ , where the minus signs indicate the reverse orientation. Since the orientation does not affect the model, we omit them for simplicity.

We compute the denominator using the Expectation-Maximization (EM) algorithm [29] for mixtures of probability distributions. For each chromosome  $\mathbf{x}_k$ , we define a hidden binary variable,  $z_k \in \{0, 1\}$ , which indicates whether the corresponding chromosome is a member of the forward or inverted population. The EM algorithm iteratively updates the model parameters. Assume we currently have  $\omega = (F_\alpha, F_\beta, R_\alpha, R_\beta, \pi)$ . The responsibility of the  $k$ -th chromosome to the forward model is denoted  $r_{0,k}(\omega)$  and given by  $r_{0,k} = P(z_k = 0 | \mathbf{x}_k, \omega)$ , similarly for the mixture model,  $r_{1,k}(\omega) = P_I(z_k = 1 | \mathbf{x}_k, \omega)$ . Note that for each  $k$ ,  $r_{0,k}(\omega) + r_{1,k}(\omega) = 1$ . Following the EM algorithm, we find the value  $\omega' = (F'_\alpha, F'_\beta, R'_\alpha, R'_\beta, \pi')$  at the next step, by maximizing the following expression:

$$\begin{aligned} \omega' = \operatorname{argmax}_{\omega' \in \Omega} & \sum_{k=1}^m [\log ((1 - \pi') P(\mathbf{x}_k | F'_\alpha, F'_\beta)) r_{0,k}(\omega) \\ & + \log (\pi' P_I(\mathbf{x}_k | R'_\alpha, R'_\beta)) r_{1,k}(\omega)]. \end{aligned} \quad (7)$$

Define  $r_0(\omega)$  to be the sum over all chromosomes of responsibilities to the forward model, and similarly define  $r_1(\omega)$  for the reverse model,

$$r_0(\omega) = \sum_{k=1}^m r_{0,k}(\omega) \text{ and } r_1(\omega) = \sum_{k=1}^m r_{1,k}(\omega). \quad (8)$$

The updated mixing parameter is then  $\pi' = \frac{r_1(\omega)}{r_0(\omega) + r_1(\omega)}$ . Similarly, we compute the updated values  $F'_\alpha, F'_\beta, R'_\alpha$  and  $R'_\beta$  by re-casting the maximization problem (7) from a sum over all chromosomes  $\mathbf{x}_k$  to a sum over all observed strings in  $\{0, 1\}^{2L}$ . We are then able to derive the updates  $F'_\alpha, F'_\beta, R'_\alpha$  and  $R'_\beta$  analytically by solving a linear system of equations. Finally, since EM finds local optima, we record the best solution found with multiple initial values for  $\omega$ .

Note that our model does not explicitly denote a chromosome as forward or inverted. While a “soft” classification can be derived from the responsibilities  $r_{0,k}$  and  $r_{1,k}$  returned by EM, we found that these responsibilities did not yield a useful classifier in practice. An alternative approach is to explicitly estimate the joint likelihood of  $\mathbf{X}$  and  $Z = (z_1, \dots, z_m)$ , following the techniques used in population substructure algorithms such as STRUCTURE [26]. However, this approach is more data intensive, and not practical for the relatively small number of chromosomes in the HapMap data.

### 2.3 Model Selection

The likelihood ratio  $\Lambda$  defined in (6) will always be less than one, since  $\Omega_0 \subseteq \Omega$  and the likelihood  $L_0$  of the null model can be obtained by setting  $\pi = 0$  in the expression for the likelihood  $L_M$  of the mixed model. Thus, we must choose a threshold on the value of  $\Omega$  at which to reject the null hypothesis  $\pi = 0$ . Classically, the likelihood ratio test is used for this purpose. This test typically employs the approximation that the distribution of  $\Gamma = -2 \log \Lambda$  asymptotically (as the size of the dataset grows large) follows the  $\chi^2$  distribution with  $d = |\Omega| - |\Omega_0|$  degrees of freedom. A  $p$ -value for

rejecting the null hypothesis is derived from the  $\chi^2$  distribution. This approach was used in [12] for identification of deletion polymorphisms.

We found that the  $\chi^2$  distribution was a poor approximation of the empirical distribution of  $\Gamma$  on the HapMap data, particularly for inversion lengths less than several hundred kilobases. Thus, we derive the empirical probability density  $f_{\Gamma;l,d}$  of  $\Gamma$  from pairs of genomic regions of similar length  $l$  and degrees of freedom  $d$ . For the analysis of the human genome, we define  $f_{\Gamma;l^*,d^*}$  as the empirical probability density of  $\Gamma$  for all block configurations with  $|l - l^*| \leq 70kb$  and  $|d - d^*| \leq 5$ . For a pair of breakpoints  $a$  and  $b$ , with  $b - a = l^*$ , log-likelihood ratio  $\Gamma^*$ , and degrees of freedom  $d^*$ , we define the score  $S(a, b) = \sum_{x \geq \Gamma^*} f_{\Gamma;l^*,d^*}(x)$ .

To achieve even greater specificity and computational efficiency, we restrict the pairs of genomic regions that we consider as possible inversion breakpoints according to the degrees of freedom. The degrees of freedom  $d$  is equal to the number of additional parameters required to specify the probability distributions for the reversed haplotype blocks  $B_1 \circ B_3$  and  $B_2 \circ B_4$  plus one additional parameter  $\pi$ . For real inversion polymorphisms, we expect that the reversed haplotype blocks are genuine haplotype blocks in the genome (i.e. regions of highly correlated SNPs), and thus the number of degrees of freedom should be relatively small. Thus, in the results below, we consider only block configurations where the degrees of freedom is no larger than 70. Note that because the number of chromosomes in the HapMap datasets is fixed at 120 (for the CEU and YRI populations) and 180 (for the CHB+JPT population), the number of parameters that can be estimated reliably is limited and thus the degrees of freedom restriction alleviates overfitting by our model.

## 2.4 Haplotype Block Definition

The final ingredient of our model is the definition of block configurations  $B_1$ ,  $B_2$ ,  $B_3$ , and  $B_4$ . While it is possible to define blocks with a fixed number of SNPs or fixed physical distance, doing so can lead to the spurious identification of long haplotypes that span all four blocks  $B_1$ ,  $B_2$ ,  $B_3$ , and  $B_4$ . For a true inversion polymorphism we expect that the concatenated block  $B_1 \circ B_2$  will have high diversity over the population because of the presence of inverted chromosomes where  $B_1 \circ B_2$  is *not* a haplotype block. Indeed, we found that many known inversions reported in [15] have breakpoints in regions of high haplotype diversity, where diversity was defined as the information theoretic entropy of the subset of  $\{0, 1\}^L$  defined by  $B_1 \circ B_2$  or  $B_3 \circ B_4$ . We use this observation to define the following adaptive procedure to identify block configurations with high entropy.

We first remove SNPs whose minor allele frequency is less than 10% and SNPs from centromeric regions. We consider regions between any two remaining SNPs (except the two spanning the centromere) as candidates for containing an inversion breakpoint. We construct blocks centered around these regions by beginning with 3 SNPs immediately to the left and right of each candidate breakpoint region, and iteratively add SNPs, one to each side symmetrically, as needed until the entire haplotype block has entropy in the top 10% of regions over the genome with the same number of SNPs. We consider at most 15 SNPs to the left and right of each candidate region. If the entropy has not exceeded the threshold at this point, we consider the presence of an inversion breakpoint

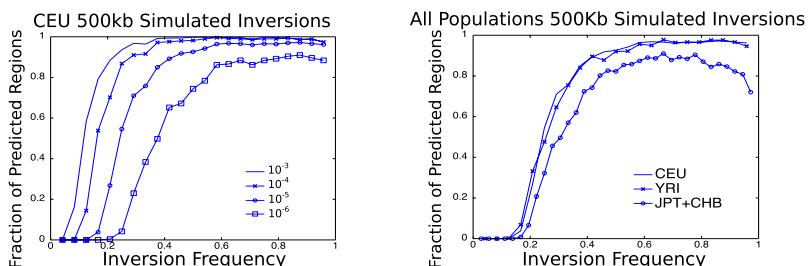
in this region unlikely and do not examine these blocks in our model. Note that this construction leads to overlapping block configurations tiled across the genome.

### 3 Results

#### 3.1 Simulated Inversions

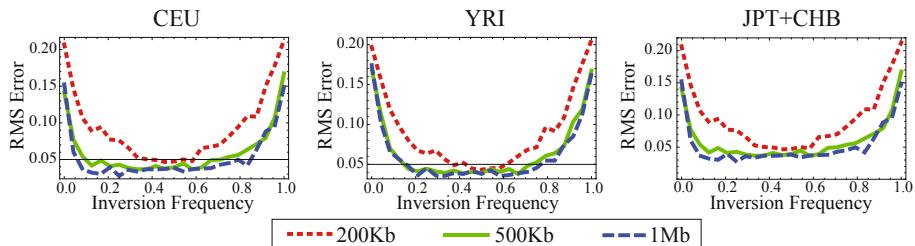
We first assessed the ability of our model to identify inversions and accurately determine their frequency on simulated data. Since there is no known method to simulate the effect of an inversion on recombination rates, we followed the approach of [23] and generated synthetic inversions in the HapMap data [1]. For a fixed inversion length  $l$  and frequency  $\pi$ , we select two random positions  $a$  and  $b$  in the genome separated by distance  $l$  and reverse the order of the SNPs between these positions in a fraction  $\pi$  of the chromosomes. We generated 500 synthetic inversions for  $l$  equal to 200kb, 500kb, and 1Mb over a range of inversion frequencies in each of the three HapMap Phase2 populations: CEU, YRI, and JPT+CHB.

Figure 2 shows the average fraction of 500kb inversions that we detect in the CEU population at various score thresholds. We successfully recover over 94% of the simulated inversions with frequency greater than 25% at a threshold of  $10^{-3}$ , and recover the majority of these inversions at a lower threshold of  $10^{-5}$ . We also detect the majority of simulated inversions of length 500kb and frequency at least 33% in all 3 HapMap populations. Although our power to detect inversions declines for shorter inversions, we still recover about 20% of all inversions with length 200kb and a frequency of at least 37.5% using a threshold of  $10^{-5}$  (Supplemental Figure 1). These results represent a substantial improvement over the method of [23], whose power to predict simulated inversions decreases rapidly for frequencies below 75% and is nearly zero for inversions with frequency 50% or lower (see Figure 2 in [23]). This limitation on their performance is a direct consequence of the use linkage disequilibrium (LD). Since LD is a quantity computed over the whole population, any perturbation in normal LD patterns resulting from an inversion is reduced as the frequency of the inversion decreases.



**Fig. 2.** (a) Average fraction of 500kb simulated inversions in CEU predicted for various score thresholds. (b) 500Kb simulated inversions in all populations at score threshold of  $10^{-5}$ .

<sup>2</sup> Supplemental Information is available at <http://www.cs.brown.edu/people/braphael/supplements/inversions>



**Fig. 3.** The average root-mean-square error in our estimation of the inversion frequency  $\pi$  for simulated inversions of lengths 200kb, 500kb, and 1Mb in the three HapMap populations

We also examined the ability of our method to estimate the inversion frequency  $\pi$  (Figure 3). The average error is less than 5% for 500kb-1Mb inversions at intermediate frequencies 20-80%, and only slightly higher for 200kb inversions. At extremely high and low frequencies, the error increases sharply. This is likely due to the small number of forward or inverted chromosomes at these frequencies; we expect that with a larger number of chromosomes than available in HapMap, the error would be less at the extreme frequencies. We note that our model is not merely finding a haplotype partition; the STRUCTURE [26,30] program failed to identify inversions in synthetic data that were identified readily by our model.

### 3.2 Estimation of False Positive Rate

Following the approach of Bansal et al. [23], we estimated the false-positive rate of our method using simulated genotype data generated by the Cosi program [31]. Cosi generates genotypes using a model with complex demographic histories, including variable recombination rates and patterns of human migration, but does not consider the effect of inversions or other large structural variants. Thus, we assume that inversions detected in simulated data from Cosi are false positives. We used the bestfit model parameters (Table 1 in [31]) and identified the “European” population with the HapMap CEU population, the “Asian” population with the combined JPT+CHB populations and the “African” population with the YRI population. For each population, we generated 25Mb chromosomes for the number of individuals in the HapMap data. We randomly removed SNPs until the simulated chromosomes had similar SNP density to the respective HapMap population. We ran our inversion prediction program with a score threshold of  $10^{-5}$  and clustered the resulting predictions with the procedure described below. We performed 100 simulations for each population and predicted a total of 4 inversions in the European population, 13 inversions the Asian population and 0 inversions in the African population. Thus, we estimate that in the Asian population where we make the most predictions, our false positive rate is 0.13 per 25Mb region, or approximately 15 false positive predictions over the whole genome. We emphasize that these simulations are not a rigorous assessment of the false-positive rate of our method, but are only a surrogate until larger collections of inversion polymorphisms become available for more careful benchmarking.

### 3.3 Known Inversion Polymorphisms

We next tested our model on the 178 validated inversion polymorphisms identified by fosmid paired-end sequencing of 8 individuals from the HapMap populations: 2 CEU individuals, 4 YRI individuals, 1 CHB individual and 1 JPT individual [15]. Of these validated inversions, 27, 25, and 21 had lengths between 200kb and 1.2Mb in the CEU, YRI, and combined JPT+CHB populations, respectively. In addition, 3, 2 and 2 of these inversions in the CEU, YRI and combined JPT+CHB populations respectively overlapped centromeric regions and were not considered by our model. For the remaining inversions, we computed the scores of all block configurations with inversion breakpoints within 50kb of the reported breakpoints.

Supplemental Tables 1, 2, and 3 list the best scoring block configuration for each of the inversions. Note that for a number of inversions (12 in CEU, 11 in YRI and 8 in JPT+CHB) there were no valid block configurations within 50kb of the inversion breakpoints because of either low SNP density or low entropy of the surrounding haplotype blocks. Most of the inversions have a block configuration with score below  $10^{-3}$  including 7/12 in the CEU population, 10/12 in the YRI population and 8/11 in the combined JPT+CHB population, and only 1/35 inversions had a score above 0.02. This shows that nearly all known inversions have “extremal” scores compared to similarly sized regions of the human genome. In addition, our predicted inversions with intermediate frequencies are largely consistent with the reported experimental validation: 4 of the 7 CEU inversions with predicted frequencies between 0.3 and 0.5 were found in only 1 of the 2 CEU individuals tested by [15]. Similarly, 4/6 of the JPT+CHB inversions and 4/7 YRI inversions with predicted frequencies between 0.3 and 0.5 were found in only a fraction of the tested individuals.

### 3.4 HapMap Predictions

We ran our method on the Phase 2 HapMap data for CEU, YRI and JPT+CHB, testing all block configurations with minimum length 200Kb and maximum length 1.2Mb on the 22 autosomes, and classifying block configurations with score below  $10^{-5}$  as candidate inversions. Since several overlapping block configurations might identify the same inversion, we clustered block configurations if both the left and right haplotype blocks overlap. We clustered block configurations with a score below  $10^{-2}$  and reported any cluster with a regions with a score below  $10^{-5}$ . For each cluster, we select the block configuration with the minimum score as representative. After clustering, we predict 125 inversions in CEU, 142 in YRI and 88 in JPT+CHB (Supplemental Table 4).

A number of our predicted inversions agree with known inversions or have other supporting evidence (Table I). In particular, we successfully identify several of the validated inversions reported in [15] that have lengths between 200kb and 1.2Mb including 3/12, 2/12 and 3/11 of the inversions in the CEU, YRI, and JPT+CHB populations respectively. Moreover, validated inversions are among the highest scoring predictions. All 3 of the validated inversions in the CEU population were in our top 63 predictions for CEU ranked by score. The two validated inversions in our YRI predictions are in the top 32 predictions and the 3 JPT+CHB validated inversions are in the top 21 predictions. Our predictions also identified an inversion in the CEU population on chr9

**Table 1.** Predicted inversions in the HapMap populations with supporting evidence. Predicted inversions in boldface indicate those where both predicted breakpoints are near (within 150kb) coordinates reported in [15]. Other predictions were supported by end-sequenced fosmid clones from [15] for members of the indicated HapMap population. The procedure to determine these supporting clones is described in [32].

Chromosome	Left Breakpoint (Mb)	Right Breakpoint (Mb)	Score	Inversion Frequency	Evidence
<b>CEU</b>					
chr1	144.827-144.899	144.942-145.821	$2.2 \times 10^{-6}$	0.21	CEU clones (2/2)
chr1	144.941-145.819	145.939-145.991	$4.4 \times 10^{-6}$	0.36	CEU clones (2/2)
<b>chr1</b>	<b>165.967-166.301</b>	<b>169.408-169.547</b>	<b><math>4.3 \times 10^{-6}</math></b>	<b>0.54</b>	<b>validated inversion</b>
chr7	5.706-5.791	6.549-6.648	$4.8 \times 10^{-6}$	0.60	CEU clones (2/2)
chr9	67.114-67.176	67.730-67.934	$4.7 \times 10^{-6}$	0.52	known inversion [18]
chr15	18.884-19.109	20.077-20.301	0	0.57	CEU clones (2/2)
<b>chr15</b>	<b>82.608-82.610</b>	<b>82.949-82.951</b>	<b><math>7.5 \times 10^{-6}</math></b>	<b>0.57</b>	<b>validated inversion</b>
<b>chr16</b>	<b>14.587-14.594</b>	<b>15.286-15.387</b>	<b><math>6.2 \times 10^{-6}</math></b>	<b>0.31</b>	<b>validated inversion</b>
<b>YRI</b>					
chr2	97.267-97.472	97.482-97.633	$6.4 \times 10^{-6}$	0.43	YRI clones (4/4)
chr2	110.196-110.205	110.340-111.117	0	0.36	YRI clones (1/4)
chr7	62.101-62.119	62.414-62.475	$3.2 \times 10^{-6}$	0.52	YRI clones (1/4)
chr8	7.228-7.773	7.821-8.123	$5.5 \times 10^{-6}$	0.44	YRI clones (4/4)
chr10	50.793-51.165	51.258-51.463	$5.7 \times 10^{-6}$	0.38	YRI clones (1/4)
chr10	81.056-81.229	81.378-81.587	0	0.40	YRI clones (2/4)
<b>chr15</b>	<b>82.613-82.708</b>	<b>82.946-82.949</b>	<b><math>1.7 \times 10^{-6}</math></b>	<b>0.34</b>	<b>validated inversion</b>
<b>chr16</b>	<b>21.516-21.528</b>	<b>22.539-22.555</b>	<b>0</b>	<b>0.33</b>	<b>validated inversion</b>
<b>JPT+CHB</b>					
chr1	<b>12.870-12.877</b>	<b>13.259-13.449</b>	<b><math>1.6 \times 10^{-6}</math></b>	<b>0.61</b>	<b>validated inversion</b>
<b>chr1</b>	<b>146.150-146.462</b>	<b>146.530-146.531</b>	<b>0</b>	<b>0.78</b>	<b>validated inversion</b>
chr2	89.141-89.382	89.676-89.761	$4.0 \times 10^{-6}$	0.60	CHB clones (1/1)
chr8	7.228-7.756	7.810-8.115	$9.2 \times 10^{-7}$	0.58	clones JPT(1/1), CHB(1/1)
chr10	51.230-51.240	51.258-51.463	$8.2 \times 10^{-6}$	0.49	CHB clones (1/1)
<b>chr16</b>	<b>14.578-14.594</b>	<b>15.277-15.286</b>	<b><math>1.7 \times 10^{-6}</math></b>	<b>0.36</b>	<b>validated inversion</b>

previously reported in [13]. Finally, we found evidence supporting a number of our predictions by analyzing the 22,375 mapped paired-end sequenced fosmid clones from [15] using a method we developed for comparing structural variants [32]. We found clones supporting an additional 4, 6 and 3 of our inversion predictions in CEU, YRI, and JPT+CHB, respectively. A number of our predicted inversion breakpoints are located in copy number variants, consistent with reports that copy number variants can be associated with inversion polymorphisms [33]. Finally, there are a number of genuinely novel predictions including some with intermediate inversion frequencies (between 30 and 80%) where our methods has the greatest accuracy (Section 3.1). We intend to pursue experimental validation of a subset of these predictions.

We compared our predictions to those reported in [23], who predict 26 inversions in the CEU population, 78 in the YRI population and 72 in the JPT+CHB populations. We note that Bansal et al. [23] trained and applied their model on Phase 1 of the HapMap data, which contains  $\approx 800k$  SNPs compared to the  $\approx 2.5M$  SNPs in Phase 2. In addition, an earlier genome assembly (Build34, July 2003) was used. We remapped the Bansal et al. predictions to Build35 of the human genome using the *LiftOver* tool from the UCSC genome browser. 17 of the 176 total predictions did not map, suggesting that a fraction of their predictions were mis-assemblies in the human reference genome.

Of the remaining predictions, 0/24 of the CEU regions, 0/65 of the JPT+CHB regions and 1/70 of the YRI regions have both breakpoints within 150kb of the breakpoints for a validated inversion in the correct population, as reported by [15]. In contrast, using the same criteria, we correctly identify 3 validated inversions in CEU, 2 in YRI, and 3 in JPT+CHB. Of course, predictions in [23] could be limited by the use of an older genome build and HapMap Phase 1 data, and indeed if we relax the criteria for overlap, we do recover a few additional inversions (Supplemental Table 5). It was not possible to perform a more direct comparison since there is no publicly-available implementation of the method. Moreover, note that it is not surprising that we obtain more predictions than [23] because we are able to make predictions for inversions with much lower frequencies in the population (See Section 3.1). Finally, the method developed by Bansal et al. [23] depended on identifying haplotype blocks with a minimal SNP density. We observed that many inversions reported by [15] were found in regions of the genome with significantly lower than average SNP density, and thus the restricted haplotype blocks might limit the power of their method.

## 4 Discussion

We present a novel method to identify inversion polymorphisms and estimate their frequencies using SNP data. Our model is based on characteristic differences in haplotype frequencies surrounding inversion breakpoints, and significantly outperforms [23] on simulated data. In particular, in contrast to [23], we are able to detect inversions that are not the major allele, with acceptable accuracy at frequencies of 25% or less. In addition, we provide estimates of inversion frequencies that are accurate over a wide range of frequencies. On the HapMap Phase 2 data, we predict 125 inversions in the CEU population, 142 inversions in the YRI population, and 88 in the combined JPT+CHB population with frequencies ranging between 20 and 92 percent. Some of these overlap known inversions, others have evidence supporting their validity, and some are genuinely novel and demand further experimental validation.

The population genetics of human structural variants are only just beginning to be determined. Notably, the model of [12] includes an estimate of the frequency of deletion polymorphisms, but no claims about this accuracy of this estimate were made, and we are unaware of any other methods to estimate frequencies of structural variants. A recent study of copy-number variants [8] revealed approximately 80% of copy number variants had minor allele frequencies greater than 5%. The frequency of inversions in *Drosophila* were shown to be stable in different populations inhabiting similar climatic environments [22] and it would be intriguing to study the frequencies of inversions in different human populations. Our estimates of inversion frequencies will be useful for prioritizing variants for validation. We emphasize that it is not expected that every inversion will leave a population genetic signature strong enough for detection from SNP data because of low frequency, short length, or the numerous other genetic forces (e.g. selection) that can confound such a signature. Nevertheless, the success of our model on many known inversions suggests that differences in haplotype frequencies near inversion breakpoints exist in some cases. Whether these differences are present for the majority of inversion polymorphisms is an open question. Further examination

of known inversions that are not detected by our approach is also warranted. Some of these might be due to low SNP density near inversion breakpoints, but others might be related to the age or frequency of the inversion polymorphism.

While our method has already demonstrated the ability to identify known and novel inversion polymorphisms, there are several directions in which our model could be improved. First, it would be very helpful to calibrate our model with a dataset containing a large number of individuals genotyped for the presence/absence of an inversion, which would provide us with a better understanding of the false positive rate of our method. We are not aware of any such datasets. Second, with larger number of samples, it may be possible to explicitly estimate membership in the forward and inverted populations. Third, we expect that improvements in the definition of haplotype blocks near the inversion breakpoints will lead to increased specificity. Numerous methods have been developed for haplotype block partitioning across a genome [24][34][35][36][37], but these methods find an optimal partition of SNPs into haplotype blocks. For inversion detection, the goal is to find *pairs* of (perhaps suboptimal) block boundaries (corresponding to the inversion breakpoints) that could serve as input to our inversion model. Incorporating uncertainty regarding the boundaries of the haplotype blocks into the probability calculation of the model is a promising avenue to explore. Fourth, extension of our model to genotype data would permit the analysis of datasets where phased haplotypes are not available. Finally, it might be possible to make additional use of other population genetic signals caused by inversions such as the reduction in recombination rates.

Detecting inversions from SNP data is a cost-effective strategy for identification of new variants and for estimating frequencies of known inversions in different populations. Although next generation sequencing technologies are reducing the costs of human genome sequencing, genotyping SNPs will remain significantly cheaper for some time, particularly for large sample sizes. Moreover, abundant SNP data is already available [1][2], and additional information about rare SNPs is coming from projects such as the 1000 Genomes Project [38]. Finally, a SNP-based approach might detect inversions that are missed by paired-end sequencing approaches. In particular, segmental duplications or repetitive sequences at inversion breakpoints might mean that some sequences near breakpoints cannot be mapped to the reference genome [39]. Surveying the landscape of inversions will increase our understanding of the role that these polymorphisms play in shaping human genome variation.

## Acknowledgments

We thank Vikas Bansal for helpful discussions. BJR is supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund.

## References

1. Frazer, K., et al.: A second generation human haplotype map of over 3.1 million SNPs. *Nature* 449, 851–861 (2007)
2. Wellcome Trust Case Control Consortium: Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature* 447, 661–678 (2007)

3. Sharp, A., Cheng, Z., Eichler, E.: Structural variation of the human genome. *Annu. Rev. Genomics Hum. Genet.* 7, 407–442 (2006)
4. Walsh, T., McClellan, J., McCarthy, S., Addington, A., Pierce, S., Cooper, G., Nord, A., Kusenda, M., Malhotra, D., Bhandari, A., Stray, S., Rippey, C., Rocanova, P., Makarov, V., Lakshmi, B., Findling, R., Sikich, L., Stromberg, T., Merriman, B., Gogtay, N., Butler, P., Eckstrand, K., Noory, L., Gochman, P., Long, R., Chen, Z., Davis, S., Baker, C., Eichler, E., Meltzer, P., Nelson, S., Singleton, A., Lee, M., Rapoport, J., King, M., Sebat, J.: Rare structural variants disrupt multiple genes in neurodevelopmental pathways in schizophrenia. *Science* 320, 539–543 (2008)
5. Stefansson, H., Helgason, A., Thorleifsson, G., Steinthorsdottir, V., Masson, G., Barnard, J., Baker, A., Jonasdottir, A., Ingason, A., Gudnadottir, V., Desnica, N., Hicks, A., Gylfason, A., Gudbjartsson, D., Jonsdottir, G., Sainz, J., Agnarsson, K., Birgisdottir, B., Ghosh, S., Olafsdottir, A., Cazier, J., Kristjansson, K., Frigge, M., Thorsteinsson, T., Gulcher, J., Kong, A., Stefansson, K.: A common inversion under selection in Europeans. *Nat. Genet.* 37, 129–137 (2005)
6. Perry, G., Dominy, N., Claw, K., Lee, A., Fiegler, H., Redon, R., Werner, J., Villanea, F., Mountain, J., Misra, R., Carter, N., Lee, C., Stone, A.: Diet and the evolution of human amylase gene copy number variation. *Nat. Genet.* 39, 1256–1260 (2007)
7. Cooper, G., Zerr, T., Kidd, J., Eichler, E., Nickerson, D.: Systematic assessment of copy number variant detection via genome-wide SNP genotyping. *Nat. Genet.* 40, 1199–1203 (2008)
8. McCarroll, S., Kuruvilla, F., Korn, J., Cawley, S., Nemesh, J., Wysoker, A., Shapero, M., de Bakker, P., Maller, J., Kirby, A., Elliott, A., Parkin, M., Hubbell, E., Webster, T., Mei, R., Veitch, J., Collins, P., Handsaker, R., Lincoln, S., Nizzari, M., Blume, J., Jones, K., Rava, R., Daly, M., Gabriel, S., Altshuler, D.: Integrated detection and population-genetic analysis of SNPs and copy number variation. *Nat. Genet.* 40, 1166–1174 (2008)
9. Perry, G., Ben-Dor, A., Tselenko, A., Sampas, N., Rodriguez-Revenga, L., Tran, C., Scheffer, A., Steinfeld, I., Tsang, P., Yamada, N., Park, H., Kim, J., Seo, J., Yakhini, Z., Laderman, S., Bruhn, L., Lee, C.: The fine-scale and complex architecture of human copy-number variation. *Am. J. Hum. Genet.* 82, 685–695 (2008)
10. McCarroll, S., Hadnott, T., Perry, G., Sabeti, P., Zody, M., Barrett, J., Dallaire, S., Gabriel, S., Lee, C., Daly, M., Altshuler, D.: Common deletion polymorphisms in the human genome. *Nat. Genet.* 38, 86–92 (2006)
11. Conrad, D., Andrews, T., Carter, N., Hurles, M., Pritchard, J.: A high-resolution survey of deletion polymorphism in the human genome. *Nat. Genet.* 38, 75–81 (2006)
12. Corona, E., Raphael, B., Eskin, E.: Identification of deletion polymorphisms from haplotypes. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 354–365. Springer, Heidelberg (2007)
13. Tuzun, E., Sharp, A.J., Bailey, J.A., Kaul, R., Morrison, V.A., Pertz, L.M., Haugen, E., Hayden, H., Albertson, D., Pinkel, D., Olson, M.V., Eichler, E.E.: Fine-scale structural variation of the human genome. *Nat. Genet.* 37, 727–732 (2005)
14. Korbel, J.O., Urban, A.E., Affourtit, J.P., Godwin, B., Grubert, F., Simons, J.F., Kim, P.M., Palejev, D., Carriero, N.J., Du, L., Taillon, B.E., Chen, Z., Tanzer, A., Saunders, A.C.E., Chi, J., Yang, F., Carter, N.P., Hurles, M.E., Weissman, S.M., Harkins, T.T., Gerstein, M.B., Egholm, M., Snyder, M.: Paired-end mapping reveals extensive structural variation in the human genome. *Science* 318(5849), 420–426 (2007)

15. Kidd, J.M., Cooper, G.M., Donahue, W.F., Hayden, H.S., Sampas, N., Graves, T., Hansen, N., Teague, B., Alkan, C., Antonacci, F., Haugen, E., Zerr, T., Yamada, N.A., Tsang, P., Newman, T.L., Tüzin, E., Cheng, Z., Ebling, H.M., Tusneem, N., David, R., Gillett, W., Phelps, K.A., Weaver, M., Saranga, D., Brand, A., Tao, W., Gustafson, E., McKernan, K., Chen, L., Malig, M., Smith, J.D., Korn, J.M., McCarroll, S.A., Altshuler, D.A., Peiffer, D.A., Dorschner, M., Stamatoyannopoulos, J., Schwartz, D., Nickerson, D.A., Mullikin, J.C., Wilson, R.K., Bruhn, L., Olson, M.V., Kaul, R., Smith, D.R., Eichler, E.E.: Mapping and sequencing of structural variation from eight human genomes. *Nature* 453, 56–64 (2008)
16. Levy, S., Sutton, G., Ng, P., Feuk, L., Halpern, A., Walenz, B., Axelrod, N., Huang, J., Kirkness, E., Denisov, G., Lin, Y., MacDonald, J., Pang, A., Shago, M., Stockwell, T., Tsiamouri, A., Bafna, V., Bansal, V., Kravitz, S., Busam, D., Beeson, K., McIntosh, T., Remington, K., Abril, J., Gill, J., Borman, J., Rogers, Y., Frazier, M., Scherer, S., Strausberg, R., Venter, J.: The diploid genome sequence of an individual human. *PLoS Biol.* 5, e254 (2007)
17. Wheeler, D., Srinivasan, M., Egholm, M., Shen, Y., Chen, L., McGuire, A., He, W., Chen, Y., Makhijani, V., Roth, G., Gomes, X., Tartaro, K., Niazi, F., Turcotte, C., Irzyk, G., Lupski, J., Chinault, C., Song, X., Liu, Y., Yuan, Y., Nazareth, L., Qin, X., Muzny, D., Margulies, M., Weinstock, G., Gibbs, R., Rothberg, J.: The complete genome of an individual by massively parallel DNA sequencing. *Nature* 452, 872–876 (2008)
18. Iafrate, A., Feuk, L., Rivera, M., Listewnik, M., Donahoe, P., Qi, Y., Scherer, S., Lee, C.: Detection of large-scale variation in the human genome. *Nat. Genet.* 36, 949–951 (2004)
19. Feuk, L., MacDonald, J., Tang, T., Carson, A., Li, M., Rao, G., Khaja, R., Scherer, S.: Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee DNA sequence assemblies. *PLoS Genet.* 1, e56 (2005)
20. Chaisson, M., Raphael, B., Pevzner, P.: Microinversions in mammalian evolution. *Proc. Natl. Acad. Sci. U.S.A.* 103, 19824–19829 (2006)
21. Kirkpatrick, M., Barton, N.: Chromosome inversions, local adaptation and speciation. *Genetics* 173, 419–434 (2006)
22. Hoffmann, A., Sgrò, C., Weeks, A.: Chromosomal inversion polymorphisms and adaptation. *Trends Ecol. Evol. (Amst.)* 19, 482–488 (2004)
23. Bansal, V., Bashir, A., Bafna, V.: Evidence for large inversion polymorphisms in the human genome from HapMap data. *Genome Res.* 17, 219–230 (2007)
24. Patil, N., Berno, A., Hinds, D., Barrett, W., Doshi, J., Hacker, C., Kautzer, C., Lee, D., Marjoribanks, C., McDonough, D., Nguyen, B., Norris, M., Sheehan, J., Shen, N., Stern, D., Stokowski, R., Thomas, D., Trulson, M., Vyas, K., Frazer, K., Fodor, S., Cox, D.: Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* 294, 1719–1723 (2001)
25. Daly, M., Rioux, J., Schaffner, S., Hudson, T., Lander, E.: High-resolution haplotype structure in the human genome. *Nat. Genet.* 29, 229–232 (2001)
26. Pritchard, J., Stephens, M., Donnelly, P.: Inference of population structure using multilocus genotype data. *Genetics* 155, 945–959 (2000)
27. Price, A., Patterson, N., Plenge, R., Weinblatt, M., Shadick, N., Reich, D.: Principal components analysis corrects for stratification in genome-wide association studies. *Nat. Genet.* 38, 904–909 (2006)
28. Sridhar, S., Rao, S., Halperin, E.: An efficient and accurate graph-based approach to detect population substructure. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 503–517. Springer, Heidelberg (2007)
29. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B* 39(1), 1–38 (1977)
30. Falush, D., Stephens, M., Pritchard, J.: Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies. *Genetics* 164, 1567–1587 (2003)

31. Schaffner, S.F., Foo, C., Gabriel, S., Reich, D., Daly, M.J., Altshuler, D.: Calibrating a coalescent simulation of human genome sequence variation. *Genome Res.* 15(11), 1576–1583 (2005)
32. Sindi, S.S., Helman, E., Bashir, A., Raphael, B.J.: A geometric approach for classification and comparison of structural variants. In: Bioinformatics. Proc. ISMB/ECCB 2009 (in press, 2009)
33. Koolen, D., Vissers, L., Pfundt, R., de Leeuw, N., Knight, S., Regan, R., Kooy, R., Reyniers, E., Romano, C., Fichera, M., Schinzel, A., Baumer, A., Anderlid, B., Schoumans, J., Knoers, N., van Kessel, A., Sistermans, E., Veltman, J., Brunner, H., de Vries, B.: A new chromosome 17q21.31 microdeletion syndrome associated with a common inversion polymorphism. *Nat. Genet.* 38, 999–1001 (2006)
34. Zhang, K., Deng, M., Chen, T., Waterman, M., Sun, F.: A dynamic programming algorithm for haplotype block partitioning. *Proc. Natl. Acad. Sci. U.S.A.* 99, 7335–7339 (2002)
35. Anderson, E., Novembre, J.: Finding haplotype block boundaries by using the minimum-description-length principle. *Am. J. Hum. Genet.* 73, 336–354 (2003)
36. Wang, N., Akey, J., Zhang, K., Chakraborty, R., Jin, L.: Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. *Am. J. Hum. Genet.* 71, 1227–1234 (2002)
37. Kimmel, G., Shamir, R.: GERBIL: Genotype resolution and block identification using likelihood. *Proc. Natl. Acad. Sci. U.S.A.* 102, 158–162 (2005)
38. 1000 Genomes Project. Technical report (2008), <http://www.1000genomes.org>
39. Feuk, L., Carson, A.R., Scherer, S.W.: Structural variation in the human genome. *Nat. Rev. Genet.* 7(2), 85–97 (2006)

# On the Relationship between DNA Periodicity and Local Chromatin Structure

Sheila M. Reynolds, Jeff A. Bilmes, and William Stafford Noble

University of Washington,  
Seattle, Washington, USA  
[{sheila,bilmes}@ee.washington.edu](mailto:{sheila,bilmes}@ee.washington.edu)  
[noble@gs.washington.edu](mailto:noble@gs.washington.edu)

**Abstract.** DNA periodicity and its relationship to the formation of nucleosomes has been investigated extensively using autocorrelation and Fourier transform methods. We provide a precise treatment of the mathematical foundation for this type of analysis, and we apply the resulting method to quantify dinucleotide periodicity in several datasets. We begin by demonstrating, via simulation, the sensitivity of our method relative to previous methods. We then provide evidence of pervasive  $\sim 10$  bp periodicity in *S. cerevisiae*, with stronger periodicity in sequences associated with positioned nucleosomes. In human, although repeat-masked sequences do not exhibit significant periodicity on average, we find that experimentally determined nucleosome positions show a periodicity of the AA dinucleotide similar to that found in *S. cerevisiae*. Furthermore, transcription start sites in the human genome are marked by a sharp drop in the 10 bp periodicity of the AA dinucleotide, while occupied CTCF sites are surrounded by a local increase.

**Keywords:** DNA periodicity, nucleosome, dinucleotide, chromatin.

## 1 Introduction

A relationship between DNA sequence periodicity and local curvature of the DNA molecule and hence local chromatin structure has been hypothesized for over 30 years, going back to a “kinky helix” model proposed by Crick and Klug in 1975 [1]. The strongest periodic component in DNA sequences (excluding simple repeats) is generally observed in coding regions and is induced by the codon length and the bias in both amino acid usage and codon usage. This codon effect results in a spike in the spectrum at the normalized frequency  $f = 1/3$ , corresponding to a period of  $T = 3$  bp. A second, weaker periodicity near 10 bp has also been observed in many different organisms, in coding, non-coding and repeat-masked sequences. This periodicity has been linked to the pitch of the DNA helix [2] as well as to the alternation of hydrophobic and hydrophilic amino acids in protein sequences [3]. The earliest evidence of this periodicity was based on just 36 kb of DNA sequence collected from several different eukaryotes as well as certain viruses [2]. Satchwell *et al.* isolated and sequenced 177 chicken

nucleosomes, and observed a 10 bp periodicity of both AA/TT and GG/CC with the minor groove of AA/TT facing predominantly inward toward the histone and GG/CC facing outward [4]. Similar ~10 bp periodicity signals derived from short patterns (up to tetra-nucleotides) have been observed in a wide variety of organisms and datasets, both natural and artificial [5], [6], [7], [8], [9]. These ~10 bp periodicities are strongly correlated with nucleosome positioning, supporting the hypothesis that periodic sequence elements in phase with the DNA helix are related to large-scale bending of the DNA molecule. DNA bendability has been extensively modeled [10], [11], [12], and experimentally measured [13], with a general consensus that poly(dA:dT) tracts are extremely stiff [14], while some short sequences are very flexible, particularly the CA/TG dinucleotide and the CAG/CTG trinucleotide [15], and others such as TA are context-dependent [10].

Most previous approaches to quantifying periodicity in DNA sequences have been based on Fourier techniques which require that the symbolic DNA sequence be translated first into a numerical sequence. The spectrum of the numerical sequence can then be estimated directly using the Fourier transform [7], [16], [17], or by first computing the autocorrelation function [2], [18], [19], [20]. Sequence periodicity has also been studied using a machine-learning based approach which sought to learn a periodic nucleotide pattern in an unsupervised fashion using a cyclic hidden Markov model (HMM) [23], [24]. In the current work, we describe the drawbacks to these previously described approaches to characterizing DNA periodicity and we introduce a mathematically precise approach to evaluating the spectral content of DNA sequences.

With respect to the HMM analysis [24], we report that the scarcity of the CpG dinucleotide appears to be the main contributing factor that led the 10-state cyclic HMM to learn the apparently periodic (not-T)(A/T)(G) pattern. Using a dynamic Bayesian network (DBN) similar to the HMM described by Baldi *et al.* [24], we were able to reproduce this pattern by training on the repeat-masked ENCODE regions of the human genome. However, we found that the same pattern is learned from random DNA generated according to a second order Markov model which reproduces only the single, di- and trinucleotide statistics of human DNA. Additionally we found that the same pattern can be learned by smaller cyclic models constrained to allow period lengths as short as 4-6 bases.

We introduce ACSE (AutoCorrelation Spectral Estimation), a method for quantifying the periodicity of a specified fixed length pattern in DNA, which includes a null model and an estimation of the variance in the estimated spectral amplitude, and we demonstrate that our method is more sensitive to weak evidence of periodicity than previous spectral methods. We have applied our method to a variety of datasets from the human and yeast genomes, including the complete yeast genome, the human ENCODE regions, experimentally identified nucleosome sequences from human and yeast, and human DNA sequences proximal to transcription start sites (TSSs) and CTCF binding sites. We show that nucleosome sequences in both genomes exhibit increased ~10 bp periodicity, especially of the AA/TT dinucleotide. This result contrasts with a recent study that found no evidence of 10-bp periodicity in human nucleosome sequences [25].

In addition, we report a sharp decrease of the 10-bp AA/TT periodicity in the vicinity of TSSs, and a local increase around binding sites of the CTCF insulator protein. These results indicate a strong relationship between DNA periodicity and local chromatin structure and are consistent with the classical statistical positioning theory of nucleosome organization [26], [27], which posits that nucleosomes are stochastically positioned along the genome and are distributed between boundary events that comprise nucleosome-free regions, such as those known to be found at active promoters, insulators or enhancers.

## 2 Methods

Spectral estimation seeks to estimate the frequency content of a time-dependent waveform, representing it as the weighted sum of a family of sinusoids. In this application, the dimension of “time” is genomic position, and in order to use classical spectral estimation techniques, the DNA sequence must first be translated into a numerical sequence. Our method uses a common approach to represent a DNA sequence as a binary sequence: given a DNA sequence  $s$  of length  $S$  and a  $k$ -mer  $m$  of length  $k$ , we create a binary sequence  $b$  of length  $S-k+1$  such that

$$\begin{aligned} b_i = 1 & \text{ if } s_{i:i+k-1} = m \text{ and} \\ b_i = 0 & \text{ otherwise,} \end{aligned} \tag{1}$$

where  $s_{i:i+k-1}$  is the length  $k$  substring in  $s$ , starting at position  $i$ . The number of 1’s in the binary sequence is equal to the total number of occurrences of the  $k$ -mer, including overlapping copies. For example, the 10 base sequence GCAAAGCTAA becomes 001100001 for the dinucleotide AA.

The binary sequence can then be transformed from the “time” domain to the frequency domain in two different ways. Some methods [7], [16], [17] convert directly to the frequency domain via the Fourier transform, and then use the magnitude squared of the resulting complex spectrum. Another approach is to compute the autocorrelation function out to some maximum lag (typically in the range 50 to 500 bases) [2], [18], [19], [20], and then the power spectrum is obtained by a Fourier transform of the autocorrelation function. Under certain conditions, namely that the time-series is wide sense stationary<sup>1</sup> (WSS) [21], and that the two-sided, symmetric autocorrelation function is used, these two approaches are mathematically equivalent. The previously cited approaches have, however, routinely truncated the autocorrelation function which, as we will show, negatively impacts the sensitivity and accuracy of the resulting spectrum.

We prefer the autocorrelation-based approach over the direct Fourier transform approach for three reasons. The first is simply that the autocorrelation provides easily interpretable information about the self-similarity of a sequence,

---

<sup>1</sup> A process is said to be WSS if its first and second moments are time-invariant, resulting in an autocorrelation function that depends only on the time lag. This assumption does not generally hold for DNA segments, *e.g.* those spanning GC-rich and GC-poor regions, but it is not an unreasonable approximation.

irrespective of subsequent transforms. Second, this approach provides a natural way to combine any number of sequences of varying lengths such that each sequence will contribute in proportion to the number of times the  $k$ -mer is present in the sequence. Third, because the variance of the spectral estimates increases as the square of the Fourier transform length [22], the direct transform of several hundred kilobases of sequence requires some form of smoothing in the frequency domain. In our approach, the symmetric autocorrelation function is windowed<sup>2</sup> prior to the Fourier transform in order to reduce the variance in the resulting spectrum and to minimize the spectral leakage from strong periodic signals (such as the 3 bp codon periodicity). It is also useful to extend a time series with zeros prior to computing the Fourier transform in order to increase the resolution in the frequency domain. (This practice is known as *zero-padding*, and although it cannot increase the information content in a signal, it results in a smoothly interpolated spectrum.) In the results presented here, we have generally used a Fourier transform size of 720 which produces spectra with samples at several integer values of  $T$ , thereby reducing the amount of spectral leakage due to strong peaks that may exist at those positions, and with a resolution of 0.14 bp near  $T=10$  bp, which allows us to distinguish fairly subtle variations in periodicity.

In the case of the binary sequence described here, the autocorrelation function has a probabilistic interpretation. Since  $E[b_i]$  (defined in ⑩) is the probability of observing the  $k$ -mer of interest at position  $i$  in the DNA sequence, the autocorrelation function  $R(d)$  is equivalent to the joint probability of observing two 1s in the binary sequence, one at position  $i$  and the second at a relative lag  $d$ :

$$R(d) = R(-d) = E[b_i b_{i-d}] = \Pr[b_i = 1 \text{ and } b_{i-d} = 1] \quad (2)$$

If the binary sequence represents occurrences of the AA dinucleotide, then:

$$\begin{aligned} R_{AA}(d) &= \Pr[b_i = 1 \text{ and } b_{i-d} = 1] \\ &= \Pr[s_i = A, s_{i+1} = A, s_{i-d} = A, s_{i-d+1} = A] \end{aligned} \quad (3)$$

In order to test whether a particular DNA sequence shows evidence of periodicity, a null model of the spectrum is required. We derive an analytic null model by modeling a random DNA sequence with no periodicity in which the nucleotides are independent and identically distributed. The autocorrelation function of the binary sequence representing the AA dinucleotide derived from such a random DNA sequence can be described by the following three equations:

$$R_{AA}(0) = \Pr[A]^2 \quad (4)$$

$$R_{AA}(\pm 1) = \Pr[A]^3 \quad (5)$$

$$R_{AA}(d) = \Pr[A]^4 \quad \text{for all } |d| > 1. \quad (6)$$

A dinucleotide composed of two distinct bases (*e.g.* GC), results in an autocorrelation function with a value of 0 at  $\pm 1$ :

$$R_{GC}(0) = \Pr[G]\Pr[C] \quad (7)$$

---

<sup>2</sup> A point-by-point multiplication of the symmetric autocorrelation function with a symmetric, tapered window such as the Hann window.

$$R_{GC}(\pm 1) = 0 \quad (8)$$

$$R_{GC}(d) = Pr[G]^2 Pr[C]^2 \text{ for all } |d| > 1. \quad (9)$$

In both of these cases, the autocorrelation function for dinucleotide  $m$  can be expressed as the sum of four terms:

$$R_m(d) \propto \delta(d) + U_m \delta(d+1) + U_m \delta(d-1) + V_m \quad (10)$$

where  $U_m$  is positive for a dinucleotide such as AA or CC and negative for a dinucleotide such as TA or GC. ( $U_m$  and  $V_m$  are constants that depend only on  $R(0)$ ,  $R(\pm 1)$  and  $R(d)$ , and  $\delta(d)$  is the Kronecker delta.) The Fourier transform of (10) yields the power spectrum:

$$S_m(k) \propto 1 + 2U_m \cos(2\pi k/K) + V_m \delta(k) \quad (11)$$

where  $K$  is the length of the symmetric autocorrelation function being transformed, and  $k$  is an integer in the range  $[-K/2, +K/2]$ . The normalized frequency  $f$  is defined as  $k/K$  and has a range  $[-1/2, 1/2]$ , and corresponds to periods of length  $T = 1/f$  with a range of  $[2, \infty]$ . The cosine term in the spectrum for a dinucleotide such as AA results in a local maximum at  $f = 0$  and a local minimum at  $f = 1/2$ , meaning more energy at lower frequencies (longer periods) and less energy at higher frequencies (shorter periods). For a dinucleotide such as TA, the reverse is true, with more spectral energy at higher frequencies and less at lower frequencies. An intuitive explanation of this effect is that consecutive AA dinucleotides result in the low frequency binary signal 1111111111 ( $f=0$  and  $T=\infty$ ), while consecutive TA dinucleotides result in the high frequency signal 1010101010 ( $f=1/2$  and  $T=2$ ).

Real DNA sequences are of course not random as described above, and yet the autocorrelation function of the binary sequences generated from real DNA sequences is dominated by the same three components: (i)  $R(0) \gg R(d) \forall d \neq 0$ ; (ii) either  $R(\pm 1) = 0$ , or  $R(\pm 1) > R(d) \forall |d| > 1$ ; and (iii) a relatively flat or slowly-decaying background level with small-amplitude variations for all  $|d| > 1$ . The extent to which the dinucleotide spectrum estimated from real DNA deviates from the random model described above can be estimated by fitting a cosine to the spectrum (11) using a linear least-squares fit in cosine space. This fitted cosine will serve as our null model for the dinucleotide of interest.

Finally, in order to estimate the variance in the spectral estimate, we generated random DNA sequences of varying lengths, applied ACSE to each one and computed the variance in the estimate. The random DNA sequences varied in length from 150 bases to 500 kb. For each length, we generated 10,000 random sequences and computed the estimated ACSE spectrum. A linear regression ( $R^2 > 0.989$ ) of the ratio of the standard deviation to the mean amplitude against five independent variables yielded the following error model:

$$\begin{aligned} \log_{10} \left( \frac{\sigma}{\mu} \right) = & -0.513 \log_{10} N + 0.000791 D_{max} - 0.124 b \\ & - 0.572 p + 0.226 f + 1.042 \end{aligned} \quad (12)$$

where  $N$  is the total sequence length used in estimating the spectrum;  $D_{max}$  is the maximum lag in the autocorrelation function;  $b=1$  if the  $k$ -mer is a repeat (*e.g.* AA, CC) and  $b=0$  if it is not (*e.g.* AT, GC);  $p$  is the average probability of observing the  $k$ -mer ( $0 < p < 1$ ); and  $f$  is the normalized frequency ( $f=1/T$ , and  $f < 1/2$ ). As expected, increasing  $N$  by a factor of four reduces the standard deviation by approximately a factor of 2. The remaining terms are less significant assuming  $N$  is large, *e.g.*  $\geq 100$  kb.

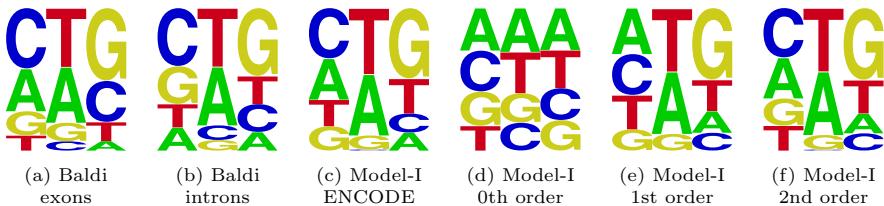
## 3 Results

### 3.1 A Cyclic HMM Identifies Spurious Periodicity

Baldi *et al.* [24] used a cyclic HMM to identify an apparently periodic signal in human genomic DNA. The signal is roughly characterized by a three-letter pattern of the form (not-T)(A/T)(G) and was reported to occur approximately every 10 bp. The HMM that identified this pattern consists of 10 states, each of which emits a single nucleotide. The state-transition matrix is defined such that each state  $i \in \{0, \dots, 9\}$  can transition to one of three states:  $i$  (a self-loop),  $i+1$  (the next consecutive state), or  $i+2$  (skipping over one state), with the addition being modulo-10 to create a cycle. Because of the potential for a state to self-loop, the number of nucleotides emitted prior to transitioning to a different state follows a geometric distribution, and the duration of one complete cycle can range between a minimum length of 5 nucleotides (if every other state is skipped) and an unbounded maximum length. The number of free parameters in this model is 50: 30 for the emission distributions and 20 for the transition probabilities.

Working within the framework of dynamic Bayesian networks (DBN) which include and extend HMMs, we reimplemented the HMM described above, and developed two additional models based on this idea in an attempt to both recreate and expand upon this previous result. In both of our models, we sought to constrain the complete cycle length by allowing only one or two “background” states to self-loop (*i.e.*, emit more than one nucleotide before transitioning to a distinct state), and by not allowing any states to be skipped. These background states are not implemented in the typical manner of a self-looping HMM state; rather, we use the DBN framework to specify a histogram of allowed lengths for these states (for details of a similar DBN with finite length models, see [28]).

Model-I is an eight-state model based on the hypothesis that there is a three nucleotide pattern that occurs roughly every 10 bases, and that there may be a second, complementary pattern that also occurs every 10 bases but out of phase with the first. States 1-3 represent the primary pattern and states 5-6 the complementary pattern; each of these states emits a single nucleotide. States 0 and 4 are “background” states and emit between one and four nucleotides, according to a shared length distribution. Each state  $i$  transitions directly to the next state,  $i+1$  (modulo-8). The length of one complete cycle is therefore constrained between 8 and 14 nucleotides. The total number of free parameters in this model is 27: 24 for the emission distributions and an additional 3 for the length distribution.



**Fig. 1.** Patterns learned by (a) the Baldi HMM trained on exons, (b) the Baldi HMM trained on introns, (c) Model-I trained on the ENCODE repeat-masked data, and Model-I trained on (d) 0th order (e) 1st order and (f) 2nd order random DNA sequences

Model-II is one of a class of  $N$ -state models, also similar to the cyclic HMM, but with additional constraints. In each model, state 0 is the background state and emits 1, 2 or 3 nucleotides, while states 1 through  $N - 1$  each emit a single nucleotide. Again each state  $i$  transitions only to state  $i+1$  (modulo- $N$ ). The total cycle length is thus constrained between  $N$  and  $N+2$  bases. The total number of free parameters in these models is  $3N + 2$  ( $3N$  for the emission distributions, and an additional 2 for the length distribution for state 0).

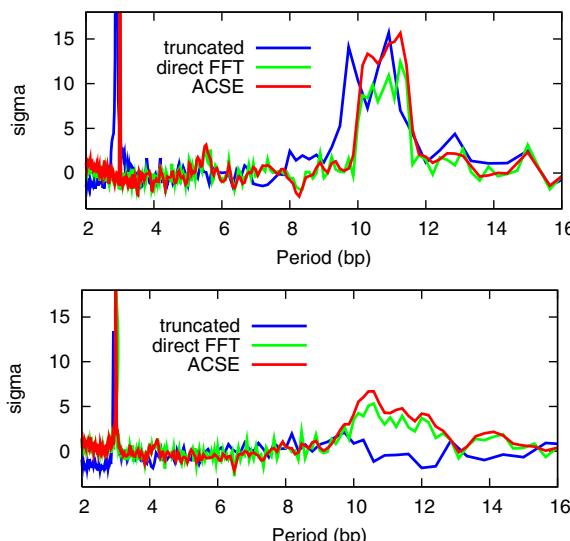
When trained on the repeat-masked ENCODE regions, Model-I learned a three position pattern (Fig. IIc) very similar to the ones reported by Baldi *et al.* (Fig. IIa and b). By “pattern” we mean the learned emission probabilities from any three consecutive states in the model that best matches the (not-T)(A/T)(G) pattern. However, in trying to understand this result with the goal of expanding upon it, we discovered that the pattern could be replicated by training on simulated DNA generated according to a second order Markov model, *i.e.* a generative model (trained on the same repeat-masked ENCODE sequences) in which the probability of each nucleotide depends on the previous two nucleotides. DNA generated according to such a model reproduces the statistics of the training data for single, di- and trinucleotides. The pattern in Fig. IId was learned from 0th order random DNA (independent and identically-distributed nucleotides). As expected, no distinctive pattern was found. However, Figs. II(e) and (f) show the corresponding patterns learned from 1st and 2nd order random DNA. Both patterns closely resemble the one learned from the ENCODE repeat-masked data. Furthermore, using Model-II, we found that the same pattern is learned by cyclic models of different sizes (data not shown), with as few as 4 states (which allows periods of length 4-6 bp), showing again that the perceived periodicity is spurious. Based on these experiments, we conclude that it is the rarity (on average) of the CpG dinucleotide that causes this type of sequential-state, probabilistic model to learn this C(A/T)G pattern. The fact that the learned pattern is stronger (lower entropy) when trained on human DNA than when trained on yeast DNA also supports this, as CpG's are far more rare in human than in yeast.

Despite the fact that these results show that C(A/T)G does not exhibit  $\sim 10$  bp periodicity, there are other indications that this trinucleotide may be related to DNA bending and nucleosome formation. Numerous papers (most recently

[29], for example) describing a relationship between the periodicity of the (not-T)(A/T)G (also referred to as VWG) motif and chromatin structure have cited the work by Baldi *et al.* as a starting point. Perhaps the relationship between C(A/T)G and chromatin structure is instead due to the extreme flexibility of this trinucleotide [14]. It should also be noted that CTG, one of the six codons that code for leucine, is the most commonly used codon in human, while CAG, one of just two codons that code for glutamine, is the third most common – together they account for over 7% of all codons. Overall, in the ENCODE repeat-masked data, CTG/CAG is the fourth most common trinucleotide (after AAA/TTT, AAT/ATT, and AGA/TCT), accounting for 4.4% of all trinucleotides. Intriguingly, repeats of the CTG/CAG trinucleotide are also responsible for several degenerative disorders [30].

### 3.2 Simulation Results Show ACSE Has Greater Sensitivity and Accuracy

Next, to demonstrate the power of ACSE to identify subtle periodic signals, we applied the method to simulated DNA with embedded periodic signals. Our simulation creates a random DNA sequence with equal representation of all four bases, and then adds to the DNA sequence two noisy periodic signals: one at 3 bp, and another with a period length that slowly increases from 10 bp to 11.5 bp across the length of the sequence. Figure 2 compares the output of ACSE with the two previously described methods: the direct Fourier transform approach,

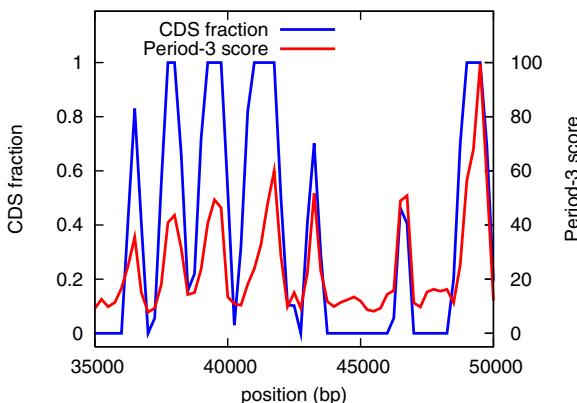


**Fig. 2.** Spectra obtained from the three different methods described: on synthesized DNA containing periodic signals at 3 bp and between 10 and 11 bp (top); and for the AA dinucleotide in *S. cerevisiae* chromosome IV (bottom).

and the truncated autocorrelation approach for the simulated DNA described above, and for *S. cerevisiae* chromosome IV. The output of ACSE and the direct FFT have both had the null model spectrum subtracted, and all three have been normalized such that the regions between 4 and 9 bp and 14 and 24 bp combined have zero mean and unit standard deviation. For the simulated data, ACSE correctly shows a relatively smooth, broad peak between 10.1 and 11.4 bp, while the truncated autocorrelation method predicts two distinct peaks, one at 9.7 bp and the other at 10.9 bp. The truncated autocorrelation approach also errs on the exact position of the 3 bp peak, placing it instead at 2.9 bp. The direct FFT approach (after smoothing) yields a curve comparable to ACSE, although somewhat noisier. For yeast chromosome IV, the truncated autocorrelation method largely misses the evidence of the  $\sim$ 10 bp periodicity of the AA dinucleotide.

### 3.3 Identifying Coding Regions in Yeast

As a proof of concept that dinucleotide periodicity can be used to identify biologically meaningful regions in DNA segments, we developed a *period 3 score*, defined to be the sum over all 10 dinucleotides of the maximum 3 bp periodicity observed on either the forward or reverse strand. We applied this scoring technique to the genome of *S. cerevisiae*, computing the score on short blocks of length 500 bp, and comparing the score to the fraction of bases within that block that lie in coding regions (excluding those coding regions that are annotated as “dubious”). The average Pearson correlation between the period 3 score and the coding fraction was 0.51, with the highest correlation observed for the mitochondrial chromosome (0.85), illustrated in Fig. 3.



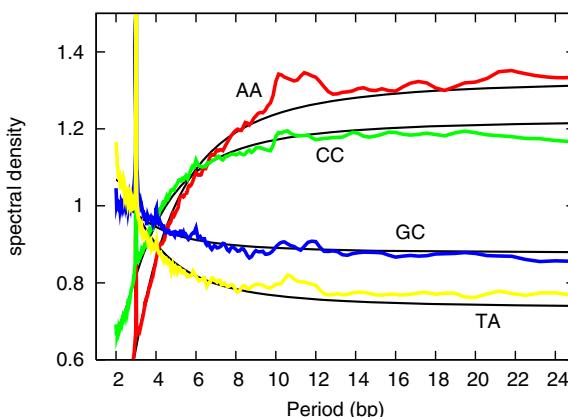
**Fig. 3.** Dinucleotide period 3 score (in red, y-axis on the right) computed on 500 bp blocks, compared to the fraction of the block that is coding (in blue, y-axis on the left). The plot shows a representative 15 kb region of the yeast mitochondrial chromosome.

### 3.4 Genome-Wide Evidence for ~10 bp Periodicity of AA/TT Dinucleotide in *S. cerevisiae*

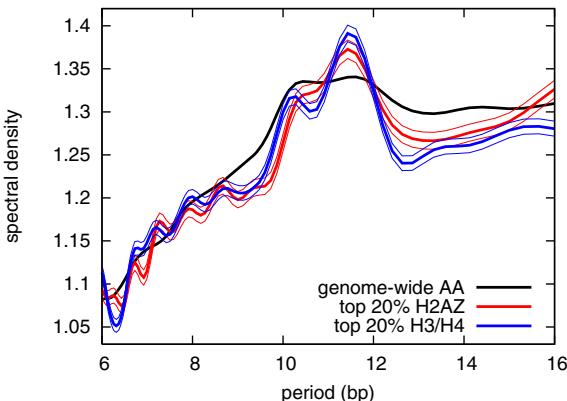
Next, we computed the dinucleotide spectra from genome-wide autocorrelation functions for *S. cerevisiae*, as well as for each chromosome individually. As shown in Fig. 4, the strongest evidence for ~10 bp periodicity is seen when analyzing the AA/TT dinucleotide. Each dinucleotide spectrum in Fig. 4 is accompanied by a null model curve, and deviations from that null model represent increases (above the null curve) or decreases (below) in periodic signal energy relative to what would be expected in a random signal. The other three dinucleotides show much less, if any, evidence of genome-wide 10 bp periodicity. The spectra computed separately for each chromosome show some differences from the genome-wide average (data not shown). Overall, for every chromosome in yeast, we observe a clear periodicity in the 10-11 bp range for the AA/TT dinucleotide.

### 3.5 Positioned Nucleosomes in Yeast Show Increased Periodicity

A comprehensive map of *S. cerevisiae* nucleosomes containing the histone variant H2A.Z in functionally important regions was described by Albert *et al.* [31], including a list of 41,103 nucleosome positions. More recently over one million nucleosomes obtained using antibodies against histones H3 and H4 were sequenced [32], and 54,750 consensus nucleosome locations were identified and assigned confidence scores. After removing nucleosome positions that were within 157 bp of any higher-scoring positions, we were left with 38,356 H2A.Z nucleosome positions and 49,751 H3/H4 positions. (As all nucleosomes contain the H3 and H4 histones, although these two datasets are derived from different experiments, the H2A.Z set is essentially a subset of the H3/H4 set.) Several dinucleotide spectra for these nucleosome core sequences show greater evidence of periodicity between



**Fig. 4.** Dinucleotide spectra averaged across the entire yeast genome (12.3 Mb) for AA, CC, GC and TA, with null model curves for each



**Fig. 5.** AA dinucleotide spectra: genome-wide (black), highest scoring H2AZ (red) and H3/H4 nucleosome positions (blue). Each group of three curves corresponds to the mean and mean  $\pm$  one (estimated) standard deviation.

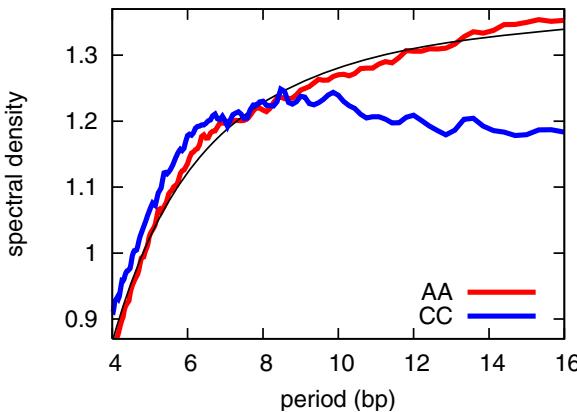
10 and 12 bp than the genome-wide spectra. Using only the highest scoring nucleosome positions, which are thought to be the most stably positioned, yields spectra even more significantly different from the genome-wide background. Figure 5 shows the dinucleotide spectra computed using the highest scoring 20% from each dataset as compared to the genome-wide curve. Both dinucleotides show two distinct peaks, one at 10.4 bp and the other at 11.6 bp. These peaks may indicate different preferred periodicities in different portions of the core nucleosomal sequence, as the double helix underwinds at sites of major-groove bending and overwinds at sites of minor-groove bending [33], [34].

### 3.6 No Evidence of Genome-Wide Periodicity Human

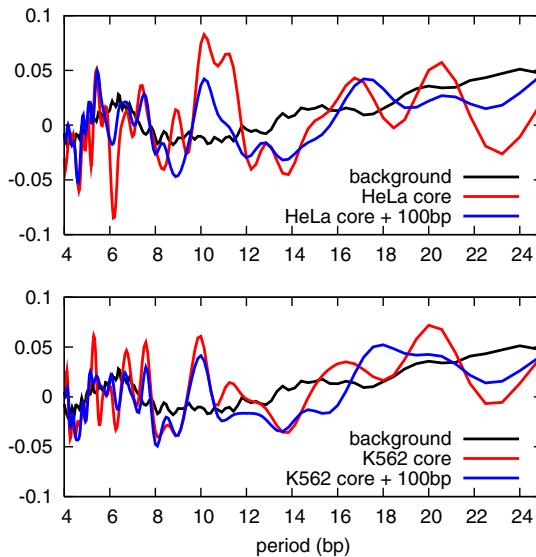
Turning to the human genome, we began by analyzing the repeat-masked ENCODE regions. These regions comprise 1% of the human genome, selected to cover regions of varying gene content and varying concentrations of non-coding conserved elements [35]. Our analysis shows little to no evidence of dinucleotide periodicity when we average across the repeat-masked ENCODE regions. The AA dinucleotide spectrum deviates remarkably little from the null model curve, and none of the other dinucleotides exhibit any strong periodicity near 10 or 12 bp although some deviate significantly in overall shape from the null model. Figure 6 shows the spectra for AA and CC, with the null model for AA.

### 3.7 Human Nucleosome Sequences Show Evidence of Periodicity

In contrast, when we focus on nucleosome sequences, we observe strong evidence of periodicity. Using experimentally determined nucleosome positions in the HOX cluster [25], we extracted 147 bp nucleosome core sequences at 1086 positions based on HeLa data and 1169 positions for K562. The HeLa nucleosome cores show evidence of 10 bp periodicity of the AA dinucleotide, as shown



**Fig. 6.** AA and CC dinucleotide spectra based on the repeat-masked ENCODE regions, with the null model for AA in black



**Fig. 7.** AA dinucleotide spectra, after subtracting null model, for HeLa and K562 nucleosome sequences: genome-wide background, core 147 bp, and core+100 bp

in Fig. 7. The spectra in this pair of plots are shown after subtracting the null model curve, and are compared to the background curve for AA obtained from the repeat-masked ENCODE regions (Fig. 6). The positioned nucleosomes derived from the HeLa data show a significant increase over both the null model and the background between 10 and 12 bp, with local maxima at 10.14 and 11.25 bp. The shape of this peak is similar to the one observed for the AA dinucleotide in yeast (Fig. 4). More strikingly, if we expand the amount of sequence

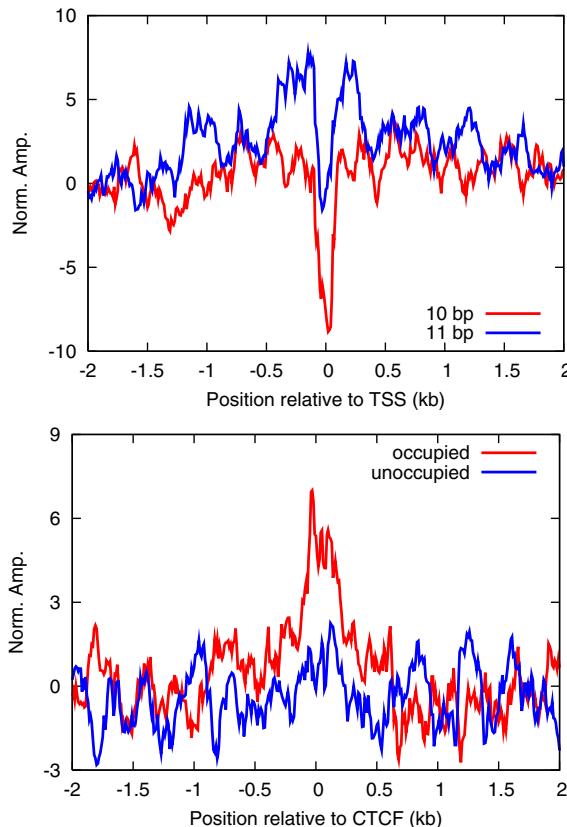
used by an additional 50 bases on either side of the original 147 bp core sequence, the 11.25 bp peak disappears completely while the 10.14 bp peak is somewhat reduced. The spectrum computed from the K562 nucleosome positions is not as striking although there is still a doublet peak with local maxima at 10.00 and 11.25 bp, and again the 11.25 bp peak disappears when the sequence window is widened. Restricting the analysis to the 589 nucleosome positions that are shared between the two subsets (data not shown), the periodicity signal is slightly stronger than the one shown for HeLa in Fig. 7, which may indicate that the remaining K562 positions represent less well positioned nucleosomes or nucleosomes that have been shifted from the most energetically favorable positions.

### 3.8 Decreased Periodicity Near Transcription Start Sites

Using 20,334 transcription start sites from the RefSeq Genes track, we extracted short (150 bp) segments of DNA centered at positions relative to each TSS ranging from 10 kb upstream to 10 kb downstream, and staggered by 10 bases. For each set of sequences, we computed the AA dinucleotide spectrum and extracted the amplitudes at 10 and 11 bp. We then chose the regions between 5 and 10 kb away from the TSS (upstream and downstream) to serve as a background model, and normalized both traces relative to that background. As shown in Fig. 8, shortly before and after the TSS, the 11 bp amplitude is significantly elevated relative to the background while the 10 bp amplitude is relatively flat. Both drop sharply in the immediate vicinity of the TSS. We examined the average AA/TT counts within the same windows to see if this effect could be entirely attributed to local increases or decreases in the counts of these dinucleotides, but this was not the case. The sharp decrease in periodicity near the TSS will inhibit nucleosome formation, while the increase in the 11 bp periodicity upstream and downstream of the TSS will encourage the stable positioning of the canonical -1 and +1 nucleosomes.

### 3.9 Dinucleotide Periodicity Increases Near Occupied CTCF Sites

CTCF is a DNA-binding protein which binds to insulator elements to restrict access to transcriptional promoters. CTCF is believed to play a wide-spread role in gene regulation [36], and a relationship between CTCF binding and strongly positioned nucleosomes has been shown experimentally [37]. We used a list of 6432 *occupied* and the same number of *unoccupied* CTCF binding sites [37] to examine the AA dinucleotide periodicity in the region surrounding CTCF sites using the same approach used above for transcription start sites. The spectral amplitudes at both 10 and 11 bp peak in the vicinity of the occupied CTCF sites, while there is little difference from the background for the unoccupied CTCF sites (Fig. 8b plots only the 11 bp spectral amplitude). The unoccupied CTCF sites represent predicted sites and may include a large number of false positives, while the occupied CTCF sites have been experimentally proven to be true positives. Based on our results, a significant difference between true sites



**Fig. 8.** (a) Average spectral amplitude at 10 and 11 bp for AA/TT dinucleotide relative to the TSS of 20,334 human genes. The amplitude has been normalized to have zero mean and unit standard deviation in the regions 5 to 10 kb from the TSS. (b) Average spectral amplitude for AA/TT dinucleotide at 11 bp for occupied and unoccupied CTCF sites, similarly normalized.

and false sites may be the local curvature of the DNA induced by AA periodicity, despite the local similarity in DNA sequence at the binding site. This increased local curvature may be necessary for CTCF to effectively bind the DNA.

## 4 Discussion

We have described a spectral estimation technique that is both mathematically precise and more sensitive than previously published approaches. Unlike previous methods, ACSE includes an analytic null model as well as a model of the variance in the spectral amplitude estimates. Provided sufficient data, ACSE can identify weak periodic signals in DNA sequences. Previous autocorrelation based approaches have truncated the autocorrelation function, using  $R(d)$  only

for  $d \geq 1$  or  $d \geq 2$ . Truncating the autocorrelation function reduces the sensitivity of the spectral estimation to weak periodic signals and distorts the resulting spectrum. Based on truncated autocorrelation functions some studies have reported not finding any indication of periodicity near 10 bp either in complete human chromosomes [18] or nucleosome core sequences [25].

We have confirmed previously-reported genome-wide evidence of  $\sim$ 10 bp periodicity in *S. cerevisiae*. The spectral pattern shows a doublet peak, with one local maximum near 10 bp and a second near 11 bp. Sequences associated with well positioned nucleosomes show stronger spectral peaks than the genome-wide average, in particular for the 11 bp peak. There is no similar widespread evidence of periodicity in the human genome; however, positioned nucleosomes within the HOX cluster show evidence of the same doublet peak for the nucleosome core sequence. Extending the analysis to include even just an additional 50 bases on either side of the core causes the 11 bp peak to disappear. In the neighborhood of transcription start sites, we observed a sharp decrease in the AA dinucleotide periodicity at both 10 and 11 bp, and it is interesting to note that the 11 bp periodicity is significantly increased both immediately before and after the TSS whereas the 10 bp periodicity is not. This apparent distinction between periodicities of 10 and 11 bases may indicate regions in the nucleosome core where the double helix is alternately under- or overwound [33], [34]. Finally, we find that CTCF sites show a local increase in the strength of the AA periodicity at both 10 and 11 bp, but more strongly at 11 bp. No crystal structure of the CTCF-DNA complex is available, but this evidence of periodicity at CTCF binding sites may indicate that some curvature of the DNA is required for effective binding. Alternatively, these sites may be occupied by nucleosomes when CTCF is not bound.

Overall, our analysis suggests a clear relationship between local periodic patterns in DNA and local chromatin architecture. The lack of a global periodic signal in human, and the correspondence between local periodic signals and functionally significant chromatin events, such as promoters and insulators, supports the classical statistical positioning theory of nucleosome organization.

## References

1. Crick, F.H.C., Klug, A.: Kinky helix. *Nature* 255, 530–533 (1975)
2. Trifonov, E.N., Sussman, J.L.: The pitch of chromatin DNA is reflected in its nucleotide sequence. *Proceedings of the National Academy of Sciences of the United States of America* 77, 3816–3820 (1980)
3. Herzel, H., Trifonov, E.N., Weiss, O., Grosse, I.: Interpreting correlations in biosequences. *Physica A* 249, 449–459 (1998)
4. Satchwell, S.C., Drew, H.R., Travers, A.A.: Sequence periodicities in chicken nucleosome core DNA. *Journal of Molecular Biology* 191, 659–675 (1986)
5. Drew, H.R., Travers, A.A.: DNA bending and its relation to nucleosome positioning. *Journal of Molecular Biology* 186, 773–790 (1985)
6. Shrader, T., Crothers, D.: Artificial nucleosome positioning sequences. *Proceedings of the National Academy of Sciences of the United States of America* 86, 7418–7422 (1989)

7. Widom, J.: Short-range order in two eukaryotic genomes: relation to chromatin structure. *Journal of Molecular Biology* 259, 579–588 (1996)
8. Segal, E., Fondufe-Mittendorf, Y., Chen, L., Thåström, A., Field, Y., Moore, I.K., Wang, J.Z., Widom, J.: A genomic code for nucleosome positioning. *Nature* 44, 772–778 (2006)
9. Fire, A., Alcazar, R., Tan, F.: Unusual DNA structures associated with germline genetic activity in *Caenorhabditis elegans*. *Genetics* 173, 1259–1273 (2006)
10. Packer, M.J., Dauncey, M.P., Hunter, C.A.: Sequence-dependent DNA structure: dinucleotide conformational maps. *Journal of Molecular Biology* 295, 71–83 (2000)
11. Packer, M.J., Dauncey, M.P., Hunter, C.A.: Sequence-dependent DNA structure: tetranucleotide conformational maps. *Journal of Molecular Biology* 295, 85–103 (2000)
12. Beveridge, D.L., Dixit, S.B., Barreiro, G., Thayer, K.M.: Molecular dynamics simulations of DNA curvature and flexibility: helix phasing and premelting. *Biopolymers* 73, 380–403 (2004)
13. Gabrielian, A., Simoncsits, A., Pongor, S.: Distribution of bending propensity in DNA sequences. *FEBS Letters* 393, 125–140 (1996)
14. Bomble, Y.J., Case, D.A.: Multiscale modeling of nucleic acids: insights into DNA flexibility. *Biopolymers* 89, 722–731 (2008)
15. Travers, A.A.: The structural basis of DNA flexibility. *Philosophical Transactions of the Royal Society of London. Series A Mathematical, Physical and Engineering Sciences*. 362, 1423–1438 (2004)
16. Thåström, A., Lowary, P.T., Widlund, H.R., Cao, H., Kubista, M., Widom, J.: Sequence motifs and free energies of selected natural and non-natural nucleosome positioning DNA sequences. *Journal of Molecular Biology* 288, 213–229 (1999)
17. Bailey, K.A., Pereira, S.L., Widom, J., Reeve, J.N.: Archaeal histone selection of nucleosome positioning sequences and the prokaryotic origin of histone-dependent genome evolution. *Journal of Molecular Biology* 303, 25–34 (2000)
18. Holste, D., Grosse, I., Beirer, S., Schieg, P., Herzel, H.: Repeats and correlations in human DNA sequences. *Physical Review E* 67 (2003)
19. Hosid, S., Trifonov, E.N., Bolshoy, A.: Sequence periodicity of *Escherichia coli* is concentrated in intergenic regions. *BMC Molecular Biology* 5, 14 (2004)
20. Schieg, P., Herzel, H.: Periodicities of 10–11 bp as indicators of the supercoiled state of genomic DNA. *Journal of Molecular Biology* 343, 891–901 (2004)
21. Grimmett, G.R., Stirzaker, D.R.: Probability and Random Processes. Oxford University Press, USA (2001)
22. Jenkins, G.M., Watts, D.: Spectral Analysis and Its Applications. Emerson-Adams Press, USA (1998)
23. Baldi, P., Brunak, S., Chauvin, Y., Englebrecht, J., Krogh, A.: Periodic sequence patterns in human exons. In: ISMB, pp. 30–38 (1995)
24. Baldi, P., Brunak, S., Chauvin, Y., Krogh, A.: Naturally occurring nucleosome positioning signals in human exons and introns. *Journal of Molecular Biology* 263, 503–510 (1996)
25. Kharchenko, P.V., Woo, C.J., Tolstorukov, M.Y., Kingston, R.E., Park, P.J.: Nucleosome positioning in human HOX gene clusters. *Genome Research* 18, 1554–1561 (2008)
26. Kornberg, R.: The location of nucleosomes in chromatin: specific or statistical. *Nature* 292, 579–580 (1981)
27. Kornberg, R.D., Lorch, Y.: Twenty-five years of the nucleosome, fundamental particle of the eukaryote chromosome. *Cell* 98, 285–294 (1999)

28. Reynolds, S.M., Käll, L., Bilmes, J.A., Noble, W.S.: Transmembrane topology and signal peptide prediction using dynamic Bayesian networks. *PLoS Computational Biology* 4, 11 (2008)
29. Takasuka, T.E., Cioffi, A., Stein, A.: Sequence information encoded in DNA that influence long-range chromatin structure correlates with human chromosome functions. *PLoS ONE* 3, 7 (2008)
30. Wang, Y.H.: Chromatin structure of repeating CTG/CAG and CGG/CCG sequences in human disease. *Front Bioscience* 122, 4731–4741 (2007)
31. Albert, I., Mavrich, T.N., Tomsho, L.P., Qi, J., Zanton, S.J., Schuster, S.C., Pugh, B.F.: Translational and rotational settings of H2A. Z nucleosomes across the *Saccharomyces cerevisiae* genome. *Nature* 446, 572–576 (2007)
32. Mavrich, T.N., Ioshikhes, I.P., Vinters, B.J., Jiang, C., Tomsho, L.P., Qi, J., Schuster, S.C., Albert, I., Pugh, B.F.: A barrier nucleosome model for statistical positioning of nucleosomes throughout the yeast genome. *Genome Research* 18, 1073–1083 (2008)
33. Richmond, T.J., Davey, C.A.: The structure of DNA in the nucleosome core. *Nature* 423, 145–150 (2007)
34. Tolstorukov, M.Y., Colasanti, A.V., McCandlish, D.M., Olson, W.K., Zhurkin, V.B.: A novel roll-and-slide mechanism of DNA folding in chromatin: implications for nucleosome positioning. *Journal of Molecular Biology* 371, 725–738 (2007)
35. ENCODE Consortium: The ENCODE (ENyclopedia Of DNA Elements) Project. *Science*. 306, 636–640 (2004)
36. Xie, X., Mikkelsen, T.S., Gnirke, A., Lindblad-Toh, K., Kellis, M., Lander, E.S.: Systematic discovery of regulatory motifs in conserved regions of the human genome, including thousands of CTCF insulator sites. *Proceedings of the National Academy of Sciences of the United States of America* 104, 7145–7150 (2007)
37. Fu, Y., Sinha, M., Peterson, C.L., Weng, Z.: The insulator binding protein CTCF positions 20 nucleosomes around its binding sites across the human genome. *PLoS Genetics* 4, 7 (2008)

# Phylogenies without Branch Bounds: Contracting the Short, Pruning the Deep Extended Abstract

Constantinos Daskalakis<sup>1</sup>, Elchanan Mossel<sup>2,\*</sup>, and Sébastien Roch<sup>1</sup>

<sup>1</sup> Microsoft Research

<sup>2</sup> UC Berkeley and Weizmann Institute

**Abstract.** We introduce a new phylogenetic reconstruction algorithm which, unlike most previous rigorous inference techniques, does not rely on assumptions regarding the branch lengths or the depth of the tree. The algorithm returns a forest which is guaranteed to contain all edges that are: 1) sufficiently long and 2) sufficiently close to the leaves. How much of the true tree is recovered depends on the sequence length provided. The algorithm is distance-based and runs in polynomial time.

## 1 Introduction

In Evolutionary Biology, the speciation history of a family of related organisms is generally represented graphically by a *phylogeny*, that is, a tree where the leaves are the observed (extant) species and the branchings indicate speciation events. Traditional approaches for reconstructing phylogenies from homologous molecular sequences extracted from the observed species [1,2] are typically computationally intractable [3,4,5,6,7], statistically inconsistent [8], or they require impractical sequence lengths [9,10,11,12]. Nevertheless, over the past decade, much progress has been made in the design of efficient, fast-converging reconstruction techniques, starting with the seminal work of Erdős et al. [13]. The algorithm in [13], often dubbed the Short Quartet Method (SQM), is based on well-known distance-matrix techniques, that is, it relies on estimates of the evolutionary distance between each pair of species (roughly the time elapsed since their most recent common ancestor). However, unlike other popular distance methods such as Neighbor-Joining [14], the key behind SQM’s performance is that it discards long evolutionary distances, whose estimates from sequence comparisons are known to be statistically unreliable. The algorithm works by first building subtrees of small diameter and, in a second stage, glueing the pieces back together.

The Short Quartet Method is in fact guaranteed to return the correct topology from polynomial-length sequences in polynomial time with high probability. But this appealing theoretical performance comes at a price. The results of [13] rely

---

\* E.M. is supported by an Alfred Sloan fellowship in Mathematics and by NSF grants DMS-0528488, and DMS-0548249 (CAREER) and by ONR grant N0014-07-1-05-06.

critically on biological assumptions which, although reasonable, are often not met in practice (see Section 1.3 for a formal statement):

- a) [*Dense Sampling of Species*] The observed species are “closely related.” In particular, there are no exceptionally long branches in the phylogeny.
- b) [*Absence of Polytomies*] The phylogeny is bifurcating. In fact, Erdős et al. assume that speciation events are sufficiently far apart to be easily distinguished.

The point of a) is that it implies a natural bound on the depth of the tree which in turn ensures that enough information about the deep parts of the tree diffuses to the leaves. As for Assumption b), it guarantees that a clear signal can be extracted from each branch of the phylogeny. It is obvious—at least intuitively—that assumptions such as a) and b) are necessary to secure the type of results Erdős et al. obtain: *the guaranteed reconstruction of the full phylogeny*. Hence, to improve over SQM and obtain strong guarantees under more general conditions, one has to relax this last requirement.

In this paper, we design an algorithm which provides strong reconstruction guarantees without Assumptions a) and b). We show that our algorithm is guaranteed to recover a *forest* containing all edges that are “sufficiently long” and “sufficiently close” to the leaves. In fact, we allow a trade-off between the resolution of short branches and the depth of the reconstructed forest, a feature of potential practical interest. Also, we guarantee that our reconstructed forest has the desirable property of being *disjoint* (although the presence of short edges leads us to allow deep intersections of very short branches between the subtrees). Moreover, our algorithm does not require the knowledge of a priori bounds on branch lengths or tree depth. Finally if Assumptions a) and b) are satisfied, we recover the whole phylogeny and provide an alternative to the algorithm of Erdős et al.

Precise statements are given in Section 1.2. For a full comparison to related work see also Section 1.3. For a lack of space, the proofs of our results are not included in this extended abstract. But they can be found in the full version of the paper [15].

## 1.1 What Can We Hope to Reconstruct?

Well-known identifiability results [16] guarantee that phylogenies—or at least their idealized stochastic models—can be fully reconstructed given enough data at the leaves. However, molecular data gathered from current species are in essence limited, which begs the question: *How much of tree can we really hope to reconstruct?* We pointed out above two important sources of difficulties: short branches produce a low signal that may be hard to detect; similarly, untangling the deep parts of the tree presents challenges that are well documented (see, e.g., [17, 18]). Note that these issues are fundamentally “information-theoretic” and that they affect all reconstruction methods.

To avoid these difficulties, most *rigorous* methods impose restrictions on the length of the branches and/or the depth of the tree, which may be unsatisfactory from a practical perspective. On the other hand, commonly used methods

in *practice*, such as likelihood and bayesian methods, typically produce several candidate trees as well as confidence estimates. But theoretical guarantees on the quality of such outputs are hard to obtain.

Here, we seek to give strong reconstruction guarantees without any assumption on the true phylogeny. Our goal is to recover, for any given amount of data, as much of the tree as can rigorously be reconstructed with high confidence. Since the full phylogeny may not always be recoverable, we are led to a more flexible solution concept: we output a *contracted subforest* of the true phylogeny. That is, we output a forest containing all branches that are “sufficiently long” and “sufficiently recent”; note that “sufficiently” here is determined (information-theoretically) by the size of the data (usually in terms of sequence length). In the remainder of this section we formalize this notion.

*The input.* Formally, a phylogeny is a *weighted, multifurcating tree* on a set of leaves  $L$ , which we identify with the labels  $[n] = \{1, \dots, n\}$ . We denote a phylogeny by  $T = (V, E; L, \lambda)$ . Here  $V$  and  $E$  are respectively the vertex and edge set of the tree, and  $\lambda : E \rightarrow (0, +\infty)$  assigns a weight to each edge (the branch length). We assume that all internal vertices  $V - L$  have degree at least 3.

A phylogeny is naturally equipped with a so-called additive metric on the leaves  $d : L \times L \rightarrow (0, +\infty]$  defined as follows

$$\forall u, v \in L, \quad d(u, v) = \sum_{e \in P_T(u, v)} \lambda_e,$$

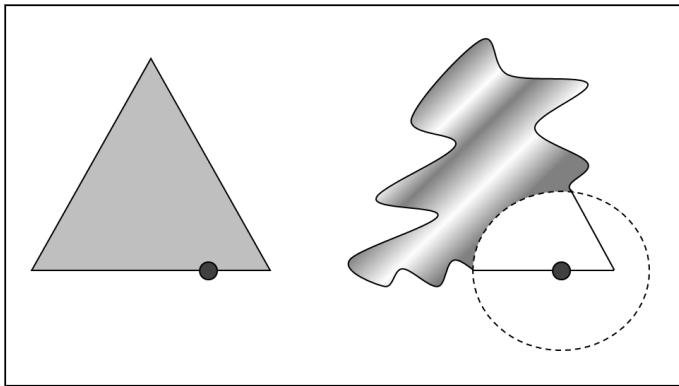
where  $P_T(u, v)$  is the set of edges on the path between  $u$  and  $v$  in  $T$ . Often  $d(u, v)$  is referred to as the “evolutionary distance” between species  $u$  and  $v$ . Since under the assumptions above there is a one-to-one correspondence between  $d$  and  $\lambda$ , we write either  $T = (V, E; L, d)$  or  $T = (V, E; L, \lambda)$ . We also sometimes use the natural extension of  $d$  to the internal vertices of  $T$ . We denote by  $\mathcal{T}$  the set of all phylogenies on any number of leaves.

It is well-known that given an additive metric  $d$  one can reconstruct the corresponding phylogeny  $T$ . However, in practice, one can only derive an *estimate*  $\hat{d}$  of  $d$ , the accuracy of which depends on the sequence length. (This estimate is known in the literature as the “distance matrix”.) Our goal in this paper is to reconstruct a phylogeny—or as much of it as possible—from this “distorted” version of its additive metric. A well-known property of  $\hat{d}$  is that estimates of long distances are unreliable. The following definition formalizes this phenomenon. See Figure 1 for an illustration.

**Definition 1 (Distorted Metric [19,20]).** Let  $T = (V, E; L, d)$  be a phylogeny and let  $\tau, M > 0$ . We say that  $\hat{d} : L \times L \rightarrow (0, +\infty]$  is a  $(\tau, M)$ -distorted metric for  $T$  or a  $(\tau, M)$ -distortion of  $d$  if:

1. [Symmetry] For all  $u, v \in L$ ,  $\hat{d}$  is symmetric, that is,

$$\hat{d}(u, v) = \hat{d}(v, u);$$



**Fig. 1.** The effect of distance distortion from the perspective of a leaf. On the left hand side is the true phylogeny. On the right hand side, only distances within a certain radius represent accurately the metric underlying the phylogeny.

2. [Distortion]  $\hat{d}$  is accurate on “short” distances, that is, for all  $u, v \in L$ , if either  $d(u, v) < M + \tau$  or  $\hat{d}(u, v) < M + \tau$  then

$$|d(u, v) - \hat{d}(u, v)| < \tau.$$

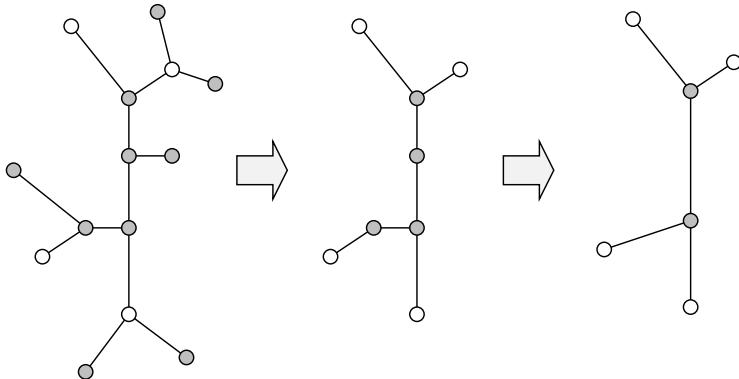
In phylogenetic reconstruction, a distorted metric is naturally derived from samples of a Markov model on a tree—a common model of DNA sequence evolution used in Biology. (See [15] for details.) In the remainder of this paper, we assume that we are given a  $(\tau, M)$ -distortion  $\hat{d}$  of an additive metric  $d$  and we seek to recover the underlying phylogeny  $T$ .

*Contraction and pruning.* Given only a  $(\tau, M)$ -distorted metric, it is clear that the best we can hope for in general is to reconstruct a forest containing those edges of  $T$  that are “sufficiently close” to the leaves. Indeed, note that two phylogenies that are identical up to depth  $M$  from the leaves, but are otherwise different, can give rise to the same distorted metric. Moreover, since we do not assume that edges are longer than the accuracy  $\tau$ , some edges may be too short to be reconstructed and, as we mentioned before, we allow ourselves to instead contract them. Hence, we are led to consider subforests of the true phylogeny where deep edges are *pruned* and short edges are *contracted*.

To formalize this idea we need a few definitions. Let us first describe what we mean by a *subforest* of a phylogeny  $T = (V, E; L, d)$ . Given a set of vertices  $V' \subseteq V$ , the *subtree of  $T$  restricted to  $V'$*  is the tree obtained 1) by keeping only nodes and edges on paths between vertices in  $V'$  and then 2) by contracting all paths composed of vertices of degree 2, except the nodes in  $V'$ . See Figure 2 for an example. We denote this tree by  $T|_{V'}$ . We typically take  $V' \subseteq L$ . A *subforest* of  $T$  is defined to be a collection of restricted subtrees of  $T$ .

We also need a notion of depth. Given an edge  $e \in E$ , the *chord depth* of  $e$  is the length of the shortest path between two leaves on which  $e$  lies. That is,

$$\Delta_c(e) = \min \{d(u, v) : u, v \in L, e \in P_T(u, v)\}.$$



**Fig. 2.** Restricting the top tree to its white nodes

We define the *chord depth of a tree*  $T$  to be the maximum chord depth in  $T$

$$\Delta_c(T) = \max \{ \Delta_c(e) : e \in E \}.$$

**Definition 2 (Contracted Subforest).** Let  $T = (V, E; L, d)$  be a phylogeny. Fix  $M > 0$ . Let  $\{L_1, \dots, L_q\}$  be the natural partition of the leaf set  $L$  obtained by removing all edges  $e \in E$  such that  $\Delta_c(e) \geq M$ . We define the  $M$ -pruned subforest of  $T$  to be the forest  $F_M(T) = (V_M, E_M)$  consisting of the trees  $\{T|_{L_1}, \dots, T|_{L_q}\}$ . The metric  $d$  is extended as follows for all  $u, v \in L$ ,

$$d_M(u, v) = \begin{cases} d(u, v), & \text{if } u, v \text{ are in the same subtree of } F_M(T), \\ +\infty, & \text{o.w.} \end{cases}$$

Similarly, we define an extension  $\lambda_M$  of  $\lambda$ .

Now, given also  $\tau > 0$ , the  $\tau$ -contracted  $M$ -pruned subforest of  $T$  is the forest  $F_{\tau, M}(T) = (V_{\tau, M}, E_{\tau, M})$  obtained from  $F_M(T)$  by contracting edges  $e \in E_M$  of weight  $\lambda_M(e) < \tau$ .

*Path-disjointness.* We require that the trees of our reconstructed forest are “not intersecting”. This is a natural condition to impose in order to obtain a meaningful reconstruction: we want to avoid as much as possible that the same branches appear in many subtrees. In fact, we can only guarantee approximate *path-disjointness* as defined below.

We first need a notion of depth for vertices. For a phylogeny  $T = (V, E; L, d)$  and a vertex  $x \in V$ , the *vertex depth* of  $x$  is the length of the shortest path between  $x$  and the set of leaves. That is,

$$\Delta_v(x) = \min \{d(u, x) : u \in L\}.$$

Given two leaves  $u, v$  of  $T$ , we denote by  $\tilde{P}_T(u, v)$  the set of vertices on the path between  $u$  and  $v$  in  $T$ .

We say that two trees are  $(\tau, M)$ -path disjoint if they are “almost disjoint” in the sense that they only share edges (if any) that are “deep” (endpoints have vertex depth at least  $M/2$ ) and “short” (length at most  $\tau$ ). More formally:

**Definition 3 (Approximate Path-Disjointness).** Let  $T = (V, E; L, d)$  be a phylogeny. Two subtrees  $T_1, T_2$  of  $T$  restricted respectively to  $L_1, L_2 \subseteq L$  are  $(\tau, M)$ -path-disjoint if  $L_1 \cap L_2 = \emptyset$  and for all pairs of leaves  $u_1, v_1 \in L_1$  and  $u_2, v_2 \in L_2$  such that

$$\tilde{P}_T(u_1, v_1) \cap \tilde{P}_T(u_2, v_2) \neq \emptyset,$$

we have:

$$\min\{\Delta_v(x) : x \in \tilde{P}_T(u_1, v_1) \cap \tilde{P}_T(u_2, v_2)\} \geq \frac{1}{2}M,$$

and, if further  $P_T(u_1, v_1) \cap P_T(u_2, v_2) \neq \emptyset$ ,

$$\max\{\lambda_e : e \in P_T(u_1, v_1) \cap P_T(u_2, v_2)\} \leq \tau.$$

More generally, a collection of restricted subtrees  $T_1, \dots, T_q$  of  $T$  are  $(\tau, M)$ -path-disjoint if they are pairwise  $(\tau, M)$ -path-disjoint. In the case  $\tau = 0$ , we simply say that the subtrees are path-disjoint.

## 1.2 Main Result and Corollaries

*Main result.* Our main result is an algorithm which, given a  $(\tau, M)$ -distorted metric, reconstructs a contracted subforest (of the true phylogeny) whose trees are approximately path-disjoint. Typically,  $M$  is much larger than  $\tau$ . In that case, we reconstruct a subforest of  $T$  with chord depth  $\approx \frac{1}{2}M$  which includes all edges of length at least  $4\tau$ . The reconstructed subtrees may “overlap” on edges of length at most  $2\tau$  at vertex depth  $\approx \frac{1}{4}M$ . In an upcoming journal version of the paper, we show that these parameters are essentially optimal. The algorithm runs in polynomial time. An implementation of the algorithm with low running time will be given in the journal version.

More precisely, we show:

**Theorem 1 (Main Result).** Let  $\tau$  and  $M$  be monotone functions of  $n$  with  $M > 3\tau$ . Let  $m > 3\tau$  be such that

$$m < \frac{1}{2}[M - 3\tau],$$

for all  $n$ . Then, there is an algorithm  $\mathcal{A}$  such that, for all phylogenies  $T = (V, E; L, d)$  in  $\mathcal{T}$  and all  $(\tau, M)$ -distortions  $\hat{d}$  of  $d$ ,  $\mathcal{A}$  applied to  $\hat{d}$  satisfies the following:

1. [Approximate Path Disjointness]  $\mathcal{A}$  returns a  $(2\tau, m - 3\tau)$ -path-disjoint sub-forest  $\hat{F}$  of  $T$ ;
2. [Depth Guarantee] The forest  $\hat{F}$  is a refinement of  $F_{4\tau, m - \tau}(T)$ ;
3. [Polynomial Time]  $\mathcal{A}$  runs in time polynomial in  $n, \log M, \log \tau$ .

We give below a few important special cases of Theorem 1. The proof of Theorem 1 can be found in the full version of the paper [15].

*Tree case.* When the amount of data is sufficient to produce a distorted metric with  $M = \Omega(\Delta_c(T))$ , we get a single component, that is, the full tree (up to those edges that are contracted).

**Corollary 1 (Tree Case).** *Let  $\tau > 0$  and  $M > 2\Delta_c(T) + 5\tau$ . Then, choosing  $m > \Delta_c(T) + \tau$  guarantees that the reconstructed forest is composed of only a tree.*

In the case of “dense” phylogenies,  $M = \Omega(\log n)$  is sufficient to reconstruct the full tree.

**Definition 4 (Dense Phylogenies (see e.g. [13])).** *We say that a collection of phylogenies  $T'$  is dense if there is a  $0 < g < +\infty$  (independent of  $n$ ) such that for all  $T = (V, E; L, \lambda) \in T'$  we have*

$$\forall e \in E, \lambda_e \leq g. \quad (1)$$

We denote by  $T_g$  the set of phylogenies satisfying (1).

**Corollary 2 (Dense Case).** *In the case of dense phylogenies,  $M = \Omega(\log n)$  suffices to guarantee the reconstruction of the full tree, up to contracted edges.*

*Absolute variant.* All rigorous algorithms prior to our work (see Section [13]) require knowledge of either the tree depth or bounds on the edge lengths to give strong reconstruction guarantees. This is not satisfactory from a practical point of view. Here given only the sequence length we provide explicit guarantees. The following result assumes that the distorted metric is derived from a Markov model on a tree. (See [15] for details.)

**Corollary 3 (Absolute Variant).** *Given a number of samples  $k = \Omega(\log n)$  from a Markov model on a tree and a chosen level of contraction  $\varepsilon > 0$  (small), one can choose  $\tau, M, m$  so that  $\mathcal{A}$  is guaranteed to return a (contracted) subforest of  $T$  containing  $F_{\varepsilon, M'}(T)$  with probability  $1 - o(1)$ , where  $M' = \Omega_\varepsilon(\log k - \log \log n)$ .*

*Complete resolution.* Finally we remark that, if we further assume that all branch lengths are bounded from below by a constant, then by choosing  $\tau$  accordingly a non-contracted forest is returned. In particular, we also recover the results of [13].

### 1.3 Related Work

Under a Markov model of evolution, the Short Quartet Method (SQM) of Erdős et al. [13] is guaranteed to recover the full phylogeny as long as the number of samples  $k$  satisfies

$$k > cf^{-2}e^{c'g\Delta_c(T)} \log n,$$

for constants  $c, c' > 0$ , where  $f$  and  $g$  are respectively lower and upper bounds on the branch lengths possibly depending on  $n$ . For instance, if  $f$  and  $g$  are constants

the sequence length needed for complete reconstruction depends polynomially in the number of species.

Mossel [19] developed a framework that allows the reconstruction of a well-behaved *forest* when sequences are too short to guarantee a complete reconstruction. More precisely, edges which are too deep (in the sense of appearing only on paths between species whose distances are not accurately known) are *pruned* from the final reconstruction. At a high level, Mossel’s Distorted Metric Method (DMM) (implicit in [19]), works in a fashion similar to SQM—except for a pre-processing phase that clusters together sufficiently related species. However, for DMM to work, lower bounds on the branch lengths are required and, moreover, these must be known by the algorithm. Following up on [19], Daskalakis et al. [21] gave a variant of DMM that runs without knowledge of a priori bounds on the branch lengths or the tree depth—making their variant somewhat more practical. However, like DMM, the algorithm in [21] does not deal properly with short edges: any part of the tree containing a short edge cannot be reconstructed by the algorithm (even though there may be adjacent edges that are in fact reconstructible). Therefore, in the presence of short edges no guarantee can be given about the depth of the reconstructed forest.

Recently Gronau et al. [22] eliminated the need for a lower bound on the branch length by *contracting* edges whose length is below a user-defined threshold. Their solution uses a Directional Oracle (DO) which closes in on the location of a leaf to be added and, in the process, contracts regions that do not provide a reliable directional signal. Although the DO algorithm does not use an explicit bound on the depth of the tree, their *reconstruction guarantee* requires such a bound, similarly to [13]. In particular, Gronau et al. leave open the question of giving a forest-building version of their algorithm. Moreover, the sequence length in [22] depends exponentially on what the authors call the  $\varepsilon$ -diameter of the tree—essentially, the maximum diameter of the contracted regions. It is natural to conjecture that an optimal result should not depend on this parameter.

For further related work on efficient phylogeny reconstruction, see also [23, 24, 25, 26, 20, 27, 28].

#### 1.4 Discussion of the Results

The following table summarizes the current status as discussed in the previous sections.

As the table emphasizes, our overarching goal is to design an algorithm with good reconstruction guarantees in the presence of both short and deep edges, whose execution does not rely on a priori bounds on branch lengths. Unfortunately, given the combinatorial complexity of Mossel’s forest-building algorithm, it is not straightforward to provide the extra flexibility of edge contraction in this framework. The novelty in our work is twofold:

- *Solution Concept:* A basic complication is that, in some sense, contraction and pruning interfere with each other. Indeed, the presence of unresolved

	<i>Execution</i>	<i>Guarantees</i>	
		Short edges OK	Deep edges OK
[13]			
[19]			✓
[21]	✓		✓
[22]	✓	✓	
<b>Our method</b>	✓	✓	✓

branches at the boundary of partially reconstructed subtrees creates the possibility of deep “undetectable” intersections. This pitfall seems to be unavoidable. One of our main contributions is to introduce the notion of approximate disjointness, which allows short and deep intersections between subtrees of the reconstructed forest. This suitable solution concept leads to a quite simple algorithm with reasonable guarantees. Moreover, the flexibility in our definition allows us to recover all previously known results as special cases.

- *Algorithmic Technique:* A natural approach to forest building used in [19][21] proceeds along the following three steps:

1. first, leaves are grouped into clusters for which all pairwise distances are accurately known (the *small* clusters);
2. by definition, the local topologies on the small clusters can be trivially reconstructed [29];
3. finally, the local topologies that intersect in the true tree are “glued” together to get a forest (the resulting forest partitions the leaves into *large* clusters).

This last step involves non-trivial combinatorial considerations. We have found that further allowing contracted edges makes this process somewhat unmanageable. Instead we use a different approach relying on simple metric arguments. In particular, we *directly* partition the leaves into large clusters, whose underlying subtrees are approximately disjoint, and provide a new straightforward method to reconstruct these subtrees.

In addition, we obtain as special cases the results discussed in Section 1.3. In particular, if there are no short edges, we recover the results of [19] and [21], where a path-disjoint forest is returned (by taking  $\tau$  equal to half the lower bound on the branch lengths in Theorem 1). If furthermore there is an upper bound on the branch lengths, we recover the results of [13] (Corollary 2). Finally, if we keep the upper bound on the edge lengths, but drop the lower bound, we recover the results of [22] (Corollary 1). In fact, we eliminate the dependence on

the  $\varepsilon$ -diameter.<sup>1</sup> Further, unlike [22], we allow an arbitrary number of states, an extension—it should be noted—that follows easily from [23] and [19].

## 2 Algorithm

The outline of the algorithm follows. There are three main phases, which are explained in detail after the outline. The input to the algorithm is a  $(\tau, M)$ -distorted metric  $\hat{d}$  on  $n$  leaves. In particular, we assume that the values  $\tau$  and  $M$  are known to the algorithm (but see also Corollary 3). Let  $m$  be as in Theorem 1. We denote the true tree by  $T = (V, E; L, d)$ . The details of the subroutines MINI CONTRACTOR and EXTENDER are detailed in Figures 4 and 6 (see also their high level description below). For lack of space, the proof of correctness of the algorithm can be found in [15].

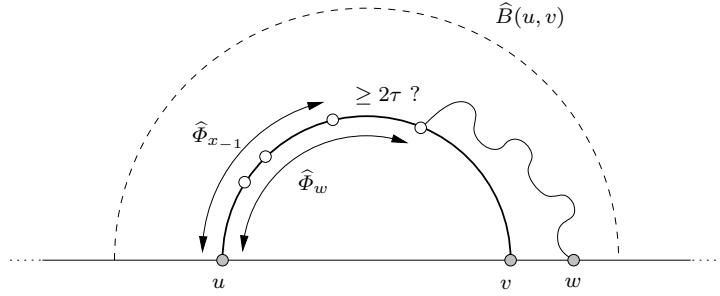
- **Pre-Processing: Leaf Clustering.** Build the distorted clustering graph  $\hat{H}_m = (\hat{V}_m, \hat{E}_m)$  where  $\hat{V}_m = [n]$  and  $(u, v) \in \hat{E}_m \iff \hat{d}(u, v) < m$ ; compute the connected components  $\{\hat{h}_m^{(i)} = (\hat{v}_m^{(i)}, \hat{e}_m^{(i)})\}_{i=1}^q$  of  $\hat{H}_m$ ;
- **Main Loop.** For all components  $i = 1, \dots, q$ :
  - For all pairs of leaves  $u, v \in \hat{v}_m^{(i)}$  such that  $(u, v) \in \hat{E}_m$ :
    - \* **Mini Reconstruction.** Compute
$$\{\psi_j(u, v)\}_{j=1}^{r(u,v)} := \text{MINI CONTRACTOR}(\hat{h}_m^{(i)}; u, v);$$
  - \* **Bipartition Extension.** Compute
$$\{\bar{\psi}_j(u, v)\}_{j=1}^{r(u,v)} := \text{EXTENDER}(\hat{h}_m^{(i)}, \{\psi_j(u, v)\}_{j=1}^{r(u,v)}; u, v);$$
  - Deduce the tree  $\hat{T}^{(i)}$  from  $\{\bar{\psi}_j(u, v)\}_{j=1}^{r(u,v)}$ ;
- **Output.** Return the resulting forest  $\hat{F}$ .

*Pre-processing: Leaf clustering.* As mentioned before, given a  $(\tau, M)$ -distortion we cannot hope to reconstruct edges that are too deep inside the tree. This results in the reconstruction of a *forest*. Therefore, the first phase of the algorithm is to determine the “support” of this forest. We proceed as follows. Consider the following graph on  $L$ .

**Definition 5 (Clustering Graph).** Let  $\tau \leq M' \leq M - \tau$ . The distorted clustering graph with parameter  $M'$ , denoted  $\hat{H}_{M'} = (\hat{V}_{M'}, \hat{E}_{M'})$ , is the following

---

<sup>1</sup> After the results of the current paper were posted on the arXiv, we were informed by S. Moran that, in parallel to our work, the authors of [22] have improved on their previous results: the dependence on the  $\varepsilon$ -diameter has been removed. A preprint of this work is currently available on the authors’ website. Note however that this new, independent work does not deal with deep edges and still makes assumptions similar to [13] restricting the depth of the generating tree.



**Fig. 3.** Illustration of routine MINI CONTRACTOR. See Figure 4 for notation.

graph: the vertices  $\widehat{V}_{M'}$  are the leaves  $L$  of  $T$ ; two leaves  $u, v \in L$  are connected by an edge  $e \in \widehat{E}_{M'}$  if

$$\hat{d}(u, v) < M'. \quad (2)$$

Note that this is an undirected graph because  $\hat{d}$  is symmetric. Similarly, we define the clustering graph with parameter  $M'$ ,  $H_{M'} = (V_{M'}, E_{M'})$ , where we use  $d$  instead  $\hat{d}$  in (2).

The first phase of the algorithm consists in building the graph  $\widehat{H}_m$  from  $\hat{d}$ . We then compute the connected components of  $\widehat{H}_m$  which we denote  $\{\hat{h}_m^{(i)}\}_{i=1}^q$ . In the next two phases, we build a tree on each of these components.

*Building the components I: Mini-reconstruction problem.* Fix a component  $\hat{h}_m^{(i)}$  of  $\widehat{H}_m$ . In this and the next phase, we seek to reconstruct a contracted tree on  $\hat{h}_m^{(i)}$ . Denote by  $T^{(i)}$  the true tree  $T$  restricted to the leaves in  $\hat{h}_m^{(i)}$ . First, we find all edges of  $T^{(i)}$  that are “sufficiently long” and lie on “sufficiently short” paths. More precisely, we consider all pairs of leaves  $u, v$  connected by an edge in  $\hat{h}_m^{(i)}$ , that is, leaves within distorted distance  $m$ . For each such pair, say  $u, v$ , the *mini reconstruction problem* consists in finding all edges in  $P_{T^{(i)}}(u, v)$  that have length longer than  $\lambda_e \geq 4\tau$ . To do this using the distortion  $\hat{d}$ , we first consider a ball  $\widehat{B}(u, v)$  of all nodes within distorted distance  $M$  of  $u$  and  $v$ , that is,

$$\widehat{B}(u, v) = \left\{ w \in \hat{h}_m^{(i)} : \hat{d}(u, w) \vee \hat{d}(v, w) < M \right\},$$

where  $a \vee b$  is the maximum of  $a$  and  $b$ .— The point of using this ball is that we can then guarantee that each edge in  $P_{T^{(i)}}(u, v)$  is “witnessed” by a quartet (i.e., a 4-tuple of leaves) in  $\widehat{B}(u, v)$  in the following sense: let  $(x_1, x_2)$  be an edge in  $P_{T^{(i)}}(u, v)$  and let  $(x_j, y_j)$ ,  $j = 1, 2$ , be an edge adjacent to  $x_j$  that is *not* in  $P_{T^{(i)}}(u, v)$ ; for  $j = 1, 2$  let  $L_{x_j \rightarrow y_j}^{(i)}$  be the leaves reachable from  $y_j$  using paths not including  $x_j$ ; then we will show that  $L_{x_j \rightarrow y_j}^{(i)} \cap \widehat{B}(u, v) \neq \emptyset$  for  $j = 1, 2$ . In other words, there is enough information in  $\widehat{B}(u, v)$  to reconstruct all edges in

**Algorithm MINI CONTRACTOR**

*Input:* Component  $\hat{h}_m^{(i)}$ ; Leaves  $u, v$ ;  
*Output:* Bipartitions  $\{\psi_j(u, v)\}_{j=1}^{r(u, v)}$ ;

- **Ball.** Let

$$\hat{B}(u, v) := \left\{ w \in \hat{h}_m^{(i)} : \hat{d}(u, w) \vee \hat{d}(v, w) < M \right\};$$

- **Intersection Points.** For all  $w \in \hat{B}(u, v)$ , estimate the point of intersection between  $u, v, w$  (distance from  $u$ ), that is,

$$\hat{\Phi}_w := \frac{1}{2} \left( \hat{d}(u, v) + \hat{d}(u, w) - \hat{d}(v, w) \right);$$

- **Long Edges.** Set  $S := \hat{B}(u, v) - \{u\}$ ,  $x_{-1} = u$ ,  $j := 0$ ;

- Until  $S = \emptyset$ :

- \* Let  $x_0 = \arg \min \{\hat{\Phi}_w : w \in S\}$  (break ties arbitrarily);
- \* If  $\hat{\Phi}_{x_0} - \hat{\Phi}_{x_{-1}} \geq 2\tau$ , create a new edge by setting  $\psi_{j+1}(u, v) := \{\hat{B}(u, v) - S, S\}$  and let  $C_{j+1} := \{x_0\}$ ,  $j := j + 1$ ;
- \* Else, set  $C_j := C_j \cup \{x_0\}$ ;
- \* Set  $S := S - \{x_0\}$ ,  $x_{-1} := x_0$ ;

- **Output.** Return the bipartitions  $\{\psi_j(u, v)\}_{j=1}^{r(u, v)}$  (where  $r(u, v)$  is the number of bipartitions generated in the previous step).

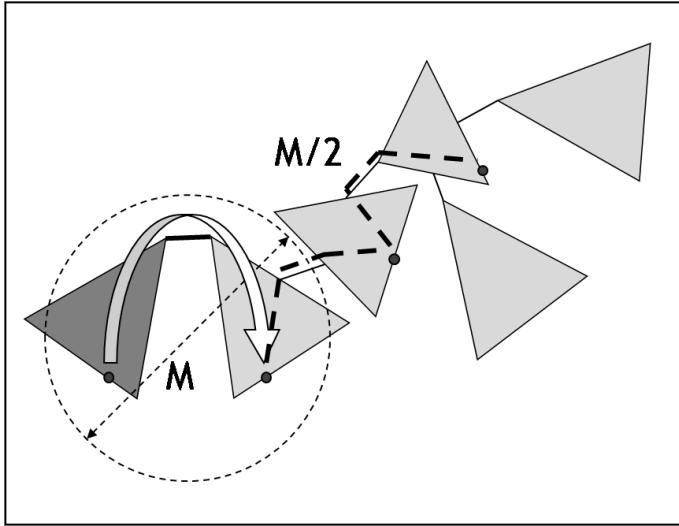
**Fig. 4.** Algorithm MINI CONTRACTOR. See Figure 3 for illustration.

$P_{T^{(i)}}(u, v)$ —at least those that are “sufficiently long.” This phase is detailed in Figure 4. An illustration is given in Figure 3.

*Building the components II: Extending the bipartitions.* The previous step reconstructs “sufficiently long” edges on balls of the form  $\hat{B}(u, v)$ . By *reconstructing an edge on  $\hat{B}(u, v)$* , we mean *finding the bipartition of  $\hat{B}(u, v)$  to which the edge corresponds*. More precisely:

**Definition 6 (Bipartitions).** Let  $T = (V, E)$  be a multifurcating tree with no vertex of degree 2. Each edge  $e$  in  $T$  induces a bipartition of the leaves  $L$  of  $T$  as follows: if one removes the edge  $e$  from  $T$ , then one is left with two connected components; take the partition of the leaves corresponding to those components. Denote by  $b_T(e)$  the bipartition of  $e$  on  $T$ . It is easy to see that given the bipartitions  $\{b_T(e)\}_{e \in E}$  one can reconstruct the tree  $T$  efficiently [29, 30, 31]. (Proceed by sequentially “splitting” clusters.)

The goal of the second phase in the main loop of our reconstruction algorithm is to *extend* the bipartitions previously built from  $\hat{B}(u, v)$  to the full component  $\hat{h}_m^{(i)}$ . To perform this task, we use the following observation: suppose we want to deduce the bipartition corresponding to edge  $e$ ; if we take the ball  $\hat{B}(u, v)$  to be much larger than  $m$  (yet small enough that it remains within our radius of precision  $M$ ), we can make sure that a path *from* a leaf in  $\hat{h}_m^{(i)}$  that is outside  $\hat{B}(u, v)$  to a leaf on the other side of the bipartition is “long.” Therefore, we can



**Fig. 5.** Illustration of routine EXTENDER. See also Figure 6.

**Algorithm EXTENDER**

*Input:* Component  $\hat{h}_m^{(i)}$ ; Bipartitions  $\{\psi_j(u, v)\}_{j=1}^{r(u, v)}$ ; Leaves  $u, v$ ;

*Output:* Bipartitions  $\{\bar{\psi}_j(u, v)\}_{j=1}^{r(u, v)}$ ;

- For  $j = 1, \dots, r(u, v)$  (unless  $r(u, v) = 0$ ):
  - **Initialization.** Denote by  $\psi_j^{(u)}(u, v)$  the vertex set containing  $u$  in the bipartition  $\psi_j(u, v)$ , and similarly for  $v$ ; Initialize the extended partition  $\bar{\psi}_j^{(u)}(u, v) := \psi_j^{(u)}(u, v)$ ,  $\bar{\psi}_j^{(v)}(u, v) := \psi_j^{(v)}(u, v)$ ;
  - **Modified Graph.** Let  $K$  be  $\hat{h}_m^{(i)}$  where all edges between  $\psi_j^{(u)}(u, v)$  and  $\psi_j^{(v)}(u, v)$  have been removed;
  - **Extension.** For all  $w \in \hat{v}_m^{(i)} - (\psi_j^{(u)}(u, v) \cup \psi_j^{(v)}(u, v))$ , add  $w$  to the side of the partition it is connected to in  $K$  (by definition of  $K$ , each  $w$  as above is connected to exactly one side);
- Return the bipartitions  $\{\bar{\psi}_j(u, v)\}_{j=1}^{r(u, v)}$ .

**Fig. 6.** Algorithm EXTENDER. See Figure 5 for an illustration.

easily determine what side of the partition each leaf in  $\hat{h}_m^{(i)}$  lies on. For details, see Figure 6. An illustration is given in Figure 5.

### 3 Concluding Remarks

An interesting question for future work is whether the *approximate* disjointness in our results can be avoided. Since we guarantee that any shared edge lies deep

inside the forest, it is tempting to simply remove all deep edges (say beyond  $m/4$ ) from the output forest. Unfortunately, many of these edges may in fact be contracted and moreover they may be clustered in “supernodes” including both deep and not-so-deep edges. It does not seem to be a trivial task to break these deep supernodes apart and preserve strong reconstruction guarantees.

## References

1. Felsenstein, J.: Inferring Phylogenies. Sinauer, Sunderland (2004)
2. Semple, C., Steel, M.: Phylogenetics. Mathematics and its Applications series, vol. 22. Oxford University Press, Oxford (2003)
3. Graham, R.L., Foulds, L.R.: Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Math. Biosci.* 60, 133–142 (1982)
4. Day, W.H.E., Sankoff, D.: Computational complexity of inferring phylogenies by compatibility. *Syst. Zool.* 35(2), 224–229 (1986)
5. Day, W.H.E.: Computational complexity of inferring phylogenies from dissimilarity matrices. *Bull. Math. Biol.* 49(4), 461–467 (1987)
6. Chor, B., Tuller, T.: Finding a maximum likelihood tree is hard. *J. ACM* 53(5), 722–744 (2006)
7. Roch, S.: A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Trans. Comput. Biology Bioinform.* 3(1), 92–94 (2006)
8. Felsenstein, J.: Cases in which parsimony or compatibility methods will be positively misleading. *Syst. Biol.*, 401–410 (1978)
9. Atteson, K.: The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica* 25(2-3), 251–278 (1999)
10. Lacey, M.R., Chang, J.T.: A signal-to-noise analysis of phylogeny estimation by neighbor-joining: insufficiency of polynomial length sequences. *Math. Biosci.* 199(2), 188–215 (2006)
11. Steel, M.A., Székely, L.A.: Inverting random functions. *Ann. Comb.* 3(1), 103–113 (1999); 3 Combinatorics and biology (Los Alamos, NM, 1998)
12. Steel, M.A., Székely, L.A.: Inverting random functions. II. Explicit bounds for discrete maximum likelihood estimation, with applications. *SIAM J. Discrete Math.* 15(4), 562–575 (electronic 2002)
13. Erdős, P.L., Steel, M.A., Székely, L.A., Warnow, T.A.: A few logs suffice to build (almost) all trees (part 1). *Random Struct. Algor.* 14(2), 153–184 (1999)
14. Saitou, N., Nei, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4(4), 406–425 (1987)
15. Daskalakis, C., Mossel, E., Roch, S.: Phylogenies without branch bounds: Contracting the short, pruning the deep (2008) (preprint),  
<http://arxiv.org/abs/0801.4190>
16. Chang, J.T.: Full reconstruction of Markov models on evolutionary trees: identifiability and consistency. *Math. Biosci.* 137(1), 51–73 (1996)
17. Philippe, H., Laurent, J.: How good are deep phylogenetic trees? *Current Opinion in Genetics & Development* 8(8), 616–623 (1998)
18. 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)

19. Mossel, E.: Distorted metrics on trees and phylogenetic forests. *IEEE/ACM Trans. Comput. Bio. Bioinform.* 4(1), 108–116 (2007)
20. King, V., Zhang, L., Zhou, Y.: On the complexity of distance-based evolutionary tree reconstruction. In: *SODA 2003: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, pp. 444–453 (2003)
21. Daskalakis, C., Hill, C., Jaffe, A., Mihaescu, R., Mossel, E., Rao, S.: Maximal accurate forests from distance matrices. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) *RECOMB 2006. LNCS (LNBI)*, vol. 3909, pp. 281–295. Springer, Heidelberg (2006)
22. Gronau, I., Moran, S., Snir, S.: Fast and reliable reconstruction of phylogenetic trees with very short edges. To appear in *SODA* (2008)
23. Erdős, P.L., Steel, M.A., Székely, L.A., Warnow, T.A.: A few logs suffice to build (almost) all trees (part 2). *Theor. Comput. Sci.* 221, 77–118 (1999)
24. Huson, D.H., Nettles, S.H., Warnow, T.J.: Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *J. Comput. Biol.* 6(3-4) (1999)
25. Csurös, M., Kao, M.Y.: Provably fast and accurate recovery of evolutionary trees through harmonic greedy triplets. *SIAM Journal on Computing* 31(1), 306–322 (2001)
26. Csurös, M.: Fast recovery of evolutionary trees with thousands of nodes. *J. Comput. Biol.* 9(2), 277–297 (2002)
27. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden Markov models. *Ann. Appl. Probab.* 16(2), 583–614 (2006)
28. Daskalakis, C., Mossel, E., Roch, S.: Optimal phylogenetic reconstruction. In: *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pp. 159–168. ACM Press, New York (2006)
29. Buneman, P.: The recovery of trees from measures of dissimilarity. In: *Mathematics in the Archaeological and Historical Sciences*, pp. 187–395. Edinburgh University Press, Edinburgh (1971)
30. Meacham, C.A.: A manual method for character compatibility analysis. *Taxon* 30, 591–600 (1981)
31. Bandelt, H.J., Dress, A.: Reconstructing the shape of a tree from observed dissimilarity data. *Adv. Appl. Math.* 7(3), 309–343 (1986)

# Detecting the Presence and Absence of Causal Relationships between Expression of Yeast Genes with Very Few Samples

Eun Yong Kang<sup>1</sup>, Ilya Shpitser<sup>2</sup>, Chun Ye<sup>3</sup>, and Eleazar Eskin<sup>4,\*</sup>

<sup>1</sup> Department of Computer Science  
University of California, Los Angeles  
Los Angeles, CA 90095  
[ekang@cs.ucla.edu](mailto:ekang@cs.ucla.edu)

<sup>2</sup> Department of Epidemiology  
Harvard School of Public Health  
Boston, MA 02115  
[ishpitse@hsph.harvard.edu](mailto:ishpitse@hsph.harvard.edu)

<sup>3</sup> Bioinformatics Program  
University of California, San Diego  
La Jolla, CA 92093  
[cye@bioinf.ucsd.edu](mailto:cye@bioinf.ucsd.edu)

<sup>4</sup> Department of Computer Science  
Department of Human Genetics  
University of California Los Angeles  
Los Angeles, CA 90095  
[eeskin@cs.ucla.edu](mailto:eeskin@cs.ucla.edu)

**Abstract.** Inference of biological networks from high-throughput data is a central problem in bioinformatics. Particularly powerful for network reconstruction is data collected by recent studies that contain both genetic variation information and gene expression profiles from genetically distinct strains of an organism. Various statistical approaches have been applied to these data to tease out the underlying biological networks that govern how individual genetic variation mediates gene expression and how genes regulate and interact with each other. Extracting meaningful causal relationships from these networks remains a challenging but important problem. In this paper we use causal inference techniques to infer the presence or absence of causal relationships between yeast gene expressions in the framework of graphical causal models. We evaluate our method using a well studied dataset consisting of both genetic variation information and gene expressions collected over yeast strains. Our predictions of causal regulators are consistent with previously known experimental evidence. In addition, our method can distinguish between direct and indirect effects of variation on a gene expression level.

---

\* To whom correspondence should be addressed.

## 1 Introduction

Inference of biological networks from high-throughput genomic data is a central problem in bioinformatics where many different types of methods have been proposed and applied to a wide diversity of datasets [1]. Several recent studies have collected data which contain both genetic variation information as well as gene expression information from a set of genetically distinct strains of an organism. Originally, these “genetical genomics” datasets were used to identify genetic variations located at specific genomic locations that affect expression levels in the form of linkages or associations [2][3]. These studies treated expression levels as quantitative traits and each associated genomic location is called an expression quantitative trait locus (eQTL). However, more recently, various statistical approaches have been applied to these datasets demonstrating them as being particularly powerful for teasing out the underlying biological networks that govern how genetic variations mediate differential gene expression and how genes regulate and interact with each other [4][5][6][7]. Some of these methods build on pioneering work in using graphical models to model gene regulatory networks [8][9][10][11][12].

Extracting meaningful causal relationships from these networks has been a challenging but important area of genetical genomics. A principled way of representing causal relationships is using graphical causal models [13][14]. Such models represent causal relationships between random variables by means of a directed acyclic graph called a causal graph, where a directed edge between two variables represents direct causal influence. The data-generating process represented by a causal graph imposes a variety of constraints, such as conditional independence constraints, on the observed data. A rich theory of causal inference has been developed [14][15] which attempts to reconstruct aspects of the graph from the pattern of constraints in the observations. Causal relationships can then be read off directly from the reconstructed graph.

The advantage of the causal inference paradigm is that predictions made are in fact causal, and so can be directly verified with knockout, siRNA or allele swap experiments. Compared to other methods such as co-expression networks which aim to capture the global structure in the regulatory network, causal inference methods attempt to identify the actual biological mechanisms regulating gene expression. Furthermore, for many applications where the final goal is to perturb the biological system in some way, causal networks are advantageous because they naturally predict the effect of possible interventions. The resulting models can be perturbed *in silico* to help guide which experimental perturbations to apply.

The disadvantage of these methods is that existing causal inference theory is a large sample theory, and is only guaranteed to work asymptotically. Unfortunately, in the case of inferring biological networks from gene expression data, there are far fewer samples than genes, which means practical applications must be a successful synthesis of ideas from both causal inference and small sample statistics.

There are two main approaches to learning causal graphs in biological networks. Score-based methods assign scores to models which both produce high

likelihood of the observed data, and have limited complexity, and search for the highest scoring model [16][17]. These methods have been used in identifying causal regulators in yeast [8][9][10][18][19][20] and causal mediators of disease in mice [21]. Constraint-based methods rule out those causal graphs inconsistent with patterns of conditional independence constraints in the observations. These methods have been applied to discovering causal relationship between pairs of genes [22].

In this paper, we discover the presence and absence of causal relationships between genes in yeast by examining their expression levels over a set of strains with random genetic variation. Causal discovery is challenging in our case because there are several thousand genes, while our number of samples is very limited. In particular, most conventional conditional independence tests or model selection algorithms are not reliable in the small sample case, since conditioning severely reduces the number of samples available, and as a result we cannot infer independence with high confidence, limiting our ability to induce features of the causal graph.

Our approach is to rely on basic properties of graphical models to infer or exclude edge directionality based on either simple unconditional independence tests which are possible to perform even in the small sample case, or on results of simple model selection amongst small causal sub-graphs of the overall causal model which have particularly strong signals. Our philosophy is that due to the small number of samples, it is impossible to accurately recover the complete causal graph. We opt to predict only the subset of the network where our predictions are likely to be correct.

We take advantage of prior biological knowledge that genetic variation affects gene expression, but not vice versa. This knowledge can be expressed graphically as forbidding directed paths from gene expression levels to genetic variation. While in general it is not possible to recover most causal structure based on unconditional independence tests, the availability of prior knowledge of biology allows us to “bootstrap” certain edge orientations, which in turn allows us to orient more paths as causal using basic properties of *d-separation* in graphical models (described below). Moreover, our method allows us to rule out certain edge orientations, and thus rule out certain causal relationships.

Our method is inherently conservative, only predicting the existence and orientation of edges in the causal graph if there is strong support from the sample data. As expected, our approach predicts only a small fraction of the complete causal regulatory network of yeast. However, the actual predictions made by the method are surprisingly consistent with previous experimentally validated knowledge of yeast gene regulation.

We demonstrate our method by analyzing the Brem et al. yeast strains [2]. This set of strains was generated by crossing a laboratory strain and a wild strain of *S. cerevisiae*. Both genetic variation and gene expression from each offspring have been collected. This dataset is characterized by several “regulatory hotspots” or regions in the genome in which a genetic variation affects the expression levels of many genes. Compared to a traditional eQTL study which

identifies correlated variation-expression level pairs, our method provides much richer information. First, our method allows us to infer regulatory relationships between pairs of genes. Second, our method can exclude causal relationships between genes. Third, even when considering only variation-expression level pairs, our method can distinguish whether a variation has a direct or an indirect effect on expression levels. While several other methods attempt to infer causal relationships between genes [18,22], our method is the first to be able to exclude regulatory relationships and distinguish between direct and indirect effects of variation.

In order to evaluate our inference of regulatory relationships, we compare our method directly to two other methods for causal inference on this data as well as to experimentally validated results. Using our method, of the 12 genes for which there is some experimental evidence that they behave as master regulators [23,18], we recover 9 of them. Furthermore, for one of our predicted genes, *ILV6*, a competing method Zhu et al., 2008 [18] was not able to identify the gene as a causal regulator based on expression data alone and was only able to discover this gene when also incorporating transcription factor binding data. We also use our method to exclude specific causal relationships. Using enrichment analysis, we find that gene targets not causally affected by a regulator are enriched for a different set of pathways and biological processes compared to gene targets which are affected by the same regulator.

In order to evaluate our ability to distinguish between direct and indirect effects of variation, we take advantage of the fact that most expression transcripts are affected by variation close to the gene through a mechanism called *cis*-regulation. *Cis*-effects most likely affect expression directly. Since our method does not have information on the relative position of the variation compared with the genes, our predictions are validated by enrichment of predicted direct regulators for *cis*-regulations.

## 2 Causal Graphs for Genetical Genomics

We first introduce the machinery of causal inference needed to formalize our approach to inferring causal relationships between a genetic variation (a SNP) and the expression of a pair of genes. Our primary object of study is the probabilistic causal model [14].

**Definition 1.** A probabilistic causal model (PCM) is a tuple  $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u}) \rangle$ , where

- $\mathbf{U}$  is a set of background or exogenous variables, which cannot be observed or experimented on, but which can influence the rest of the model.
- $\mathbf{V}$  is a set  $\{V_1, \dots, V_n\}$  of observable or endogenous variables. These variables are considered to be functionally dependent on some subset of  $\mathbf{U} \cup \mathbf{V}$ .
- $\mathbf{F}$  is a set of functions  $\{f_1, \dots, f_n\}$  such that each  $f_i$  is a mapping from a subset of  $\mathbf{U} \cup \mathbf{V} \setminus \{V_i\}$  to  $V_i$ , and such that  $\bigcup \mathbf{F}$  is a function from  $\mathbf{U}$  to  $\mathbf{V}$ .
- $P(\mathbf{u})$  is a joint probability distribution over the variables in  $\mathbf{U}$ .

PCMs represent causal relationships between observable variables in  $\mathbf{V}$  by means of the functions in  $\mathbf{F}$ : a given variable  $V_i$  is causally determined by  $f_i$  using the values of the variables in the domain of  $f_i$ . Causal relationships entailed by a given PCM have an intuitive visual representation using a graph called a causal diagram. In this graph, each node is represented by a vertex, and a directed edge is drawn from a variable  $X$  to a variable  $V_i$  if  $X$  appears in the domain of  $f_i$ . A graph obtained in this way from a model is said to be induced by said model.

A node  $Y$  is an *ancestor* of node  $Z$  in a causal diagram  $G$  if there is a directed path from  $Y$  to  $Z$ . Causal diagrams are generally assumed to be acyclic. While we expect the full yeast regulatory network to have causal cycles (they serve as common regulatory mechanisms), in this paper we concentrate our efforts on the fragments of the overall network where acyclicity holds.

One advantage of causal graphs, and graphical models in general [13][24] is their ability to represent conditional independence relations between variables in a qualitative and intuitive way using the notion of path blocking known as d-separation [13]. Two variables  $X, Y$  are d-separated if all causal and confounding paths from  $X$  to  $Y$  contain at least one variable whose value is known, and the value of no common effect of both  $X$  and  $Y$  is known. Every d-separation statement involving two nodes (or sets of nodes) in the graph corresponds to a conditional independence among the corresponding sets of variables. That is, if every path from  $X$  to  $Y$  is blocked or d-separated by  $Z$  in a causal diagram  $G$ , then  $X$  and  $Y$  are conditionally independent given  $Z$  in every probability distribution compatible with  $G$  [13]. Furthermore in stable [25] or faithful [26] models the converse is also true: conditional independencies in the observations imply the corresponding d-separation statement holds in the underlying causal graph. The faithfulness assumption thus allows us to infer aspects of the generating causal graph from conditional independence constraints apparent in the data, and is crucial for inductive causal inference. Faithfulness holds in “most” causal models, and can thus be justified on Occam’s Razor grounds [14].

Constraint-based inference of correct edge orientations in a causal diagram has two fundamental limits in practical applications. The first is that it can be difficult to collect sufficient samples to perform reliable conditional independence tests, and the second is that some causal diagrams may disagree on orientations of particular edges while entailing the same set of conditional independence constraints (such causal diagrams are called Markov-equivalent [27]).

In this paper, we will use causal graphs to represent causal interactions between genetic variations and gene expression levels in yeast. In our case the genetic variations is the set of single nucleotide polymorphisms (SNPs). In this paper, we limit our focus to inferring the presence or absence of a causal relationship between gene expression levels based on independence tests and model selection we can actually perform. We will be relying on the following three (elementary) theorems in graphical models.

**Theorem 1.** *Let  $G$  be a causal graph where  $X$  is d-connected to  $Y$  via a path ending in an arrow pointing to  $Y$ ,  $X$  is d-connected to  $Z$ , and  $X$  and  $Z$  are d-separated by  $Y$ . Then  $Y$  is an ancestor of  $Z$ .*

If we assume faithfulness, this theorem implies we can infer causal directionality based on the result of two unconditional independence tests, and one conditional independence test. In our case,  $X$  is a SNP,  $Y$  is the expression level of a gene and  $Z$  is the expression level of a second gene. We are using our prior knowledge that expression levels do not affect SNP values to satisfy one of the preconditions of the theorem, namely that the d-connected path must end in an arrow pointing to  $Y$ . In particular, if  $Y$  is a gene expression value, and  $X$  is a SNP value correlated with  $Y$ , then  $Y$  cannot cause  $X$ . Using this theorem, we are able to infer a causal relationship between the expression levels of genes  $Y$  and  $Z$ .

Unfortunately, testing whether  $X$  is conditionally independent of  $Z$  given  $Y$  in the small sample case is not feasible. An alternative approach which is more appropriate in our case is to use a model selection method, that is rather than performing the independence test, find the causal model over the local variables of interest, and read off causal directionality from its graph. In general, if we restrict ourselves to a small part of a large causal model which contains three variables  $X, Y, Z$ , the causal diagram which captures conditional independencies in the corresponding marginal distribution, that is  $P(x, y, z)$ , will be a mixed graph containing both directed and bidirected arcs, called a latent projection [27]. In latent projections, a directed arc from  $X$  to  $Y$  corresponds to a d-connected path which starts with an arrow pointing away from  $X$  and ends with an arrow pointing towards  $Y$  in the original, larger graph such that every node on the path other than  $X$  and  $Y$  is marginalized out or latent. Similarly, a bidirected arc from  $X$  to  $Y$  corresponds to a d-connected path in the larger graph which starts with an arrow pointing to  $X$ , ends with an arrow pointing to  $Y$ , and every node on the path other than  $X$  and  $Y$  is marginalized out or latent.

If we restrict ourselves to local models of three variable marginal distributions, where certain causal relationships are excluded due to prior knowledge (e.g. genes cannot cause SNPs), the complete set of causal hypotheses is captured by a small set of latent projections.

**Theorem 2.** *Let  $G$  be a causal graph where  $X$  is an ancestor of  $Y$  and  $Z$ . Then the latent projection which represents conditional independencies of  $P(x, y, z)$  is one of the graphs in Figure II(a).*

Theorem 2 allows us to select the graph in Figure II(a) which best fits the available data (we use a version of the likelihood ratio test), and use this graph to conclude causal directionality. The next theorem allows us to conclude the opposite, that a variable cannot be a causal ancestor of another.

**Theorem 3.** *Let  $G$  be a causal graph where  $X$  is d-connected to  $Y$ , and  $X$  and  $Z$  are d-separated. Then  $Y$  cannot be an ancestor of  $Z$ .*

As before, faithfulness allows us to apply this theorem to conclude the absence of causal directionality based on the results of two unconditional independence tests. In our case SNP  $X$  is associated with the expression level of gene  $Y$ , and SNP  $X$  is either independent of the expression level of gene  $Z$  or conditionally

independent given some other gene. In this case we can rule out a direct causal relation between expression levels of genes  $Y$  and  $Z$ . In our case, the possible models are shown in Figure 1(b). In the small sample case we again use a maximum likelihood method to perform such tests.

In the next section, we describe our statistical methodology in more detail.

### 3 Methodology

#### 3.1 Inference Algorithm Overview

Our algorithm for inferring the presence or absence of causal relationships of gene expression proceeds in four steps. First, we find for every gene expression, the set of potential causal SNPs using a likelihood ratio statistic. Second, we infer the presence of causal relationships between pairs of genes correlated with the same SNP by comparing the likelihoods of possible models. Third, we distinguish between direct and indirect effects of genetic variation on gene expression. Fourth, we infer the absence of causal relationships based on the results of step one and Theorem 3.

#### 3.2 Finding Potential Causal SNPs

In the first step, we attempt to find, for every gene expression level, the set of potential causal SNPs, in other words the set of SNPs which are either causal or which are confounded with causal SNPs.

To examine the (potential) causal relationship between SNP  $S_i$  and expression level  $E_j$  in our small sample case, we assume the following linear relationship between the two:  $E_j = \alpha S_i + \epsilon$ . We use an arrow notation to signify potential causality ( $\rightarrow$ ) and the negation ( $\not\rightarrow$ ) as no potential causality. Under the null hypothesis of no potential causal relationship between the SNP and expression levels ( $S_i \not\rightarrow E_j$ ), we expect  $\alpha = 0$  ( $H_0$ ). Under the alternate hypothesis of a potential causal relationship ( $S_i \rightarrow E_j$ ), we expect  $\alpha \neq 0$  ( $H_1$ ). To decide between these hypotheses we calculate the likelihood ratio statistic  $x_{ij} = -2 \log \frac{\mathcal{L}(H_0)}{\mathcal{L}(H_1)}$  which follows asymptotically the  $\chi^2$  distribution with 1 degree of freedom and a non centrality parameter of 0. We calculate the likelihood ratio statistic  $x_{ij}$  for every SNP/expression pair  $(S_i, E_j)$ . To assign significance, we shuffle the labels of the individuals  $B$  times to obtain the null statistics  $x_{ij}^0, b, b = 1, 2, \dots, B$ . Then the p-value of each SNP and expression pair can be calculated by looking at the ranking of the statistic of the pair in the permuted null statistic distribution.

We can easily estimate the false discovery rate (FDR) for our statistic using previous approaches [28]. To limit the number of potential causal networks to evaluate in subsequent steps, we filter the SNP/expression pairs for those with a FDR of  $q < 0.01$ .

Due to linkage disequilibrium or local correlation of variation, the SNPs which are correlated with expressions are not likely to be actually causal, but instead correlated with causal SNPs in the same genomic region. Since all of the SNPs are correlated in a region, this does not affect our ability to make inferences

about the causal regulatory network, but we must keep in mind that the SNPs which we predict to have direct effects are likely proxies for the true causal variants.

### 3.3 Finding Causal Relationships between Genes

The next stage of our algorithm consists of inferring causal directionality between gene expressions by using Theorem 1 and Theorem 2. Since the two unconditional independence tests have already been performed in the first step, all that remains is to test conditional independence. Unfortunately, conditional tests present a problem in the small sample case. An alternative approach is to consider multiple models consistent with the results of the unconditional independence tests where in some models the conditional independence holds, and in others it does not. If a model where the conditional independence test holds is the best fit for the data, and moreover accounts for more of the fit compared to a “default” model making no conditional independence assumptions, then we assume the conditional independence is likely true.

In our case, we are considering fragments of the causal graph consisting of a single SNP  $S$  and two expression levels  $E_i, E_j$  dependent on  $S$  (due to step 1). Figure 1(a) shows the nine possible causal models in the case that all of the elements are pairwise correlated. In  $H_1$  the SNP affects both expression levels independently. In  $H_2$  and  $H_3$  there is a direct causal relationship between the two expression levels. The “default” models  $H_4$  through  $H_9$  impose no constraints on the data and are indistinguishable based on conditional independence tests. Since they are all equivalent, for simplicity, we only consider  $H_4$  below.

We obtain information about the network whenever we predict a triplet to have a model  $H_1$ ,  $H_2$  or  $H_3$ . To distinguish between the three hypothesis  $H_1$ ,  $H_2$  and  $H_3$ , we perform likelihood ratio tests for each hypothesis against the alternative  $H_4$ , and conclude that a hypothesis is likely true if the corresponding ratio is close to unity (e.g. a simpler hypothesis accounts for the observations) and if it exceeds the other ratio (e.g. fits better than the other simple hypothesis). This is equivalent to the standard approach of performing a likelihood ratio test for model selection taking into account a complexity penalty. In this case, the complexity penalty would be applied to  $H_4$  since the model has an additional degree of freedom. We also pairwise compare the likelihoods between  $H_1$ ,  $H_2$  and  $H_3$  against each other and only consider triplets where the most likely hypothesis is more likely than the others using a threshold.

Since we assume a linear causal model, we compute the likelihoods using linear regression. The regression coefficients are interpreted as the Wright’s rule [29] sum of path products of coefficients in the underlying (and unknown) true causal graph.

### 3.4 Distinguishing between Direct and Indirect Effects of Variation

If a SNP  $S$  is associated with two genes  $E_i$  and  $E_j$ , the nine possible models are  $H_1$ ,  $H_2$ ,  $H_3$  and the “default” models  $H_4$  through  $H_9$ . The models  $H_2$  and

$H_3$  explain the associations as a direct effect of the SNP on one gene and an indirect effect on the other. Model  $H_1$  suggests that the SNP directly affects the expression levels of both genes. Since our statistical methodology uses  $H_4$  as the default model, we are unable to distinguish between direct and indirect effects if we can not classify a triplet as one of either  $H_1$ ,  $H_2$ , or  $H_3$ .

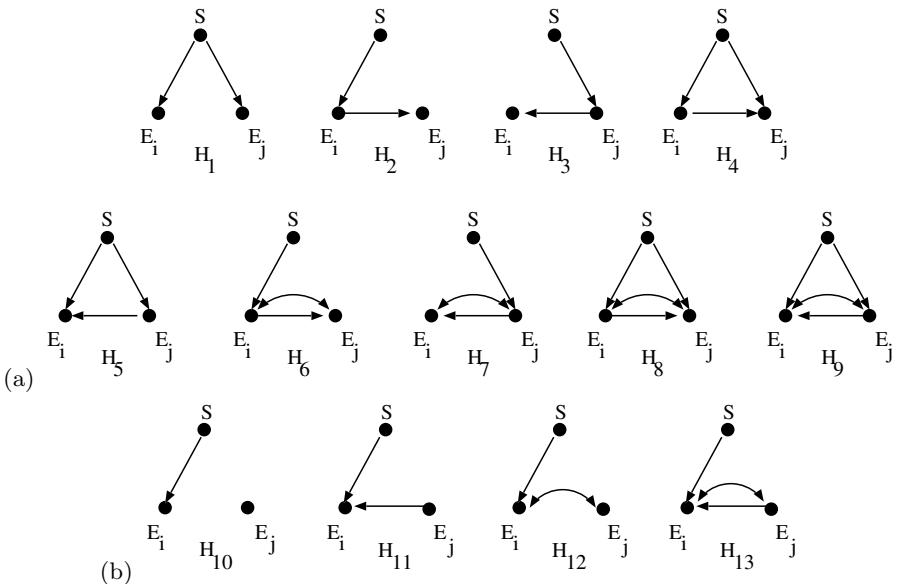
Establishing direct and indirect effects in causal analysis is always done with respect to particular model granularity. This is because it is generally always possible to observe intermediate variables between any direct cause and its effect – finer granularity removes directness of causation. In our case, when distinguishing direct versus indirect effects, the specific triplet that we are observing determines whether or not an effect is direct or indirect. Consider the following motivating example of a SNP  $S$  and three genes with the underlying network  $S \rightarrow E_1 \rightarrow E_2 \rightarrow E_3$ . If we consider the triplet  $(S, E_2, E_3)$ , the correct structure of the subgraph is  $H_2$  and  $S$  will have a direct effect on  $E_2$  and an indirect effect on  $E_3$ . Now if we consider the triplet  $(S, E_1, E_2)$ , the correct structure is again  $H_2$  and  $S$  will have a direct effect on  $E_1$  and an indirect effect on  $E_2$ . Intuitively, this is because when we consider the triplet  $(S, E_2, E_3)$ ,  $E_1$  is unobserved. Thus each prediction of a triplet as  $H_2$  or  $H_3$  induces a partial order on the causal relationships between gene pairs. After examining all pairs, we return the minimum set of causal relationships which are consistent with all of the triplet predictions.

More complicated networks introduce ambiguity into our ability to distinguish between direct and indirect effects. For example, if we add the edge  $S \rightarrow E_3$  in our example, we can still identify  $E_1 \rightarrow E_2$  as a direct effect from the triplet  $(S, E_1, E_2)$ , but are unable to identify  $E_2 \rightarrow E_3$  as a direct effect. This is because we will predict the structure of each triplet containing  $E_3$  and either  $E_1$  or  $E_2$  as  $H_4$  where the effects are ambiguous. However, if there is an additional edge in the graph  $E_3 \rightarrow E_4$ , the triplet  $(S, E_3, E_4)$  would identify  $E_3 \rightarrow E_4$  as a direct effect.

### 3.5 Excluding Causal Relationships between Genes

The ability to exclude certain causal relationships between genes, an inherent advantage of causal analysis, is important to obtaining a more complete understanding of genetic regulation. For example, a gene might be causal to a number of genes enriched for a biological process but not causal to a number of genes enriched for a different biological process even though it is correlated with both sets of genes. We attempt to determine the absence of causal relationships by looking at a SNP and a pair of genes where the SNP is the potential cause of one gene, but not the other. In this case, basic properties of d-separation (Theorem 3) guarantee that there are only four possibilities  $H_{10}$  through  $H_{13}$  (see Figure II(b)).

We identify pairs of genes where we can exclude causal relationships by identifying pairs  $E_i$  and  $E_j$  where a SNP is significantly associated FDR of  $q < 0.01$  with  $E_i$  and not associated FDR of  $q > 0.9$  with  $E_j$ . To identify significantly



**Fig. 1.** Possible causal graphs relating a triplet considering a SNP  $S$  with the level of gene expression for genes  $E_i$  and  $E_j$ . Bidirected edges denote hidden common causes. (a) Nine possible causal models consistent with  $S$  being a causal ancestor of  $E_i$  and  $E_j$  (models  $H_4$  through  $H_9$  are indistinguishable from observations of the triplet). (b) Four possible causal models consistent with  $S$  being a causal ancestor of  $E_i$  while being uncorrelated with  $E_j$ .

enriched pathways in Gene Ontology, we used Gene Set Enrichment Analysis [7] at a FDR of  $q < 0.05$ .

## 4 Results

We applied our method to an expression dataset of 5534 genes and a genotyping dataset of 2956 SNPs collected over 112 genetic segregants of yeast. After step 1, we found 42331 (SNP, expression) pairs where the SNP is causal to the expression at a FDR of  $q < 0.01$ . We constructed triplets from these causal pairs to significantly reduce the number of possible causal models to evaluate for causal relationships between the genes in step 2. For each triplet, we considered the four possible models  $H_1$ ,  $H_2$ ,  $H_3$  and  $H_4$  and identified the most likely as described above. We find the most likely of  $H_1$ ,  $H_2$  and  $H_3$  and required that the log likelihood difference of the best model be within 2 of  $H_4$ . This is equivalent to penalizing the likelihood of  $H_4$  and applying using the likelihood ratio for model selection. We found 3370 causal relationships consisting of 212 causal regulator genes and 1396 affected target genes. Table II shows the number of causal relationships, causal regulators and affected target genes discovered using various model complexity penalties for  $H_4$ . Inferring causal relationships with few

**Table 1.** Summary Statistics For Different Likelihood Thresholds

Complexity Penalty	# Causal Regulators	# Affected Genes	# Causal Relationships	# Direction Conflicts	# Causal Conflicts
1	146	1106	2135	34 (1.6%)	7 (.3%)
1.5	183	1272	2794	30 (1.1%)	11 (.4%)
2	212	1396	3370	44 (1.3%)	13 (.4%)
2.5	240	1524	3983	59 (1.5%)	18 (.5%)
3	266	1615	4571	81 (1.8%)	20 (.4%)

samples can result in directional and causal conflicts. A directional conflict occurs when the direction of causation predicted between two genes is inconsistent using different SNPs. A causal conflict occurs when the presence and absence of a causal relationship predicted between two genes is inconsistent using different SNPs. We examined the robustness of our method by quantifying the number of directional and causal conflicts. Directional conflicts result when triples containing the same pair of genes and different SNPs predict different directional causal relations between the genes. Causal conflicts result when different triples both predict and exclude the same causal edge. As Table 1 shows, consistent across complexity penalties, fewer than 3% of predicted causal relationships are in conflict. These prediction conflicts are due to the limited number of samples available. We exclude all conflict predictions from our final result.

The list of the top 50 causal regulators we found includes many genes identified in previous analyses, and is remarkably consistent with experimental findings. One way to make sense of the large number of causal relationships detected is to look for causal regulators that affect a number of genes or “causal hubs”. We will use the term “causal hub” and “causal regulator” interchangeably. Of particular interest is identifying causal regulators that are associated with “regulatory hotspots”, defined as regions of the yeast genome linked to the expression of a large number of genes. Presumably, these “causal hubs” are important regulatory elements that lead to subtle changes in expression of genes belonging to a number of different biological processes and functions. Previous analyses have identified several “regulatory hotspots” in the yeast genome but very little is known about the corresponding “causal hubs” because of the limited resolution of genotyping studies. Due to interest in “regulatory hotspots”, several groups have performed experimental validation of genes involved in regulating the hotspots [23,18]. We organize our results by grouping the causal regulators by associating them with a hotspot if they cause the expression of many genes associated with the hotspot. The top 50 genes which have the largest number of targets are summarized in Table 2.

Both Chen et al. [22] and Zhu et al. [18] applied causal inference methods to the same data allowing us to perform a direct comparison of the results. Among the genes suspected to be global regulators in the hotspots, there are a total of 12 causal regulators with some experimental evidence. Nine were proposed by

**Table 2.** Regulatory Hotspots and Corresponding Regulators

SNP Chr	SNP Loc	Regulators	Num Targets
2	390000	TAT1(49)	49
2	560000	<b>AMN1*+(226)</b> , <i>YSW1</i> <sup>+</sup> (190), <i>TBS1</i> <sup>*</sup> (177), <i>CNS1</i> <sup>*+</sup> (166) <i>ARA1</i> <sup>*</sup> (162), <i>SDS24</i> (60), <b>SUP45</b> <sup>*</sup> (31), <i>AGP2</i> (17), <i>LYS2</i> (17), <i>TOS1</i> <sup>*</sup> (16), <i>YBR137W</i> (16), <i>TYR1</i> (13),	313
3	100000	<i>NFS1</i> <sup>*</sup> (106), <i>CIT2</i> <sup>*</sup> (100), <b>LEU2</b> <sup>*+</sup> (77), <i>HIS4</i> (66), <b>ILV6</b> <sup>*+</sup> (29),	169
3	230000	<b>MATALPHA1</b> <sup>*</sup> (40), <i>MATALPHA2</i> (24)	41
5	130000	<b>URA3</b> <sup>*</sup> (17)	17
8	130000	<b>GPA1</b> <sup>*+</sup> (15),	15
12	110000	ASP3-1(13), ASP3-2 (14),	16
12	680000	<b>HAP1</b> <sup>*</sup> (22), <i>MAP1</i> <sup>*</sup> (22),	40
12	800000	<i>GAS2</i> (19)	19
12	1070000	<i>YLR464W</i> <sup>*</sup> (32), <i>YRF1-4</i> <sup>*</sup> (30), <i>YRF1-5</i> <sup>*</sup> (22), <i>YRF1-1</i> <sup>*</sup> (14)	33
14	503000	<b>SAL1</b> <sup>*+</sup> (138), <i>LAT1</i> (77), <i>COG6</i> (69), <i>TOP2</i> <sup>*</sup> (62), <i>MSK1</i> (38), <i>YNL035C</i> (38), <i>YML133C</i> (30), <i>NMA111</i> (20), <i>SWS2</i> (17), <i>NAM9</i> <sup>+</sup> (14), <i>MKT1</i> (13),	320
15	180000	<b>PHM7</b> <sup>*+(227), <i>RFC4</i>(96), <i>NDJ1</i><sup>*</sup>(69), <i>HAL9</i><sup>*</sup>(66), <i>ZEO1</i>(55), <i>WRS1</i>(38), <i>SKM1</i>(28), <i>YOL092W</i>(18),</sup>	263

the original group that collected the data: *AMN1*, *MAK5*, *LEU2*, *MATALPHA1*, *URA3*, *GPA1*, *HAP1*, *SIR3* and *CAT5* [23]. Three additional were validated in Zhu et al.: *ILV6*, *SAL1* and *PHM7* [18]. Our method discovers all but 3 of these (*MAK5*, *SIR3* and *CAT5*). We note that *SIR3* and *CAT5* have much weaker experimental evidence than the others and none of the comparison methods (neither Chen et al. [22] or Zhu et al. [18]) were able to find these three. The best validation of our method is that we were able to find *ILV6* which was experimentally validated in Zhu et al. [18]. However, Zhu et al. [18] used additional types of data (incorporating TFBS data from ChIP-chip experiments, phylogenetic conservation, and protein protein interaction data (PPI)) in order to discover *ILV6* and they claim that they would not have been able to discover *ILV6* if they used only the data that we used. We note that *ILV6* was also suggested as a regulator for this hotspot by Kulp et al. [20]. We recover the highlighted genes from Chen et al. [22] including *NAM9* which was not found by Zhu et al. [18] and is supported by “bioinformatics type evidence” (GO analysis, etc). A direct comparison to Chen et al. [22] is difficult because their results are organized in a different way, yet our results are consistent with Chen et al. [22] in that they highlight their discovery of 6 of the experimentally validated regulators which we also discover.

Table 2 summarizes our results. Experimentally validated predictions are shown in bold. Regulators with an asterisk (\*) were found by Zhu et al. [18]. Regulators marked with a plus (+) were found in the Chen et al. [22] study

and unlabeled regulators are novel predictions. In parentheses after the name of the regulator is the number of targets that we found. We note that in most cases the experimentally validated regulator is at the top of the list. We also observed that with various model complexity cut off, the ranking of predicted genes is maintained, if the model complexity cut off is less than a certain threshold. Of particular interest are a group of regulators linked to chromosome 14 which is enriched for mitochondrial genes. Previous published studies in yeast did not identify any putative regulators in this region [23]. We found a number of genes in this region including three previously identified genes *SAL1*, *NAM9* and *TOP2* and several proteins of unknown function including *NMA111* and *YNL035C*.

We validate our ability to distinguish between direct and indirect effects of variation by considering the genomic positions of SNPs and the locations of genes that they are associated with. Variation that affects expression can be classified into two broad categories: *cis*-regulation which is an effect of a variation near a gene that affects expression of the gene and *trans*-regulation which is an effect of variation located in one region of the genome affecting expression of genes in other regions. It is suspected that most *cis*-regulation is direct while *trans*-regulation may be either direct or indirect. Of the 42,331 SNP gene pairs where the SNP is associated with the expression of the gene, 11,328 are predicted as *cis*-regulated gene while 31,003 are *trans*-regulated gene. Using our approach, out of the 11,328 *cis*-regulated genes, we predict 9,385 of them to have a direct effect on expression. Out of 31,003 the *trans*-regulated genes, 20,509 of the SNP gene pairs have indirect effects. Thus *cis*-regulated genes are enriched in our predicted set of directly affected genes, while *trans*-regulated genes are enriched in indirectly affected genes.

We speculate that the identified causal regulators are likely to either directly control or perturb biological processes. However, step 3 of our analysis also identifies a collection of genes that are causally irrelevant to other genes. Combining results from these two steps can help us identify specific biological processes that are either regulated or not regulated by these causal regulators. We examined those eight significant causal regulators from our results with previous experimental validation. Table 3 shows the different GO pathways that are enriched when we examine the set of genes the regulator is causal for versus the set of genes the regulator is causally irrelevant from. The eight regulators appear to be involved in very different biological processes. For example, *AMN1* is a causal regulator for ribosome biogenesis and assembly while four other regulators *LEU2*, *MATALPHA1*, *URA3* and *ILV6* are causally irrelevant for the process. Similarly, *SAL1* is a causal regulator for the process of translation while *HAP1* and *PHM7* are causally irrelevant for the process. We notice that all significant processes are crucial for cell growth and survival but are controlled by different global regulators. The causal analysis shows that most of these global regulators participate specifically in certain biological processes. The one example of multiple regulators from the same regulatory hotspot includes *LEU2* and *ILV6*. In this case, these two regulators participate in similar biological processes of organic

**Table 3.** Significantly Enriched Processes For Causal and Not Causal Genes

Gene	Regulated Targets		Unregulated Targets	
	GO Pathway	p value	GO Pathway	p value
AMN1	Ribosome biogenesis and assembly	$1.7 \times 10^{-34}$	Establishment of localization	$2.3 \times 10^{-7}$
LEU2	Organic acid metabolic process	$3.0 \times 10^{-7}$	Ribosome biogenesis and assembly	$3.5 \times 10^{-10}$
MAT $\alpha$ 1	Biological regulation	$5.9 \times 10^{-6}$	Ribosome biogenesis and assembly	$3.6 \times 10^{-11}$
URA3	De novo pyrimidine base biosynthetic process	$4.6 \times 10^{-6}$	Ribosome biogenesis and assembly	$5.0 \times 10^{-6}$
HAP1	Mitochondrial electron transport chain	$2.4 \times 10^{-10}$	Translation	$3.6 \times 10^{-13}$
ILV6	Amine biosynthetic process	$2.4 \times 10^{-17}$	Ribosome biogenesis and assembly	$2.9 \times 10^{-16}$
SAL1	Translation	$7.9 \times 10^{-30}$	Chromosome organization and biogenesis	$1.3 \times 10^{-6}$
PHM7	Carbohydrate metabolic process	$3.7 \times 10^{-9}$	Translation	$2.0 \times 10^{-8}$

acid metabolic process and amine biosynthetic process respectively. We further confirmed the specificity of these global regulators by enrichment analysis for localization of the causal and causally irrelevant targets. For example, *SAL1*'s causal targets are enriched for localization to the ribosome while *HAP1*'s targets are enriched for localization to the mitochondrial membrane. Furthermore, both *PHM7* and *HAP1*'s causally irrelevant targets localized to cytosolic region of the cell where translation takes place. Similarly, although *LEU2* and *ILV6*'s causal targets are not enriched for a specific cellular compartment, their causally irrelevant targets are both enriched for the nucleolus where ribosome biogenesis and assembly takes place.

## 5 Conclusion

In this paper we combined principled representation of causality using graphical causal models combined with small sample statistical methods to infer the presence and absence of causal directionality in the yeast genome. Working with a dataset of genetically identical yeast strains allowed us to make strong causal assumptions about edge directionality in the underlying causal model. These assumptions, in turn, allowed us to take maximum advantage of the limited samples we had available by employing either unconditional independence tests, or simple model selection to discover or exclude causal directionality between gene expression levels.

We have demonstrated the usefulness of our method by recovering many experimentally validated causal regulators in yeast. In addition, our approach is

able to distinguish between direct and indirect variations and exclude causal relationships between genes. These results provide a rich description of the yeast gene regulation network beyond those predicted by competing causal methods.

This work motivates theoretical questions about the limits of causal inference based on either restricting or eliminating conditional independence tests, and relying strictly on unconditional tests. An interesting way to extend this work is to either empirically or theoretically characterize the strength of effects recoverable by our method.

## Acknowledgments

E.Y.K., C.Y., I.S. and E.E. are supported by the National Science Foundation Grants No. 0513612, No. 0731455 and No. 0729049, and National Institutes of Health Grant No. 1K25HL080079. This research was supported in part by the UCLA subcontract of contract N01-ES-45530 from the National Toxicology Program/National Institute of Environmental Health Sciences to Perlegen Sciences.

## References

1. Markowetz, F., Spang, R.: Inferring cellular networks—a review. *BMC Bioinformatics* 8(suppl. 6), S5 (2007)
2. Brem, R.B., Yvert, G., Clinton, R., Kruglyak, L.: Genetic dissection of transcriptional regulation in budding yeast. *Science* 296, 752–755 (2002)
3. Brem, R., Kruglyak, L.: The landscape of genetic complexity across 5,700 gene expression traits in yeast. *Proceedings of the National Academy of Sciences* 102, 1572–1577 (2005)
4. Lee, S.I., Pe'er, D., Dudley, A.M., Church, G.M., Koller, D.: Identifying regulatory mechanisms using individual variation reveals key role for chromatin modification. *Proceedings of the National Academy of Sciences* 103, 14062–14067 (2006)
5. Robinson, M., Grigull, J., Mohammad, N., Hughes, T.: Funspec: a web-based cluster interpreter for yeast. *BMC Bioinformatics* 3, 35 (2002)
6. Ghazalpour, A., Doss, S., Zhang, B., Wang, S., Plaisier, C., Castellanos, R., Brozell, A., Schadt, E., Drake, T., Lusis, A., Horvath, S.: Integrating genetic and network analysis to characterize genes related to mouse weight. *PLoS Genet* 2 (2006)
7. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., Mesirov, J.P.: From the cover: Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* 102, 15545–15550 (2005)
8. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology* 7(3-4), 601–620 (2000)
9. Pe'er, D., Regev, A., Elidan, G., Friedman, N.: Inferring subnetworks from perturbed expression profiles (2001)
10. Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D., Friedman, N.: Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat. Genet.* 34(2), 166–176 (2003)

11. Friedman, N.: Inferring Cellular Networks Using Probabilistic Graphical Models. *Science's STKE* 303(5659), 799 (2004)
12. Hartemink, A., Gifford, D., Jaakkola, T., Young, R.: Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In: *Pac. Symp. Biocomput.*, vol. 6, pp. 422–433 (2001)
13. Pearl, J.: *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco (1988)
14. Pearl, J.: *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge (2000)
15. Spirtes, P., Glymour, C., Scheines, R.: *Causation, Prediction, and Search*. MIT Press, Cambridge (2000)
16. Suzuki, J.: A construction of bayesian networks from databases based on an mdl scheme. In: *UAI 1993*, pp. 266–273 (1993)
17. Lam, W., Bacchus, F.: Learning bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence* 10(4) (1994)
18. Zhu, J., Zhang, B., Smith, E., Drees, B., Brem, R., Kruglyak, L., Bumgarner, R., Schadt, E.: Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature Genetics* 40(7), 854 (2008)
19. Bing, N., Hoeschele, I.: Genetical genomics analysis of a yeast segregant population for transcription network inference. *Genetics* 170, 533–542 (2005)
20. Kulp, D., Jagalur, M.: Causal inference of regulator-target pairs by gene mapping of expression phenotypes. *BMC Genomics* 7, 125 (2006)
21. Schadt, E.E., Lamb, J., Yang, X., Zhu, J., Edwards, S., GuhaThakurta, D., Sieberts, S.K., Monks, S., Reitman, M., Zhang, C., Lum, P.Y., Leonardson, A., Thieringer, R., Metzger, J.M., Yang, L., Castle, J., Zhu, H., Kash, S.F., Drake, T.A., Sachs, A., Lusis, A.J.: An integrative genomics approach to infer causal associations between gene expression and disease. *Nat. Genet.* 37, 710–717 (2005)
22. Chen, L., Emmert-Streib, F., Storey, J.: Harnessing naturally randomized transcription to infer regulatory relationships among genes. *Genome Biology* 8, R219 (2007)
23. Yvert, G., Brem, R., Whittle, J., Akey, J., Foss, E., Smith, E., Mackelprang, R., Kruglyak, L.: Trans-acting regulatory variation in *Saccharomyces cerevisiae* and the role of transcription factors. *Nature Genetics* 35, 57–64 (2003)
24. Jordan, M.I., Weiss, Y.: Graphical models: Probabilistic inference. In: Arbib, M. (ed.) *The Handbook of Brain Theory and Neural Networks*, 2nd edn., MIT Press, Cambridge (2002)
25. Pearl, J., Verma, T.S.: A theory of inferred causation. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pp. 441–452 (1991)
26. Spirtes, P., Glymour, C., Scheines, R.: *Causation, Prediction, and Search*. Springer, New York (1993)
27. Verma, T.S., Pearl, J.: Equivalence and synthesis of causal models. Technical Report R-150, Department of Computer Science, University of California, Los Angeles (1990)
28. Storey, J., Tibshirani, R.: Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences* 100, 9440–9445 (2003)
29. Wright, S.: Correlation and causation. *Journal of Agricultural Research* 20, 557–585 (1921)

# An Adaptive and Memory Efficient Algorithm for Genotype Imputation

Hyun Min Kang<sup>1</sup>, Noah A. Zaitlen<sup>2</sup>, Buhm Han<sup>1</sup>, and Eleazar Eskin<sup>3</sup>

<sup>1</sup> Computer Science and Engineering,

University of California, San Diego,

9500 Gilman Drive, La Jolla, CA 92093-0404

{h3kang, buhan}@cs.ucsd.edu

<sup>2</sup> Bioinformatics Program, University of California, San Diego,

9500 Gilman Drive, La Jolla, CA 92093-0419

nzaitlen@bioinf.ucsd.edu

<sup>3</sup> Department of Computer Science and Department of Human Genetics

University of California, Los Angeles,

3532-J Boelter Hall, Los Angeles, CA 90095-1596

eeskin@cs.ucla.edu

**Abstract.** Genome wide association studies have proven to be a highly successful method for identification of genetic loci for complex phenotypes in both humans and model organisms. These large scale studies rely on the collection of hundreds of thousands of single nucleotide polymorphisms (SNPs) across the genome. Standard high-throughput genotyping technologies capture only a fraction of the total genetic variation. Recent efforts have shown that it is possible to “impute” with high accuracy the genotypes of SNPs that are not collected in the study provided that they are present in a reference data set which contains both SNPs collected in the study as well as other SNPs. We here introduce a novel HMM based technique to solve the imputation problem that addresses several shortcomings of existing methods. First, our method is adaptive which lets it estimate population genetic parameters from the data and be applied to model organisms that have very different evolutionary histories. Compared to traditional methods, our method is up to ten times more accurate on model organisms such as mouse. Second, our algorithm scales in memory usage in the number of collected markers as opposed to the number of known SNPs. This issue is very relevant due to the size of the reference data sets currently being generated. We compare our method over mouse and human data sets to existing methods and show that each has either comparable or better performance and much lower memory usage. The method is available for download at <http://genetics.cs.ucla.edu/emiminim>.

## 1 Introduction

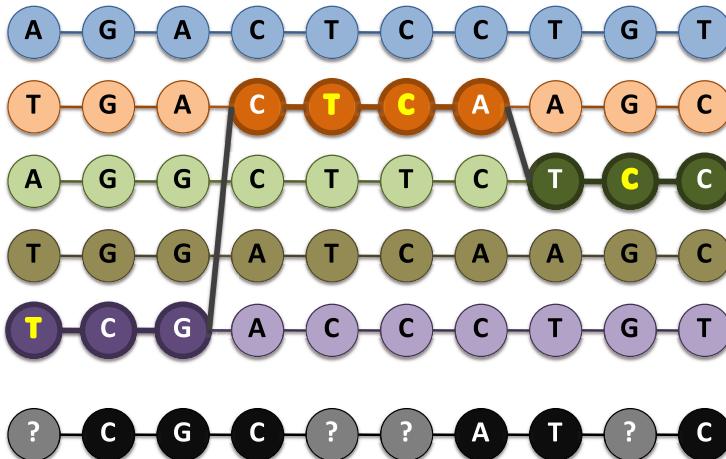
Recent advances in high-throughput genotyping technologies are helping to uncover the genetic basis of complex phenotypes in human[17], mouse[5], rat[16],

dog [8], arabidopsis [1], and other model organisms. While the vast majority of positions in a genome are identical among individuals in a population, a significant portion of positions differ. Many of these positions are single nucleotide polymorphisms (SNPs). In a typical study that attempts to identify variation involved in a trait, variation information (e.g. SNP genotypes), is collected from a set of individuals and the trait is measured in each individual. Each SNP is then correlated/associated with the trait. Any statistically significant associations are reported as possible causal variation with respect to the trait [133].

Genotyping arrays, such as those developed by Affymetrix and Illumina [126], simultaneously probe hundreds of thousands of marker SNPs in an individual's genome. While this is a significant amount of information, it is only a fraction of the millions of SNPs and other genetic variation in the population. Only complete resequencing of individual genomes will guarantee collecting all variation in a study. However, resequencing still remains prohibitively expensive. Array based genotyping is currently the most practical cost-effective method for collecting large amounts of variation information on a set of individuals. Although only a subset of individual genetic variation is collected by a genotyping array, due to the correlation structure of variation in the genome, SNPs on the array can serve as proxies for SNPs which are not collected [42]. This property is called linkage disequilibrium (LD), and it greatly extends the coverage of the array since a causal SNP need not be collected, but only strongly correlated with one of the collected markers on the array [18]. However, if the causal variants are not in LD with one of the SNPs included on the array then the study will not be able to discover the association. Thus, increasing the number of collected SNPs in the study increases the study's power to identify causal variation and is of fundamental importance.

Recently, several studies have proposed methods to increase the ability of a study to identify associations at SNPs which are not collected by "imputing" or predicting the genotypes of SNPs that are not contained in the study data set. These methods work by using a reference sample, such as the HapMap [7] for humans, which has genotyped millions of SNPs at great cost and effort. These reference samples contain both SNPs which are collected in the study as well as other SNPs. An imputation method uses the correlation patterns between the collected and uncollected SNPs inferred from the reference sample to make predictions of the uncollected SNPs in the study sample. This problem is effectively a missing data problem in which partial data is observed in the study and complete data is observed in the reference sample.

Consider the example shown in Figure 1, there is a set of reference individuals shown on top and a study individual shown on the bottom. In the reference set all the SNPs are genotyped in all five individuals. In the study individual, some of the SNPs are uncollected and denoted by a "?". The goal of imputation is to resolve the genotypes of the uncollected SNPs by using the overlap of the typed SNPs between the reference set and the study set. Our method selects the most likely reference individual for each marker (both collected and uncollected). The path in bold shown in Figure 1 denotes that the sequence of SNP values in target



**Fig. 1.** An example of the imputation problem. The five reference individuals are genotyped on all ten SNPs, while the study individual is genotyped on only six SNPs. The goal of imputation is to resolve the genotypes of the uncollected SNPs.

individual is composed of pieces of the three reference individuals along the six collected and four uncollected markers.

Multiple techniques have been successfully employed to solve the imputation problem, and Hidden Markov models (HMMs) have been amongst the most popular, and have been used in several studies in both human and mouse. However, each of these existing HMM techniques fails to address at least one of several important problems. IMPUTE [1] applies a standard population genetics model of recombination and mutation using a set of reference haplotypes. The transitional and mutational parameters of the HMM are predefined for each reference population, so the method is not adaptive to different populations or different organisms. For example, using IMPUTE on mouse genotypes will result in largely inaccurate estimates of uncollected SNPs. Moreover, it has memory requirements that grow linearly in the size of the reference data set, which may become prohibitively large as more SNPs are discovered. On the other hand, MACH [10] allows us to learn parameters of an HMM from the observed genotypes. However, in order to learn parameters easily, it uses a much simpler transitional model which does not utilize the continuous-time Markov chain model of recombination in population genetics [9]. Moreover, since the transitional parameters are estimated per each marker interval separately, MACH requires a large number of samples to be simultaneously imputed for an accurate parameter estimation. For inbred mouse imputation, the problem differs from human because the reference data sets have dramatically different linkage properties from human and do not have heterozygous genotypes. Recently, a fastPHASE [14] like method recently employed HMMs to solve the imputation problem in mouse heuristically using a predefined sets of clusters with Dirichlet prior distribution and Viterbi training method [15]. Although this recent method did impute many of the SNPs in the mouse, the average imputation error was 10.4% and 4.4% for high-confidence

genotypes, which is higher than the error rates in most of the human imputation studies.

In this paper, we propose EMINIM (Expectation-Maximized INtegrative IMputation), an adaptive genotype imputation method that learns HMM parameters with the standard population genetics model of recombination and mutation using Expectation-Maximization (EM) algorithm. Our method is motivated by the fact that the previous methods may not be applied to model organisms such as inbred mouse strains due to the predefined parameters being inappropriate or the small number of strains in the reference sample (in this case only 16). Our method utilizes various types of silent states in the HMM to estimate the EM parameters, to increase memory-efficiency, to impute genotypes at collected SNPs, and to obtain a leave-one-snp-out estimate of imputation accuracies. Our method is also more memory efficient allowing much larger data sets to be imputed. In addition, we provide an extensive implementation detail of our method that improves accuracy and computational efficiency in addition to the core statistical model, in order to facilitate further progress in the area of genotype imputation.

We applied our method to the imputation of 8.27 million SNPs that have been discovered from a resequencing of 15 inbred mouse strains, based on the 138,980 SNPs collected from the mouse HapMap project over 94 inbred strains. Imputation in mouse strains differs from human imputation because the reference datasets have drastically different linkage properties. Using a leave one out procedure, we measured the error rate of our method and compared to the recently published mouse imputation paper [15]. Our method's overall error rate is less than half the error rate of the previous method, and for high confidence genotypes our error rate is ten time smaller.

Next, we applied our method to the imputation of human HapMap SNPs from the Wellcome Trust Case-Control Consortium (WTCCC) genotypes. Our results show that our method consistently achieves similar or better imputation accuracy over different populations than other state-of-the-art methods without requiring predefined parameters for each population. Our method also shows a significant increase of memory-efficiency which can be an important technical issue when imputing genotypes of a dense SNP sets over a large number of reference samples. For example, on a recent study EMINIM used only 508 MB of memory to impute chromosome 22 while IMPUTE required 6.6 GB. This problem will become even more severe on larger chromosomes and data sets. Our method is publicly available at <http://genetics.cs.ucla.edu/eminim>.

## 2 Materials and Methods

### 2.1 The Imputation Problem

In this section we formalize the problem of imputing missing genotype data in an individual using a reference population. The terms and definitions described here will be used throughout the text. We classify the imputation problem into two categories - haploid (or inbred) imputation and diploid imputation. Suppose

that we genotype  $m$  SNP *markers* on an individual (*target individual*) and wish to determine the genotypes of additional “*uncollected*” SNPs. We will employ a set of *reference haplotype* that are genotyped on the  $m$  collected markers as well as an additional set of uncollected SNPs. In diploid model, we assume that each reference individuals is already phased into two reference haplotypes. The allele of the  $i$ -th reference haplotype at the  $t$ -th marker is represented as  $G_{i,t} \in \{0, 1, 2\}$ , where 0 represents a missing reference genotype and 1,2 represents two alleles of biallelic SNP marker. Let  $n$  be the number of reference haplotype collected at  $m$  markers in the target individual. Let  $\mathbf{d} = \{d_1, d_2, \dots, d_{m-1}\}$  be the physical or genetic distance between consecutive markers. In the target individual, the genotype at the  $t$ -th marker is represented as  $g_t \in \{1, 2\}$  in haploid model, and  $g_t \in \{\{1, 1\}, \{1, 2\}, \{2, 2\}\}$  in diploid model, where 1 and 2 represents two alleles of the biallelic SNP marker.

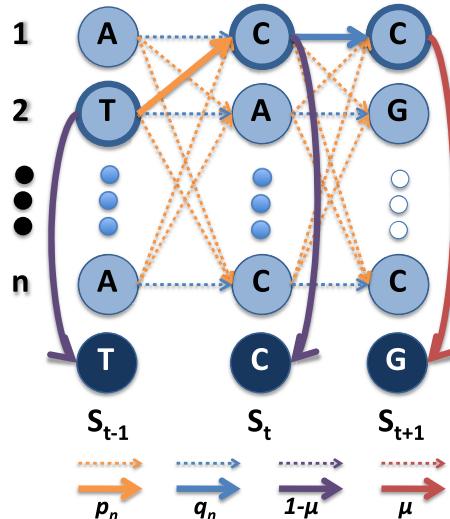
In our model, we assume that in each region, the target individual has similar haplotypes to the reference haplotypes. The goal of imputation can then be rephrased as assigning one (haploid model) or a pair of (diploid model) the  $n$  reference haplotypes to each of the  $m$  markers. From these assignments, we will assign the individual SNP genotype values to each uncollected SNP in the target individual by using the alleles of the assigned reference haplotypes at the nearby markers. We employ a hidden Markov model (HMM) to assign the reference haplotypes to the markers of the target individual. We describe the details of the HMMs used to solve the different cases of the imputation problem.

## 2.2 Imputation Algorithm for Inbred Strains (Haploid Model)

*Hidden Markov Model for imputation.* In the following section we describe the algorithm for performing imputation for haploid or inbred organisms. In this case, the reference haplotypes (or individuals) are called reference strains and we assume that we do not observe any genotypes where an individual has both alleles 1 and 2 of the SNP (i.e. no heterozygous genotypes).

The goal is then to assign one of the  $n$  reference strains to each of the  $m$  markers described in the previous section. To accomplish this we use an HMM like that shown in Figure 2. For each of the  $m$  markers there are  $n$  states corresponding to each of the reference strains. From each state there are  $n$  edges (with the exception of the states representing the final marker) directed towards the  $n$  states for the next marker. The edges corresponding to a change in the reference strain are called transitions or a recombination. Each state can also emit one of the two possible alleles for that marker. Emitting an allele that does not match the target strain is called a mutation.

Let  $S_t \in \{1, 2, \dots, n\}$  be the reference strain assigned to the target strain at marker  $t \in \{0, 1, \dots, m-1\}$ . Since some of genomic segments may not be represented by any of the reference strains, we introduce another reference strain consisting of only missing genotypes to represent the unknown reference state. We let the initial probability that marker  $S_0$  is assigned to strain  $i$  be  $\Pr(S_0 = i) = \pi_i$ , with  $\pi = \{\pi_1, \dots, \pi_n\}$ . Similar to many other methods designed for genotype imputation and haplotype phasing [11, 14], our HMM relies on a typical Markov



**Fig. 2.** An example of the hidden Markov model for the imputation problem

chain model of population genetics based on neutral Wright-Fisher model [9]. We use two parameters  $\mu$ , and  $\theta$  to be the mutation and recombination parameters, and use the standard distributions for computing the probability of transitioning between states  $p_n(x) = (1 - e^{-x})/n$  and  $q_n(x) = p_n(x) + e^{-x}$  based on the continuous-time Markov process. Figure 2 shows how these probabilities correspond to the edges in the HMM. The transition probabilities are computed from the recombination parameter and the distance between markers as follows.

$$\Pr(S_t = j | S_{t-1} = i, \theta) = \begin{cases} q_n(-\theta d_t) & i = j \\ p_n(-\theta d_t) & i \neq j \end{cases} \quad (1)$$

The probability of an observed genotype given a state is computed from the mutation parameter and the allele observed at the reference strain at the state. If the reference strain has missing genotype at the marker, then the probabilities are equally assigned between the alleles. Figure 2 shows examples of matching and mutated genotypes in the emission states of the HMM.

$$\Pr(g_t | S_t, \mu) = \begin{cases} 1 - \mu & g_t = G_{S_t, t} > 0 \\ \mu & g_t \neq G_{S_t, t} > 0 \\ 0.5 & G_{S_t, t} = 0 \end{cases} \quad (2)$$

We assume a uniform distribution of initial state probabilities,  $\pi_1 = \dots = \pi_n = 1/n$ , and learn the mutation and transition parameters from the data using the EM algorithm presented below.

*EM algorithm for learning maximum-likelihood parameters.* Many of the previous methods suggested for HMM-based imputation of missing genotypes use either predefined transitional parameters [11] or Viterbi training which may not

converge to a local maximum [15]. Other methods estimate the transition probabilities per marker interval independently to avoid the computational complexity in constraining the transition parameters consistently across different states [14][10].

In the genotype imputation of inbred mouse strains, independent unconstrained estimation of parameters at each marker is prone to inherent bias and inaccuracy because of two reasons. First, the total number of target strains is small, so estimating parameters per marker independently may be highly inaccurate. Second, these strains have complex genetic relationship, so the transitional and mutational parameters vary greatly across different strains. We constrain the parameters to be equal over the genome, but allows different transition and mutation parameters for each strain. Instead of the simple Viterbi training algorithm, we present an EM algorithm based on the exact conditional probabilities obtained from the forward-backward algorithm.

Let us denote  $X_t^- = (X_1, \dots, X_t)$  and  $X_t^+ = (X_{t+1}, \dots, X_m)$  as observed data, and  $\lambda = (\pi, \mu, \theta)$  be the initial, mutational, and transitional parameters of the hidden Markov model. The forward-backward algorithm estimates  $\alpha_t(i) = \Pr(X_t^-, S_t = i | \lambda)$  and  $\beta_t(i) = \Pr(X_t^+ | S_t = i, \lambda)$  using dynamic programming.

Let  $X = (g, G)$ . The EM algorithm starts with initial parameters  $(\mu_0, \theta_0)$ . At the E-step of  $r$ -th iteration,  $\Pr(S_t | X, \lambda_r)$  are computed from the forward-backward algorithm. Let  $S = \{S_0, \dots, S_{m-1}\}$ . At the M-step, the expected likelihood function can be written as follows.

$$\begin{aligned} Q(\mu, \theta) &= \sum_S \Pr(S | X, \lambda_r) \log \Pr(g, S | \lambda) \\ &= \sum_{t=1}^{m-1} \sum_{(S_t, S_{t-1})} \log \Pr(S_t | S_{t-1}, \theta) \Pr(S_t, S_{t-1} | X, \lambda_r) \\ &\quad + \sum_{t=0}^{m-1} \sum_{S_t} \log \Pr(g_t | S_t, \mu) \Pr(S_t | X, \lambda_r) - \log n \end{aligned} \tag{3}$$

The expectation-maximized parameters used for the next round of E-step can be obtained as follows.

$$\mu_{r+1} = \frac{\sum_{t=1}^m \sum_{S_t} I(g_t \neq G_{i,t}) \Pr(S_t | X, \lambda_r)}{\sum_{t=1}^m \sum_{S_t} I(G_{i,t} > 0) \Pr(S_t | X, \lambda_r)} \tag{4}$$

$$\theta_{r+1} = \arg \max_{\theta} \left[ \sum_{t=1}^{m-1} \sum_{(S_t, S_{t-1})} \log \Pr(S_t | S_{t-1}, \theta) \Pr(S_t, S_{t-1} | X, \lambda_r) \right] \tag{5}$$

$$= \arg \max F(\theta) \tag{6}$$

In order to estimate the joint probability  $\Pr(S_t, S_{t-1} | X, \lambda_r)$ , we introduce a silent state  $J_t$  between  $S_{t-1}$  and  $S_t$  with the following transition probabilities which keeps  $\Pr(S_t | S_{t-1})$  unchanged.

$$\Pr(J_t = (i, b) | S_{t-1} = i, \theta) = \begin{cases} q_n(-\theta d_t) & b = 0 \\ (n-1)p_n(-\theta d_t) & b = 1 \end{cases} \quad (7)$$

$$\Pr(S_t = j | J_t = (i, b), \theta) = \begin{cases} 1 & b = 0, \quad i = j \\ 1/(n-1) & b = 1, \quad i \neq j \end{cases} \quad (8)$$

The probabilities of the other transitions are set to zero. In general, the marginal probabilities of these silent states can be computed, in the following way. Let  $L$  be a silent state connecting  $S_{t-1}$  and  $S_t$  with  $\Pr(S_t | S_{t-1}, \lambda) = \sum_L \Pr(S_t | L, \lambda) \Pr(L | S_{t-1}, \lambda)$  unchanged. The forward and backward probability of any silent state  $L$  are defined as  $\alpha(L) = \sum_{S_{t-1}} \alpha_{t-1}(S_{t-1}) \Pr(L | S_{t-1})$  and  $\beta(L) = \sum_{S_t} \beta_t(j) \Pr(S_t | L) \Pr(X_t | S_t)$ . Then the objective function of M-step transitional parameter becomes

$$F(\theta) = \sum_{i=1}^{m-1} \left[ \log q_n(-\theta d_t) \sum_{i=1}^n \Pr(J_t = (i, 0) | X, \lambda_r) + \log p_n(-\theta d_t) \sum_{i=1}^n \Pr(J_t = (i, 1) | X, \lambda_r) \right] \quad (9)$$

This function can be numerically optimized using a Newton-Raphson algorithm.

*Imputation of uncollected genotypes.* Let  $h_t$  be the number of 'uncollected SNPs' that need to be imputed between marker  $(t-1)$  and  $t$ . An uncollected SNP is represented as  $(t, s)$ , where  $t \in \{1, 2, \dots, m-1\}$  and  $s \in \{1, \dots, h_t\}$ . Let  $T_{t,s} \in \{1, \dots, n\}$  be the state at an uncollected SNP  $(t, s)$ .

We again modify the HMM by adding a silent state  $T_{t,s}$  to the original HMM. The transition probabilities between  $S_{t-1}, T_{(t,s)}$ , and  $S_t$  is defined in the same way to to Equation 11 based on the distance between them. Let  $H_{i,t,s} \in \{0, 1, 2\}$  be the genotypes of  $i$ -th reference strain at the uncollected SNP  $(t, s)$ . Then distribution of the imputed genotype  $z_{t,s}$  at the uncollected SNP is estimated as follows.

$$\Pr(z_{t,s} | X, \lambda) = \begin{cases} (1-\mu) \sum_{i=1}^n I(H_{i,t,s} = z_{t,s}) p_{t,s}(i) & z_{t,s} > 0 \\ +\mu \sum_{i=1}^n I(H_{i,t,s} \neq z_{t,s}) p_{t,s}(i) & z_{t,s} = 0 \\ \sum_{i=1}^n I(H_{i,t,s} = 0) p_{t,s}(i) & z_{t,s} = 0 \end{cases} \quad (10)$$

where  $p_{t,s}(i)$  denotes the marginal probability of state  $i$  at the uncollected SNP  $(t, s)$ . When estimating leave-one-snp-out imputation accuracy, the same imputation methods are applied by pretending marker  $t$  has a silent state by ignoring the observed genotype.

*Improving computation time complexity.* A standard HMM implementation requires squared time complexity with respect to the number of individuals. It is possible to reduce the time complexity to be linear in the number of states, by leveraging the fact that the transition probabilities are uniform over different states. When  $t > 1$ ,  $\alpha_t(i)$  follows that

$$\alpha_t(S_t) = [\exp(-\theta d_t) \alpha_{t-1}(S_t) + p_n(-\theta d_t) \sum_{x=1}^n \alpha_{t-1}(x)] \Pr(X_t | S_t) \quad (11)$$

This can be computed in a constant time if  $\sum_{j=1}^n \alpha_{t-1}(x)$  are precomputed, so the computation of  $\alpha_t(S_t)$  over all states can be performed in linear time. In a similar way, the computation of  $\beta_t(S_t)$  and the computation over silent states are linear to the number of states.

### 2.3 Extension to Unphased Genotypes (Diploid Model)

When imputing human genotype data, the reference individuals are typically provided as phased haplotypes across a dense set of SNPs, and a number of unphased genotypes of a target individual are provided as a subset of SNPs. In this case, the state at each collected marker  $Z_t = (i, j)$  represents the combined states of each haplotype. Here we assume there are no missing alleles in the reference haplotypes because they are phased. However, missing alleles can also be handled in a similar way presented as in the haploid model. Their initial state probabilities are defined as  $\Pr(Z_0 = (i, j)) = \pi_{ij} = 1/n^2$ , and the transition probabilities are defined as follows.

$$\Pr(Z_t = (i, j) | Z_{t-1} = (k, l), \theta) = \begin{cases} q_n(-\theta d_t)^2 & i = k, j = l \\ p_n(-\theta d_t)q_n(-\theta d_t) & i = k \oplus j = l \\ p_n(-\theta d_t)^2 & i \neq k, j \neq l \end{cases} \quad (12)$$

where  $\oplus$  denotes exclusive OR operator. Let  $Z_t = (Z_{t,0}, Z_{t,1})$  be the individual states of each chromosome, then the imputed genotype  $(g_{t,0}|Z_{t,0}, G, g_{t,1}|Z_{t,1}, G)$  independently follows the mutational distribution in the inbred imputation.

The observed genotype  $g_t = \{g_{t,0}, g_{t,1}\} \in \{\{1, 1\}, \{1, 2\}, \{2, 2\}\}$  represent one of homozygous base alleles, heterozygous alleles, or homozygous mutant alleles. Based on these probability models, an HMM can be constructed with  $n^2$  states for each collected marker.

Let  $b_t$  be the number of state changes between marker  $t - 1$  and  $t$ . In order to estimate EM parameters, we introduce a silent state  $J_t$  connecting  $Z_{t-1}$  and  $Z_t$ , with the following transition probabilities.

$$\Pr(J_t = (k, l, b_t) | Z_{t-1} = (i, j)) = \begin{cases} q_n(-\theta d_t)^2 & k = i, l = j, b_t = 0 \\ 2(n-1)p_n(-\theta d_t)q_n(-\theta d_t) & k = i, l = j, b_t = 1 \\ (n-1)^2 p_n(-\theta d_t)^2 & k = i, l = j, b_t = 2 \end{cases} \quad (13)$$

$$\Pr(Z_t = (i, j) | J_t = (k, l, b)) = \begin{cases} 1 & k = i, l = j, b_t = 0 \\ 1/(2n-2) & (k = i \oplus l = j), b_t = 1 \\ 1/(n-1)^2 & k \neq i, l \neq j, b_t = 2 \end{cases} \quad (14)$$

From the expectation maximized parameters in the M-step it follows that

$$\mu_{r+1} = \frac{1}{m} \sum_{t=0}^{m-1} \sum_{Z_t} \eta(g_t, G_{Z_{t,0}, t}, G_{Z_{t,1}, t}) \Pr(Z_t | X, \lambda_r) \quad (15)$$

$$\theta_{r+1} = \arg \max_{\theta} \left[ \sum_{t=1}^{m-1} \sum_{(Z_t, Z_{t-1})} \log \Pr(Z_t | Z_{t-1}, \theta) \Pr(Z_t, Z_{t-1} | X, \lambda_r) \right] \quad (16)$$

$$= \arg \max_{\theta} G(\theta) \quad (17)$$

where  $\eta(g, h_1, h_2)$  is the number of mismatched alleles between genotype  $g$  and  $\{h_1, h_2\}$ .  $G(\theta)$  can be rewritten to be numerically optimized using a Newton-Raphson algorithm as follows.

$$\begin{aligned} \sum_{t=1}^m & [2 \log q_n(-\theta d_t) \Pr(b_t = 0 | X, \lambda_r) + \log (p_n(-\theta d_t) q_n(-\theta d_t)) \Pr(b_t = 1 | X, \lambda_r) \\ & + 2 \log (p_n(-\theta d_t)) \Pr(b_t = 2 | X, \lambda_r)] \end{aligned} \quad (18)$$

Similar to the inbred case, each uncollected genotypes is imputed without increasing memory by adding a silent state. For example,  $\alpha_t(Z_t)$  can be computed in a linear time with the number of states if we precompute  $\sum_{x=1}^n \alpha_t(Z_{t,b}, x)$  for each  $Z_{t,b}$  and  $\sum_{x=1}^n \sum_{y=1}^n \alpha_t(x, y)$  in order to increase the computational complexity.

## 3 Results

### 3.1 Genotype Imputation of 94 Inbred Mouse Strains

A recent NIEHS/Perlegen mouse resequencing project identified 8.27 million SNPs among 16 inbred mouse strains [5]. The Broad mouse HapMap project collected genotypes over 94 strains at 138,980 SNPs, which is only 1.7% of the number of SNPs identified in the resequencing project. We can achieve high imputation accuracy even with such a small fraction of the SNPs because of the very long regions of linkage disequilibrium.

We evaluated the accuracy of our genotype imputation method through leave-one-out analysis. For each of 16 resequenced strains, we ran our EMINIM algorithm to impute the genotypes at NIEHS/Perlegen SNPs using the mouse HapMap genotypes and the NIEHS/Perlegen SNPs of the rest 15 strains. Singleton SNPs polymorphic only in the target strain were removed in the evaluation of accuracy since they are not able to be imputed using the rest of strains. The leave-one-strain-out validation provides a conservative estimate of the genome wide imputation accuracy of a unresequenced strain using 16 resequenced strains.

The overall average imputation error over 12 classical strains is 2.40%. We classified the imputed genotypes into the 'high-confidence' category if the posterior probability is greater than 0.98, and 'medium-confidence' in between 0.8 and 0.98. When considering only high-confidence imputed genotypes after discarding 18.9% of low and medium confidence genotypes, the average imputation error

**Table 1.** Imputation error rates of inbred strains. First two rows (LOOCV) use leave-one-strain-out estimation using 15 reference strains and 138,980 combined SNPs of the target strain across 12 classical inbred strains. The unknown reference strain is used in the first but not in the second. Last two rows uses WTCHG genotypes as validation set, and impute those genotypes in 47 WTCHG strains not included in the reference strains. The fraction of imputed genotypes in each category is shown within a parenthesis.

	Confidence cutoff high (> 0.98)	high + medium (> 0.8)	all ( $\geq 0$ )
LOOCV with unknown reference	0.37% (81%)	0.81% (90%)	2.40% (100%)
LOOCV without unknown reference	0.52% (76%)	1.24% (93%)	2.46% (100%)
36 non-wild WTCHG strains	0.35% (89%)	1.00% (96%)	2.25% (100%)
all 47 WTCHG strains	0.37% (72%)	1.98% (89%)	4.86% (100%)

significantly reduces to 0.37%. When including wild-derived strains, the imputation error significantly increases. The average imputation error between four wild-derived strains was 19.2%, each of them ranging from 13.0% (WSB/EiJ) to 34.0% (CAST/EiJ). None of the wild-derived strains have high-confidence imputed genotypes due to high estimates of mutation rates.

Unlike previous imputation methods based on hidden Markov models, we introduce an additional state to account for genomic regions that are not explained by any of the reference strain. We compared this model to one without an additional state, by computing imputation accuracy using leave-one-strain-out cross-validation. The results over 12 classical inbred strains show that the overall imputation error increased from 2.40% to 2.46%. More notably, the average imputation errors in high confidence category increased from 0.37% to 0.52%, and the coverage of high-confidence category reduced from 81.1% to 75.7%. This suggests that our model with additional state for unknown reference strains significantly affects the imputation accuracy probably because some genomic segments are not well characterized by any of the 16 reference strains.

Next, we evaluated the imputation accuracy by comparing the genotypes typed for 78 non-resequenced strains. We used the Wellcome Trust genotypes as a validation set and evaluated how accurately our method can impute the genotypes in the validation set using the 16 resequenced strains as reference strains. 62 strains out of 94 strains were genotyped by Wellcome Trust, and 47 of them were not included in the 16 reference strains. A total of 493,033 genotypes in the validation set were evaluated for imputation accuracy, and the overall imputation error was 4.86%. 353,704 (71.7%) genotypes fall into high-confidence genotypes, and the imputation errors on these high-confidence genotypes are 0.37%. It should be noted that our imputation errors for high-confidence category is more than ten times smaller than the recently published results which used a different imputation method at a similar level of call-rate [15]. Their imputation errors at high-confidence genotypes were reported to be 4.4% with 69.5% call rate. When excluding eleven wild-derived strains the average error reduced to 2.26%, which is slightly lower than what we observed in 12 classical inbred strains with leave-one-strain-out cross-validation. Among the rest of 36 non-wild

strains, 344,747 (88.9%) genotypes out of 387,817 fall into the high-confidence category with an average imputation error of 0.35%, suggesting a high coverage of the mouse HapMap SNP sets with high imputation accuracy.

### 3.2 Imputation of HapMap SNPs in WTCCC Samples

We applied EMINIM to impute the uncollected HapMap SNPs of the 1,376 WTCCC control samples. We compared the imputation accuracy and memory efficiency with other published methods to demonstrate the robustness of our method. Our evaluation of chromosome 22 can be extrapolated to the estimate the performance of each method on a genome-wide scale.

First, we evaluated the accuracy of our method by randomly choosing 25% of SNPs out of the collected SNPs and imputing them from the rest of the collected SNPs. We varied the initial HMM transitional parameters of each method and observed the changes of the imputation accuracy to compare the adaptivity of the methods against the bias of the initial parameter. While IMPUTE shows a considerable change of imputation accuracy based on the transitional parameters, EMINIM shows almost the same accuracy regardless of the initial values of HMM parameters, because the optimal parameters are learned from the genotype and haplotype data using EM algorithm. The accuracy table consistently shows that our method has a higher accuracy than the previous methods (Table 2). The imputation accuracy of MACH was outperformed by EMINIM and IMPUTE. Since MACH does not use genetic map as input, we ran EMINIM using physical map instead of genetic map to compare their performance in the absence of genetic map, and EMINIM still showed a higher accuracy.

Next, we compared the memory efficiency between different algorithms. Since each imputation method requires a significantly large amount of memory to impute a large genomic region, the memory efficiency is an important issue when practically using the methods. The methods requiring too large amount of memory need to partition the chromosome into smaller segments at the expense of increased error rates and redundant computation. While IMPUTE requires 6.6GB of memory space, to impute all the polymorphic HapMap SNPs in chromosome 22, EMINIM requires only 508MB of memory, and MACH used 502MB of memory. This is mainly due to the fact that IMPUTE consumes memory space for each uncollected SNP while EMINIM requires memory space only for collected SNPs using a silent state when imputing each uncollected SNP. Such a difference may be substantial in a larger chromosome such as chromosome 1 which has more than five times as many SNPs as chromosome 22. In this case, EMINIM is expected to use 2.5GB of memory while IMPUTE may require 33GB of memory space. Such a difference may be more crucial as the number of reference samples increases. The overall CPU time of EMINIM was 4.7 hours with Intel Xeon E5320 Processor, which was faster than IMPUTE 0.5.0 (6.5 hours) and MACH 1.0 (7.2 hours with only 10 rounds), despite the fact that EMINIM runs HMM multiple times per individuals to estimate the EM parameters.

**Table 2.** Estimated error rates of each imputation method with different transition parameters across different confidence cutoffs.  $\theta = 3.8$  is suggested by Marchini et. al.[\[11\]](#), and different parameters are applied to demonstrate the effect of the initial parameters. The values in parenthesis represents the fraction of imputed genotypes with confidence above the threshold. Note that MACH does not provide the posterior probability at genotype level.

confidence cutoff	> 0.9	> 0.8	> 0.7	all
EMINIM ( $\theta_0 = 3.8$ )	1.23% (81%)	2.09% (87%)	3.07% (91%)	6.57% (100%)
EMINIM ( $\theta_0 = 0.38$ )	1.23% (81%)	2.09% (87%)	3.07% (91%)	6.57% (100%)
EMINIM ( $\theta_0 = 0.038$ )	1.23% (81%)	2.09% (87%)	3.07% (91%)	6.57% (100%)
IMPUTE ( $\theta = 3.8$ )	1.35% (81%)	2.25% (87%)	3.21% (91%)	6.61% (100%)
IMPUTE ( $\theta = 0.38$ )	2.79% (87%)	4.00% (91%)	5.04% (94%)	7.52% (100%)
IMPUTE ( $\theta = 0.038$ )	3.97% (88%)	5.19% (92%)	6.16% (95%)	8.23% (100%)
EMINIM (physical map)	1.50% (79%)	2.62% (86%)	3.78% (91%)	7.29% (100%)
MACH	N/A	N/A	N/A	7.69% (100%)

## 4 Conclusion

We have proposed an adaptive and memory efficient imputation method EMINIM. Our method adaptively learns HMM parameters using an exact EM algorithm. As a result, both in the human and inbred mouse strain imputation problems, our method is shown to outperform previous imputation methods specifically designed for each organism. In addition, the memory requirement of our method is independent of the number of uncollected SNPs by utilizing silent states, which significantly increase the scalability and computational efficiency of our method to genome-wide imputation.

## Acknowledgements

H.M.K., N.A.Z., B.H. and E.E. are supported by the National Science Foundation Grants No. 0513612, No. 0731455 and No. 0729049, and National Institutes of Health Grant No. 1K25HL080079. N.A.Z is supported by the Microsoft Research Fellowship. H.M.K and B.H. are supported by the Samsung Scholarship. This research was supported in part by the UCLA subcontract of contract N01-ES-45530 from the National Toxicology Program/National Institute of Environmental Health Sciences to Perlegen Sciences.

## References

1. Borevitz, J.O., Hazen, S.P., Michael, T.P., Morris, G.P., Baxter, I.R., Hu, T.T., Chen, H., Werner, J.D., Nordborg, M., Salt, D.E., Kay, S.A., Chory, J., Weigel, D., Jones, J.D., Ecker, J.R.: Genome-wide patterns of single-feature polymorphism in *Arabidopsis thaliana*. Proc. Natl. Acad. Sci. U.S.A. 104, 12057–12062 (2007)
2. Collins, F.S., Brooks, L.D., Chakravarti, A.: A DNA polymorphism discovery resource for research on human genetic variation. Genome Res. 8, 1229–1231 (1998)

3. de Bakker, P.I., Yelensky, R., Pe'er, I., Gabriel, S.B., Daly, M.J., Altshuler, D.: Efficiency and power in genetic association studies. *Nat. Genet.* 37, 1217–1223 (2005)
4. Devlin, B., Risch, N.: A comparison of linkage disequilibrium measures for fine-scale mapping. *Genomics* 29, 311–322 (1995)
5. Frazer, K.A., Eskin, E., Kang, H.M., Bogue, M.A., Hinds, D.A., Beilharz, E.J., Gupta, R.V., Montgomery, J., Morenzoni, M.M., Nilsen, G.B., Pethiyagoda, C.L., Stuve, L.L., Johnson, F.M., Daly, M.J., Wade, C.M., Cox, D.R.: A sequence-based variation map of 8. 27 million SNPs in inbred mouse strains 448, 1050–1053 (2007)
6. Gunderson, K.L., Steemers, F.J., Lee, G., Mendoza, L.G., Chee, M.S.: A genome-wide scalable SNP genotyping assay using microarray technology. *Nat. Genet.* 37, 549–554 (2005)
7. International HapMap Consortium. A second generation human haplotype map of over 3.1 million SNPs. *Nature* 449, 851–861 (October 2007)
8. Karlsson, E.K., Baranowska, I., Wade, C.M., Salmon Hillbertz, N.H., Zody, M.C., Anderson, N., Biagi, T.M., Patterson, N., Pielberg, G.R., Kulkoski, E.J., Comstock, K.E., Keller, E.T., Mesirov, J.P., von Euler, H., Kämpe, O., Hedhammar, A., Lander, E.S., Andersson, G., Andersson, L., Lindblad-Toh, K.: Efficient mapping of mendelian traits in dogs through genome-wide association. *Nat. Genet.* 39, 1321–1328 (2007)
9. Kingman, J.F.C.: On the genealogy of large populations. *Journal of Applied Probability* 19, 27–43 (1982)
10. Li, Y., Willer, C.J., Ding, J., Scheet, P., Abecasis, G.R.: Rapid Markov chain haplotyping and genotype inference (in submission) (2006)
11. Marchini, J., Howie, B., Myers, S., McVean, G., Donnelly, P.: A new multipoint method for genome-wide association studies by imputation of genotypes. *Nat. Genet.* 39, 906–913 (2007)
12. Matsuzaki, H., Dong, S., Loi, H., Di, X., Liu, G., Hubbell, E., Law, J., Berntsen, T., Chadha, M., Hui, H., Yang, G., Kennedy, G.C., Webster, T.A., Cawley, S., Walsh, P.S., Jones, K.W., Fodor, S.P., Mei, R.: Genotyping over 100,000 SNPs on a pair of oligonucleotide arrays. *Nat. Methods* 1, 109–111 (2004)
13. Risch, N., Merikangas, K.: The future of genetic studies of complex human diseases. *Science* 273, 1516–1517 (1996)
14. Scheet, P., Stephens, M.: A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am. J. Hum. Genet.* 78, 629–644 (2006)
15. Szatkiewicz, J.P., Beane, G.L., Ding, Y., Hutchins, L., de Villena, F.P.-M., Churchill, G.A.: An imputed genotype resource for the laboratory mouse. *Mamm. Genome* 19, 199–208 (2008)
16. The STAR Consortium. SNP and haplotype mapping for genetic analysis in the rat. *Nat. Genet.* 40, 560–566 (May 2008)
17. The Wellcome Trust Case Control Consortium. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls 447, 661–678 (2007)
18. Zaitlen, N., Kang, H.M., Eskin, E., Halperin, E.: Leveraging the HapMap correlation structure in association studies. *Am. J. Hum. Genet.* 80, 683–691 (2007)

# A Statistical Framework for the Functional Analysis of Metagenomes

Itai Sharon<sup>1</sup>, Amrita Pati<sup>2</sup>, Victor M. Markowitz<sup>3</sup>, and Ron Y. Pinter<sup>1</sup>

<sup>1</sup> Department of Computer Science, Technion, Haifa, Israel

<sup>2</sup> Genome Biology Program, DOE Joint Genome Institute, 2800 Mitchell Dr, Walnut Creek, CA 94598

<sup>3</sup> Biological Data Management and Technology Center, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

{itaish,pinter}@cs.technion.ac.il, {apati,VMMarkowitz}@lbl.gov

**Abstract.** Metagenomic studies consider the genetic makeup of microbial communities as a whole, rather than their individual member organisms. The functional and metabolic potential of microbial communities can be analyzed by comparing the relative abundance of gene families in their collective genomic sequences (metagenome) under different conditions. Such comparisons require accurate estimation of gene family frequencies. We present a statistical framework for assessing these frequencies based on the Lander-Waterman theory developed originally for Whole Genome Shotgun (WGS) sequencing projects. We also provide a novel method for assessing the reliability of the estimations which can be used for removing seemingly unreliable measurements. We tested our method on a wide range of datasets, including simulated genomes and real WGS data from sequencing projects of whole genomes. Results suggest that our framework corrects inherent biases in accepted methods and provides a good approximation to the true statistics of gene families in WGS projects.

**Keywords:** metagenomics, functional analysis, function comparison, Lander-Waterman.

## 1 Introduction

It has been estimated that less than 1% of the microbial species living on earth have been cultivated in the laboratory. Our inability to culture the vast majority of the rest is partially due to the complex conditions in which these organisms live, conditions that cannot be fully reconstructed in the lab. Metagenomics is a new and rapidly developing field that makes it possible to study uncultured organisms and their ecological systems. Several important discoveries have been made in recent years by extracting data directly from the environment, including the discovery of proteorhodopsin [1]. Metagenomic surveys, in which both sampling and sequencing are done randomly, provide a useful way to study microbial communities directly from the environment. In this type of projects researchers are usually interested in determining parameters related to community structure, such as the number of species and their diversity, as well as parameters related to functional capabilities of organisms in the

environment. Several metagenomic surveys were performed to-date in diverse environments such as the sea [2, 3, 6, 13, 14], acid mine drainage [4], human distal gut [5], and more.

Metagenomic studies present us with new computational challenges that are quite different from those of classical genomics. In most cases the amount and coverage of sequence data is insufficient to ensure assembly and classification of sequences into different microbial populations, thus preventing even limited population-specific genomic and metabolic reconstruction. Consequently, a prevalent method for analyzing metagenomic datasets is to compare the relative frequencies of gene families between datasets to highlight over- and under-represented functions in a given microbial community [11, 12]. Such comparisons require a measure of confidence in the observed differences in gene family frequencies between metagenomic datasets which are usually based on statistical tests. Statistical tests that have been applied to metagenome dataset comparisons are based on re-sampling [11, 12, 26]. These methods may produce reliable results but re-sampling does not scale well computationally with increased dataset sizes.

Recall that DNA sequencing – for both metagenomic and conventional genomic studies – is done using a process known as Whole Genome Shotgun (WGS) sequencing [20, 21, 22]. First, the extracted DNA is sheared randomly using physical or chemical means to fragments ranging in size between a few kilo base pairs (Kbps) to a few dozen Kbps. Next, the fragments are inserted into vectors which are used for constructing a genomic library. Last, clones from the library go through pair-end sequencing in which approximately 800bps from each side of each clone are sequenced using Sanger sequencing<sup>1</sup>. The outcome of this process is a set of many pair-end reads that can be used for all types of analysis including assembly, binning, and gene finding. Lander and Waterman [16] developed a statistical model for single-genome WGS sequencing projects that makes it possible to estimate parameters such as the expected number of contigs (continuous assembled reads) as a function of coverage depth, where coverage is the average number of reads covering each position in the genome.

Functional analysis of metagenomic data provides valuable insight into the kind of biological functions that are predominantly performed by organisms in a given environment  $E$ . One way of performing such an analysis is by aligning the metagenomic sequence data (also referred to as a “the metagenome”) against databases such as COG [8] or Pfam [9] and identifying the most abundant protein families in it (note that we use the terms “gene family” and “protein family” interchangeably, as implied by context). Once the frequency of each protein family in the metagenome has been estimated, it is possible to identify those families most essential to the survival of certain species in  $E$  by comparing them with the frequencies observed in other metagenomes taken from similar or different environments. This process, termed *function comparison*, has become a common routine in metagenomic works [6, 12, 25, 26] and has proven to be very useful. For example, DeLong *et al.* [6] collected metagenomic samples from seven depths ranging between 10 to 4000 meters from a Pacific

---

<sup>1</sup> Alternatively it is possible to sequence DNA directly from the raw sample using pyrosequencing. While Sanger sequencing is considered in this paper, the framework described is also applicable to pyrosequencing with slight modifications.

Ocean station near Hawaii. By performing function comparison on the seven samples using the COG database, the authors were able to identify gene families that are characteristic of certain depths, such as photosynthetic-related genes that are most abundant in shallow water. Other families had significant representation in all depths, suggesting that they represent functions required by a wide range of microbes regardless of their environment. This illustrates the importance of comparing metagenomes: high abundance of a gene family does not necessarily imply relevance to specific conditions in the environment from which it was obtained. By performing function-comparison it is possible to differentiate between environment-specific and environment-independent functions.

Frequency estimation of gene families is usually done using a process we term the *read-counts approach*. The frequency of a family  $P$  is given by the number of reads carrying members of  $P$  divided by the number of reads carrying members of any other family  $P'$  (see Section 1.1 below). While being simple and straightforward, this approach "favors" families composed of longer genes and assigns them higher frequencies. In order to see why, consider two equally abundant gene families  $P_1$  and  $P_2$  in which members of  $P_1$  are on average twice as long as members of  $P_2$ . Assuming that reads are sampled and sequenced uniformly across all positions in all genomes in the environment, then the above process should yield more reads containing portions of  $P_1$  than reads containing portions of  $P_2$  as a direct effect of the difference in lengths. In the hypothetical event of reads of length 1 (single base pair) the estimated frequency of  $P_1$  should be twice the estimated frequency of  $P_2$ . This bias, known as the *read-counts bias*, is not negligible, as the length of genes may vary between a few dozen base pairs (*e.g.* tRNA genes) and a few thousand base pairs (*e.g.* the photosynthetic genes *psaA* and *psaB* and many other genes).

The scope of our work is function analysis based on low-level function databases, in which each family represents a single functionality such as protein (*e.g.* COG, [8]) or domain (*e.g.* Pfam [9] and TIGRfam [10]). We present a statistical framework for estimating family frequencies that is based on the abovementioned Lander-Waterman model and explain how derived frequencies may be used for functional comparison of metagenomic datasets. In addition, we provide a novel method for assessing reliability of computed frequencies which can be used for removing seemingly unreliable measurements. We have tested our method on both simulated and real WGS sequencing data from projects of whole genomes. Our tests indicate a substantial improvement of our method over existing ones.

Our contribution is 3-fold: (i) correction of the read-counts bias which is present in all methods that have been suggested to-date; (ii) for the first time a complete theoretical statistical framework with reasonable assumptions is being suggested, and (iii) the most extensive testing performed to-date, both on synthetic as well as (again, for the first time) on real data.

## 1.1 Previous Work

Functional characterization of metagenomic data involves (i) identifying protein Coding Sequences (CDSs) in unassembled or partially assembled metagenomic sequences using an *ab initio* or evidence-based gene finder, then (ii) associating these CDSs with protein families, such as COGs, Pfams, and TIGRfams, and subsequently

(iii) comparing the relative abundance of protein families. Protein coding sequences are associated with protein families using BLAST against sequence databases such as COG, reverse position-specific BLAST (RPS-BLAST) against position specific scoring matrices (PSSMs, e.g. the CDD database [27]), or using Hidden Markov Models (HMMs, e.g. the Pfam and TIGRFam databases).

Several methods for function comparison have been proposed to date [6, 11, 17, 26]. They differ in the way they perform function comparison, but compute the frequency for a protein family  $P$  from the proportion of reads in a metagenomic sample  $M$  that are associated with  $P$  when  $M$  is compared to a function database using BLAST [24]. Observe that in such an approach, each read may be associated with multiple protein families, and hence, counted several times. We refer to this approach as the read-counts frequency computation approach; as mentioned above, this method tends to overestimate the frequencies of longer genes. The method described in [6] and [11] begins with the assessment of frequencies for each family as described above, and then computes the distance between the frequencies of each protein family in the two metagenomes using simulations. The computation of the  $p$ -value for this distance is also done using simulations. The method is supposed to be assumption-free, but the simulation process described is – in fact – a binomial one. In addition, the generation of the random distribution is somewhat ad-hoc. Finally, the simulations may make the process computationally intensive, possibly unnecessarily since the distributions may be computed directly considering the fact that the process is binomial.

The method employed in the IMG/M system [11] is also based on the computation of family-frequencies using the read-counts approach, but differs in its function comparison procedure. Given a protein family  $P$ , and two metagenomes  $M_1$  and  $M_2$  for which the respective frequencies  $f_1$  and  $f_2$  of association of  $P$  with reads has been computed, the method first computes the distance between  $f_1$  and  $f_2$  and an associated  $p$ -value for the distance. The  $p$ -value computation is based on the null hypothesis that the raw counts of occurrence of  $P$  among reads in both  $M_1$  and  $M_2$  can be approximated by a binomial distribution whose Bernoulli probability is computed as a pooled probability from the counts of occurrence of  $P$  in both metagenomes. Such a computation is biased towards the frequency of  $P$  in the larger metagenome, which is undesirable. Also the use of a common source distribution is likely to reduce the significance of the difference between the two metagenomes. The work described in [17] is a set of new statistical methods for comparing communities. Rather than using databases such as COG or Pfam, the authors employ clustering algorithms for putting together related proteins. This approach may extend functional analysis beyond known genes at the expense of the reliability of the results.

## 2 Methods

In this section, we propose a statistical model for computing the frequency of a protein family among reads for a given genome. This will serve as a more accurate replacement of the binomial proportion that has been widely used in the scientific literature to model protein family frequencies. The proposed statistical model derives its roots from the Lander-Waterman theory [16] on the statistics of WGS sequencing projects. The model is described with respect to a single genome; minor adjustments

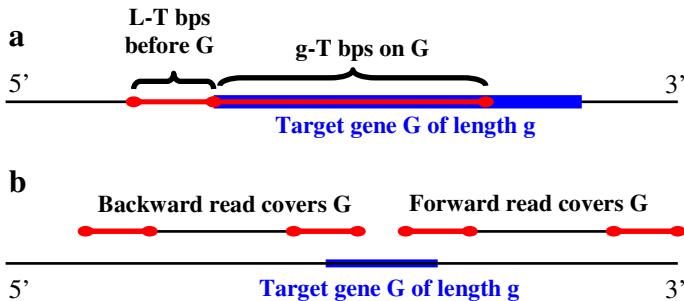
which are required in order to make the theory suitable for metagenomes are described later.

## 2.1 Estimating the Frequency of a Protein Family

Consider a WGS sequencing project in which  $N$  independent clones ( $2N$  pair-end reads) of average read length  $L$  are sequenced randomly from a genome of length  $\Gamma$ . Assume that each clone may start at any position in the genome with equal probability  $\alpha = \frac{N}{\Gamma}$ . This description is consistent with [16] and may be extended for metagenomes as will be shown later in Section 2.3. The number of clones starting at each position is a Poisson-distributed random variable with mean  $\alpha$ . The probability of  $j$  clones starting at a given position is:

$$f(j; \alpha) \sim \text{Poisson}(\alpha) = \frac{\alpha^j \cdot e^{-\alpha}}{j!} \quad (1)$$

Let  $G$  be a gene with length  $g$ . Assume that a read must contain at least  $T$  base pairs,  $L > T$ , of  $G$  in order for  $G$  to be detected. First, we estimate the number of reads containing a detectable part of  $G$ . Then, there are  $2(L+g-2T)$  positions at which a clone might begin (see Fig. 1) in order for  $G$  to be detected on one of its pair-end reads.



**Fig. 1.** (a) Any read of length  $L$  that begins within the  $L-T$  base pairs before the target gene or somewhere on the first  $g-T$  base pairs on the gene will cover at least  $T$  base pairs of the gene. Overall there are  $L+g-2T$  positions in which such reads may begin. (b) Each clone may cover  $G$  with either its forward or backward reads. Combined with (a) this gives a total of  $2(L+g-2T)$  positions.

The number of reads containing a detectable part of  $G$  can be represented by a random variable  $R_G$  that is the sum of  $2(L+g-2T)$  independent Poisson variables with mean  $\alpha$ .  $R_G$  is therefore Poisson distributed with mean  $\lambda_g = 2\alpha(L + g - 2T)$ .

The function analysis process is based on identifying genes that belong to gene families that are captured as COGs or Pfams. Each such family contains several genes whose lengths are usually similar. Let  $P$  be a protein family composed of genes whose average length is  $g$ . In the genome of length  $\Gamma$ , suppose that  $C_P$  genes are associated

with  $P$ . Assuming that the occurrences of the genes associated with  $P$  are independent of each other, the number of reads associated with such genes can be modeled by a random variable  $R_P$ , which is the sum of  $C_P$  independent Poisson variables, each with mean  $\lambda_g$ . As deduced above,  $R_P$  is Poisson distributed with mean

$$\lambda_P = 2\alpha(L + g - 2T) \cdot C_P \quad (2)$$

The numbers resulting from BLASTing the metagenome reads against the protein families database yield an empirical estimate for  $R_P$ , while we are interested in estimating  $C_P$  for computing the frequency of  $P$  from all genes in the genome of length  $\Gamma$ . From Equation 2 we have,

$$C_P = \frac{\lambda_P}{2\alpha \cdot (L + g - 2T)} \quad (3)$$

Substituting  $\lambda_P$  with  $R_P$ , we get an estimator  $\hat{C}_P$  for  $C_P$  when the value of  $R_P$  is known as follows:

$$\hat{C}_P = \frac{R_P}{2\alpha \cdot (L + g - 2T)} \quad (4)$$

In the case of metagenomes, all the required parameters for computing  $C_P$  are available, except for the total length  $\Gamma$  of the genome, required for the computation of  $\alpha$ .

Let  $D$  be the set of protein families such that reads are associated with members of  $D$ . Then, the proportion of reads,  $F_P$ , associated with a protein family  $P$  can be approximated by dividing  $\hat{C}_P$  by the total number of genes belonging to any gene family:

$$F_P = \frac{\hat{C}_P}{\sum_{Q \in D} \hat{C}_Q} = \frac{R_P}{2\alpha \cdot (L + g - 2T)} \Big/ \sum_{Q \in D} \frac{R_Q}{2\alpha \cdot (L + g_Q - 2T)} = \frac{R_P}{(L + g - 2T)} \Big/ \sum_{Q \in D} \frac{R_Q}{(L + g_Q - 2T)} \quad (5)$$

where  $g_Q$  is the average length of genes in protein family  $Q$ .

Once computed,  $F_P$  can be used for function analysis. As seen in Equation 5,  $\alpha$  is eliminated from the expression for  $F_P$  resulting in an expression comprising only known parameters. The denominator in this case covers all genes in the genome and should be accurate enough for WGS projects of reasonable size; the numerator depends on the observed number of reads for the gene family where higher read count means more accurate frequency estimation. A high read count is the result of longer genes and a higher number of occurrences of the gene family in the genome. Next we provide a method for estimating the accuracy of the observed frequency.

## 2.2 Computing Confidence Bounds

The estimation of  $F_P$  is based on the observed number of reads covering members of protein family  $P$ . In the event of significant inaccuracy in this estimation, later stages such as function comparison will also be affected and will yield incorrect conclusions.

Here, we provide a method for estimating a range of possible frequencies for the gene family which may have generated the observed counts with probability higher than some user-defined threshold  $\varepsilon$ . For abundant families this range is expected to be narrow, while for rare families this range is going to be wide.

Specifically, we need to compute  $F_P^{\min}$  and  $F_P^{\max}$  such that for every  $f \in [F_P^{\min}, F_P^{\max}]$ ,  $\Pr[R_P | f] \geq \varepsilon$ , and there exists no other  $f'$  such that  $f' \notin [F_P^{\min}, F_P^{\max}]$  and  $\Pr[R_P | f'] \geq \varepsilon$ . These upper and lower bounds on the frequencies can then be used for filtering gene families for which frequencies have ranges that are too wide.

We compute the interval in the following manner. Start with a Maximum Likelihood estimation  $\hat{\lambda}_P = \text{Observed}(R_P)$  for the parameter of the Poisson distribution in Equation 3. Iteratively look for factors  $c^{\min}$  and  $c^{\max}$  such that  $\Pr(R \geq R_P | c^{\min} \cdot \hat{\lambda}_P) < \varepsilon$  and  $\Pr(R \leq R_P | c^{\max} \cdot \hat{\lambda}_P) < \varepsilon$ . From Equation 2 it follows that

$$C_P \alpha = \frac{\lambda_P}{2(L + g - 2T)} \quad (6)$$

Therefore, multiplying  $\lambda_P$  by a constant  $c$  is equivalent to multiplying  $C_P$  by  $c$ , since  $\alpha$  is not changed, and a simple multiplication of the frequencies as described here is sufficient.

### 2.3 Transition from Genomes to Metagenomes

So far we have discussed genomes, while, in fact, we are interested in analyzing metagenomes. In this section we show that Equation 5 can be used for metagenomes as well. In order to see why, consider a metagenomic sample  $S$  containing  $m$  different species, where species  $i$  has a genome of length  $\Gamma_i$  and is represented by  $n_i$  members. As a first step in the WGS sequencing process all genomes in  $S$  are sheared into clones; next,  $N$  clones are chosen at random and undergo pair-end sequencing. Overall, there are  $n_i \Gamma_i$  base pairs in  $S$  associated with the genome of species  $i$ , and the total

length of all the genomes in  $S$  is  $\sum_{j=1}^m n_j \Gamma_j$ . As in the case of a single genome it is

assumed that a clone may begin at each position on the genome of any organism in  $S$ ; therefore, assuming that a total of  $N$  clones undergo pair-end sequencing, the expected

number of clones extracted from the genome of species  $i$  is  $N \cdot n_i \Gamma_i / \sum_{j=1}^m n_j \Gamma_j$ . From

Equation 1 it follows that the expected number of clones beginning at each position on the genome of species  $i$  is

$$\alpha_i^S = \left( \frac{N \cdot n_i \Gamma_i}{\sum_{j=1}^m n_j \Gamma_j} \right) \cdot \frac{1}{\Gamma_i} = \frac{N \cdot n_i}{\sum_{j=1}^m n_j \Gamma_j} \quad (7)$$

Given species  $i$  with genome of length  $\Gamma_i$  and  $C_p^i$  genes of average length  $g$  on this genome that are associated with protein family  $P$ , it follows from Equation 2 that the number of reads that cover genes associated with  $P$  in genome  $i$  ( $R_p^i$ ) is a Poisson random variable with mean

$$\lambda_p^i = 2\alpha_i^S (L + g - 2T) \cdot C_p^i \quad (8)$$

The total number of reads covering genes associated with  $P$  anywhere in the sample,  $R_p^S$ , can now be expressed as the sum of  $m$  Poisson random variables. Using Equation 7, this is a Poisson random variable with mean

$$\lambda_p^S = \sum_{i=1}^m \lambda_p^i = 2(L + g - 2T) \cdot \sum_{i=1}^m \alpha_i^S \cdot C_p^i = \frac{2N(L + g - 2T)}{\sum_{j=1}^m n_j \Gamma_j} \sum_{i=1}^m n_i \cdot C_p^i = \frac{2N(L + g - 2T)}{\sum_{j=1}^m n_j \Gamma_j} C_p^S \quad (9)$$

where  $C_p^S$  is the total number of genes associated with  $P$  in  $S$ . By replacing  $\lambda_p^S$  with  $R_p^S$ , which is the observed number of reads covering genes associated with  $P$  in  $S$ , we obtain an estimator  $\hat{C}_P^S$  for  $C_p^S$ :

$$\hat{C}_P^S = \frac{R_P^S \cdot \sum_{j=1}^m n_j \Gamma_j}{2N \cdot (L + g - 2T)} \quad (10)$$

Thus, the proportion of reads in  $S$  that are associated with  $P$  is given by

$$F_P^S = \frac{\hat{C}_P^S}{\sum_{Q \in D} \hat{C}_Q^S} = \frac{R_P^S \cdot \sum_{j=1}^m n_j \Gamma_j}{2N \cdot (L + g - 2T)} \Bigg/ \frac{R_Q^S \cdot \sum_{j=1}^m n_j \Gamma_j}{\sum_{Q \in D} 2N \cdot (L + g_Q - 2T)} = \\ = \frac{R_P^S}{(L + g - 2T)} \Bigg/ \sum_{Q \in D} \frac{R_Q^S}{(L + g_Q - 2T)} \quad (11)$$

For all practical purposes, Equation 11 is the same as Equation 5 since it does not contain any information related to  $S$ , such as the number of species and their genome lengths.

## 2.4 Performing Function Comparison

Given a reference metagenome  $B$  and a metagenome of interest  $A$ , for a protein family  $P$  we are interested in estimating the significance of the difference between the estimated frequencies of  $P$  in  $B$  and  $A$ ,  $F_P^B$  and  $F_P^A$ , respectively. Here we propose a novel approach for assessing the significance of the difference between the two frequencies.

Recall that  $F_P^B$  and  $F_P^A$  are derived from the observed number of reads containing some part of  $P$  in  $A$  and  $B$ ,  $R_P^B$  and  $R_P^A$ , respectively. We are interested in computing  $\text{equivalent}(R_P^A, B)$ , the number of reads in  $B$  that would yield the frequency  $F_P^A$ . Considering Equation 5 (or its metagenomic equivalent Equation 11), this goal can be formulated by

$$F_P^A = \frac{\text{equivalent}(R_P^A, B)}{(L + g - 2T)} \Bigg/ \sum_{Q \in D} \frac{R_Q^B}{(L + g_Q - 2T)} \quad (12)$$

This provides an explicit expression for  $\text{equivalent}(R_P^A, B)$ :

$$\text{equivalent}(R_P^A, B) = F_P^A \cdot (L + g - 2T) \cdot \sum_{Q \in D} \frac{R_Q^B}{(L + g_Q - 2T)} \quad (13)$$

Since all components of Equation 13 are available,  $\text{equivalent}(R_P^A, B)$  can be computed. Taking  $R_P^B$  as the maximum likelihood estimation for  $\lambda_P^B$ , the parameter of the Poisson variable describing the distribution of reads covering members of  $P$  in  $B$ , it is possible to compute the probability to observe at least (or at most)  $\text{equivalent}(R_P^A, B)$  reads covering  $P$  in  $B$ . This probability may be interpreted as a significance measure for the difference between the frequencies  $F_P^B$  and  $F_P^A$ .

## 3 Results

In order to analyze the performance of our model we have used three types of data: (i) simulated data of single genomes, (ii) real data from genome sequencing projects, and (iii) real metagenomic data. Simulated data is generated according to our assumptions, and therefore provides an opportunity to evaluate the method under controlled conditions. In addition, it is possible to generate different datasets with varying parameters such as gene family distributions, coverage, and population structure. As explained above, single genomes may be regarded as a special case of metagenomes; they are simpler to analyze and generate and were therefore used for evaluation. Genome sequencing projects provide the opportunity to test our framework on real data (generated in a similar way to metagenomes) with known gene family statistics. Real metagenomes lack information regarding the underlying statistics but provide us with

the opportunity to check real-world cases. Evaluation done using such data is mostly qualitative.

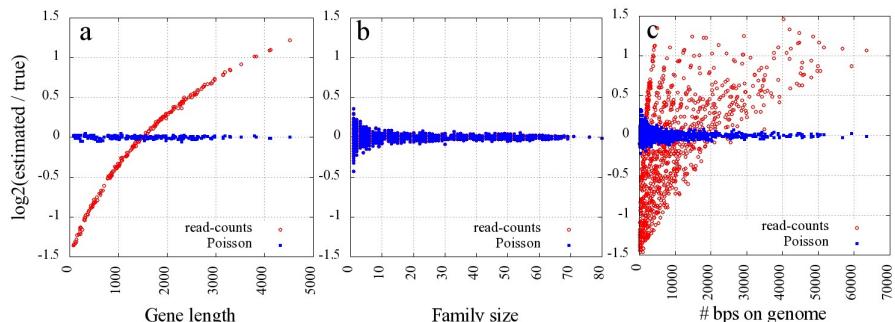
### 3.1 Evaluation of Simulated Data

We have written a program that generates simulated genomes and WGS sequencing projects, including genes and their families and reads' distribution across the genome. The simulator also implements our statistical framework, as well as the read-counts based frequency estimator for protein families. In order to compare the performance of the two frequency estimators we have used the following quality measure:

$$Q(M(P)) = \log_2 \frac{M(P)}{F_P^{True}} \quad (14)$$

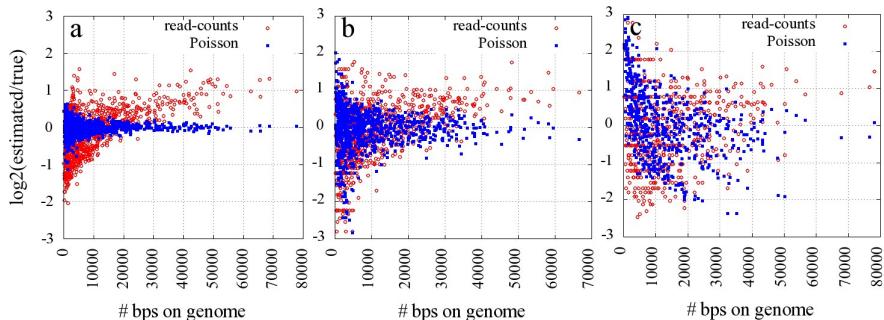
in which  $F_P^{True}$  is the true frequency of protein family  $P$  in the (meta)genome and  $M(P)$  is the estimation. When  $M(P) = F_P^{True}$  it is the case that  $Q(M(P)) = 0$ , while  $Q(M(P)) < 0$  when  $M(P)$  underestimates  $F_P^{True}$ , and  $Q(M(P)) > 0$  when  $M(P)$  overestimates  $F_P^{True}$ .

As a first step we have tested the read-counts bias. For this purpose we have synthesized three genomes of length 10 Mbps each: (i) a genome that contains genes associated with 200 protein families of lengths ranging between a few dozens to a few thousands bps and a similar number of genes associated to each family, (ii) a genome containing genes associated with several hundreds of protein families of constant length but a different number of genes associated with each family, and (iii) a combination of (i) and (ii) – genomes with protein families of varying lengths and varying number of associated genes. Other than the length and the number of genes associated with the protein families all other parameters remained constant across all simulated genomes, including the number of reads (1M) and their lengths (942 bps) and the distance between genes (60 bps). Note that the resulting coverage is extremely high; this was done in order to test the behavior of each method when all data is available.



**Fig. 2.** Quality of frequency estimations as a function of (a) gene size, (b) family size, and (c) a combination of both. In all cases coverage is extremely high (94.2X) in order to demonstrate the inherent differences between the methods even when redundant data is available.

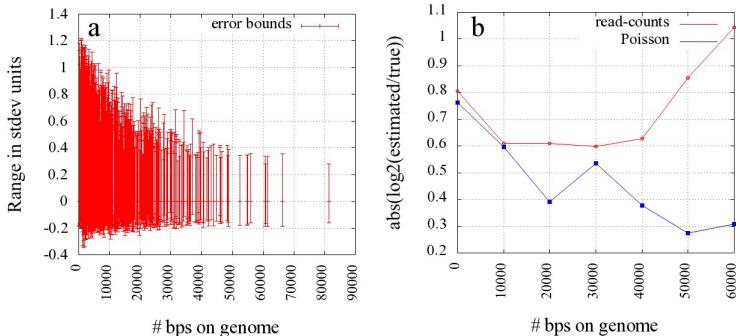
Fig. 2 summarizes the results of these three tests. As can be seen from Fig. 2a, our framework scales with any gene length while read-counts based estimations tend to overestimate the frequencies of long genes and to underestimate the frequencies of short genes. When all genes across the genome are of exactly the same length both methods yield the same estimations, regardless of differences in the number of genes associated with each family (Fig. 2b). This is not surprising considering the fact that when  $g_i=g_j$  for any two families  $i$  and  $j$ , Equation 5 is reduced to simple division of read-counts with no other scaling required. As can be seen from Fig. 2b, frequency estimations are less accurate for small families, which is also reasonable. Our method remains reliable also when both the number of genes associated with a family and their lengths vary (Fig. 2c), while the read-counts approach produces inaccurate estimations. The accuracy of the method increases when more bps are associated with family  $P$ . A large number of bps associated with family  $P$  may be the result of both long genes, or a large amount of genes associated with the family.



**Fig. 3.** The influence of coverage on frequency estimation for (a) coverage=9.42X, (b) 0.942X and (c) 0.0942X

Next, we were interested in analyzing the influence of different coverage levels on the quality of the estimations. For this purpose we have used the setup of the genome whose behavior is described in Fig. 2c and tried a different number of reads each time (resulting in different coverage levels). At high coverage levels typical to genome sequencing projects (Fig. 3a) our method remains relatively accurate. As expected, the quality of the estimations decreases with lower coverage levels (Fig. 3, b and c). In all cases our method outperforms the read-counts estimator and provides significantly better results (see Fig. 4b).

Fig. 4a shows confidence bounds computed for different families as a function of the number of bps associated with the family. As can be clearly seen, the confidence range decreases (in terms of standard deviation units) with the more associated bps available. This property of the confidence bounds scheme is desirable, considering the fact that our method derives better frequency estimations for protein families to which



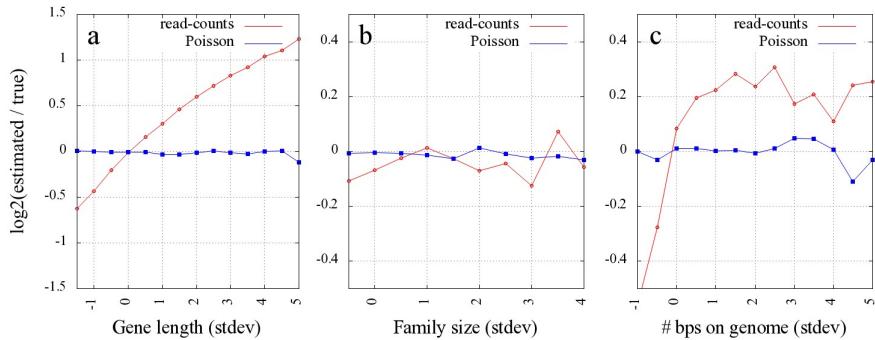
**Fig. 4.** (a) Reliability bounds as a function of the number of base pairs associated with protein families,  $\epsilon=0.1$  used. (b) Log-ratio error as a function of the number of base pairs associated with different families. For each range  $[x \cdot 10\text{Kbps}, (x+1) \cdot 10\text{Kbps}]$  median log-ratio error for all families in the range is shown. Confidence bounds become narrow with the increase in the family's share on the genome, in line with the increased accuracy of frequency estimation.

many bps are associated across the genome (Fig. 4b). This can also be used for filtering families with a wide confidence range.

### 3.2 Evaluation of Data Derived from WGS Sequencing Projects

Raw data of WGS sequencing projects of single genomes is available through NCBI's Trace DB. Using data from [18] we were able to reconstruct read coordinates and also to identify pairs of reads that belong to the same clones. Overall we were able to reconstruct information for 44 genomes (see Appendix for complete list of genomes). COG families were assigned to these genomes by BLASTing (blastx) their genes (extracted from the IMG system [7]) against the COG database with an e-value threshold of  $1e-50$ . Next the number of reads containing members of each COG family was decided and used for producing frequency estimations by both our and the read-counts approach. Evaluation of estimations was done using Equation 14 as before. Results are summarized in Fig. 5 (see figure legend for description of figure generation process). As can be observed from the figure, behavior of both estimation methods fit nicely the behavior that was predicted in our simulations. While our method generated predictions that are, on average, close to the real frequencies and are not affected by the length of family members, the read-counts approach produces estimations that are biased by gene length. Both methods are not affected by the number of family members.

It is important to note that while our method produces unbiased estimations, its frequency estimations are, on average, 20-30% off the real value for all gene lengths. The estimation error is affected by the number of base pairs associated with a family on the genome (more base pairs yield better estimations). This error rate is smaller than or equal to the error of the read-counts approach for all cases.



**Fig. 5.** Summary of data analysis for 44 single genome WGS projects.(a) Log-ratio between estimations and true frequencies, sorted by family length. For each genome, average and standard deviation of gene lengths were computed and used for binning evaluations into  $x0.5$  standard deviation bins. Next the median for each bin was computed and kept with medians of the same bin from other genomes. Last the median over all medians for each bin was computed and used for generating the figure. (b) Same process as before using number of members for each family as the key. (c) Same process, using number of base pairs occupied by each family on the genome as the key.

### 3.3 Evaluation of Real Data

While metagenomes are the goal of our method, it is impossible to validate the results generated from them. However, in order to get a sense of what the results look like we have compared two depths datasets (10 and 500 meters) from [6]. In order to assign genes to protein families we have used the COG database: each read was BLASTed against the COG database with the best hit being assigned to the read. The 10 meters dataset is composed of 7,842 reads whose average length is 954 bps, while the 500 meters dataset consists of 9,027 reads with average length of 971 bps. Overall a total of 2,426 genes from the 10 meters dataset have been assigned to COG families and 2,997 genes from the 500 meters were similarly assigned.

While family frequencies usually show correlation with the number of hits, there are many exceptions: for example, COG0187 was found 16 times in the 10 meters dataset but was assigned a frequency of 0.53%, less than the frequency of COG1024 with only 13 hits (0.59%). While COG1024's average length is 270 bps the length of COG0187 is 685 bps which explains why it received more hits. The frequency of COG0085 with 17 hits in the 500m dataset is 0.35%, less than half the frequency of COG0316 (0.77%) with almost the same number of hits (18). Again, the difference is the result of gene lengths. Using the 500 meters dataset for the background distribution we were able to discover COG families whose frequency significantly differs between the two datasets. The results are not always observable by simply considering the read-counts; for example, the  $p$ -value assigned to the difference between the frequencies of COG0719 (ABC-type transport system involved in Fe-S cluster assembly, permease component) was quite significant ( $7 \cdot 10^{-5}$ ) despite a non-impressive difference in the read-counts (9 vs. 3 in the 10 and 500 meter datasets, respectively). Such cases convincingly demonstrate the need for a reliable statistical model when doing function analysis.

## 4 Discussion

All the methods suggested to-date for the functional comparison of metagenomes suffer from a read-counts bias. Our method is the first one to correct this problem, allowing for more reliable and credible analyses of this kind of data. This is achieved by offering a comprehensive statistical framework, based on sound theoretical foundations, that – with reasonable assumptions – allows us to assess the significance of the evidence for the presence of a protein family of interest in a given genome. Moreover, we present the most extensive evaluation performed to-date of such methods, which includes – for the first time – validation on real data. Our results suggest that the proposed framework provides a good approximation to the statistics of gene families in WGS projects. Observed estimation errors may be explained by reasons that are related to the definition of gene families. For example, the assignment of a gene to a family is hardly ever to the whole length of the family but rather to some part of it. This may result from the target family carrying several domains, while the assigned gene may carry only one of them. However, when we compute the expected frequency we consider the whole family's length, which is clearly unrealistic. The use of domain-specific databases, such as Pfam, may address this problem and improve frequency estimations.

Whereas our framework is most suitable for gene family based functional analysis, we believe that it is not suitable for databases organized according to high-level functionalities such as pathways (e.g. KEGG, [23]) or subsystems (e.g. SEED, [15]). This is also true for other methods described in the past, including read-counts based approaches, and is the result of the complex nature of these databases. Each family-equivalent in these databases is composed of several low-level functions, some of which may be shared by several pathways or subsystems. Moreover, the meaning of the results is unclear since we do not know how to interpret likely cases in which only some of the members of some high-level organization are present. Therefore it is necessary to better define the problem for these cases and develop an appropriate framework.

**Acknowledgments.** We would like to thank Rotem Sorek for sharing his data and to Zohar Yakhini, Ernest Szeto, and Konstantinos Mavromatis for fruitful discussions. The work presented in this article was partially supported by the Director, Office of Science, Office of Biological and Environmental Research, Life Sciences Division, US Department of Energy under Contract No. DE-AC03-76SF00098.

## References

1. Beja, O., Aravind, L., Koonin, E.V., Suzuki, M.T., Hadd, A., et al.: Bacterial Rhodopsin: Evidence for a New Type of Phototrophy in the Sea. *Science* 289(5486), 1902–1906 (2000)
2. Venter, J.C., Remington, K., Heidelberg, J.F., Halpern, A.L., Rusch, D., et al.: Environmental Genome Shotgun Sequencing of the Sargasso Sea. *Science* 304(5667), 66–74 (2004)
3. Angly, E.A., Felts, B., Salamon, P., Edwards, E.A., Carlson, C., et al.: The Marine Viromes of Four Oceanic Regions. *PLoS Biol.* 4(11) (2006)
4. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., et al.: Community Structure and Metabolism through Reconstruction of Microbial Genomes from the Environment. *Nature* 428(6978), 37–43 (2004)
5. Gill, S.R., Pop, M., Deboy, R.T., Eckburg, P.B., Turnbaugh, P.J., et al.: Metagenomic Analysis of the Human Distal Gut Microbiome. *Science* 312(5778), 1355–1359 (2006)

6. DeLong, E.F., Preston, C.M., Mincer, T., Rich, V., Hallam, S.J., et al.: Community Genomics among Stratified Microbial Assemblages in the Ocean's Interior. *Science* 311(5760), 496–503 (2006)
7. Markowitz, V.M., Szeto, E., Palaniappan, K., Grechkin, Y., Chu, K., et al.: The Integrated Microbial Genomes (IMG) System in 2007: Data Content and Analysis Tool Extensions. *Nucleic Acids Res.* 36(Database Issue), DS528–DS533 (2008)
8. Tatusov, R.L., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Kiryutin, B., et al.: The COG Database: an Updated Version Includes Eukaryotes. *BMC Bioinformatics* 4, 41 (2003)
9. Finn, R.D., Tate, J., Mistry, J., Coggill, P.C., Sammut, J.S., et al.: The Pfam Protein Families Database. *Nucleic Acids Res.* 36(Database Issue), D281–D288 (2008)
10. Haft, D.H., Selengut, J.D., White, O.: The TIGRFAMs Database of Protein Families. *Nucleic Acids Res.* 31, 371–373 (2003)
11. Rodriguez-Brito, B., Rohwer, F., Edwards, R.A.: An Application of Statistics to Comparative Metagenomics. *BMC Bioinformatics* 20(7), 162 (2006)
12. Tringe, S.G., von Mering, C., Kobayashi, A., Salamov, A.A., Chen, K., et al.: Comparative Metagenomics of Microbial Communities. *Science* 308(5721), 554–557 (2005)
13. Rusch, D.B., Halpern, A.L., Sutton, G., Heidelberg, K.B., Williamson, S., et al.: The Sorcerer II Global Ocean Sampling Expedition: Northwest Atlantic through Eastern Tropical Pacific. *PLoS Biol.* 5(3), e77 (2007)
14. Yooseph, S., Sutton, G., Rusch, D.B., Halpern, A.L., Williamson, S.J., et al.: The Sorcerer II Global Ocean Sampling Expedition: Expanding the Universe of Protein Families. *PLoS Biol.* 5(3), e16 (2007)
15. Overbeek, R., Begley, T., Butler, R.M., Choudhuri, J.V., Chuang, H.Y., et al.: The Subsystems Approach to Genome Annotation and its Use in the Project to Annotate 1000 Genomes. *Nucleic Acids Res.* 33, 5691–5702 (2005)
16. Lander, E.S., Waterman, M.S.: Genomic Mapping by Fingerprinting Random Clones: a Mathematical Analysis. *Genomics* 2(3), 231–239 (1988)
17. Schloss, P.D., Handelssman, J.: A Statistical Toolbox for Metagenomics: Assessing Functional Diversity in Microbial Communities. *BMC Bioinformatics* 9(34) (2008)
18. Sorek, R., Zhu, Y., Creevey, C., Francino, M.P., Bork, P., Rubin, E.M.: Genome-wide Experimental Determination of Barriers to Horizontal Gene Transfer. *Science* 318(5855), 1449–1452 (2007)
19. Mavromatis, K., Ivanova, N., Barry, K., Shapiro, H., Goltsman, E., et al.: Use of Simulated Data Sets to Evaluate the Fidelity of Metagenomic Processing Methods. *Nature Methods* 4, 495–500 (2007)
20. Sanger, F., Coulson, A.R., Hong, G.F., Hill, D.F., Petersen, G.B.: Nucleotide Sequence of Bacteriophage Lambda DNA. *J. Mol. Biol.* 162, 4 (1982)
21. Fleischmann, R.D., Adams, M.D., White, O., Clayton, R.A., Kirkness, E.F., et al.: Whole-genome Random Sequencing and Assembly of *Haemophilus influenzae* Rd. *Science* 269(5223), 496–512 (1995)
22. Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., et al.: The Sequence of the Human Genome. *Science* 291(5507), 1304–1351 (2001)
23. Kanehisa, M., Goto, S.: KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.* 28, 27–30 (2000)
24. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic Local Alignment Search Tool. *J. Mol. Biol.* 215, 403–410 (1990)
25. Martín-Cuadrado, A.B., López-García, P., Gottschalk, G., Rodríguez-Valera, F.: Metagenomics of the Deep Mediterranean, a Warm Bathypelagic Habitat. *PLoS ONE* 2, 914 (2007)

26. Warnecke, F., Luginbuhl, P., Ivanova, N., Ghassemian, M., Richardson, T.H., et al.: Metagenomic and Functional Analysis of Hindgut Microbiota of a Wood Feeding Higher Termite. *Nature* 450, 560–565 (2007)
27. Marchler-Bauer, A., Anderson, J.B., Chitsaz, F., Derbyshire, M.K., DeWeese-Scott, C., et al.: Specific Functional Annotation with the Conserved Domain Database. *Nucleic Acids Res.* 37(Database Issue), D205–D210

## Appendix: Genomes Used for Real Data Analysis

**Table 1.** Information for participating genomes in the real data analysis

Organism	Genome Size	Coverage	# Clones	Avg. Read Length
Acidobacterium sp. Ellin 345	5650368	11.1	27373	1035.9
Alkalilimnicola ehrlichei MLHE-1	3275944	12.8	16913	1014.4
Anabaena variabilis ATCC29413	6365727	17.1	36406	934.8
Anaeromyxobacter dehalogenans 2CP-C	5013479	12.0	15691	1007.5
Bacillus anthracis str ames	5227293	20.6	38924	856.4
Baumannia cicadellinicola	686194	29.0	2394	980.0
Campylobacter jejuni rm1221	1777831	12.2	9316	881.1
Carboxydothermus hydrogenoformans z-2901	2401520	14.8	12188	910.7
Chlorobium tepidum tls	2154946	13.8	10995	814.2
Colwellia psychrerythraea 34h	5373180	11.5	27652	902.2
Dechloromonas aromatica RCB	4501104	24.9	59247	850.7
Dehalococcoides ethenogenes 195	1469720	16.5	4861	907.3
Ehrlichia canis Jake	1315030	23.1	8229	968.3
Ehrlichia chaffeensis str Arkansas	1176248	14.6	7631	925.8
Frankia sp. CcI3	5433628	15.5	29142	990.6
Geobacter sulfurreducens pca	3814139	14.2	18579	916.6
Listeria monocytogenes str 4b f2365	2905187	17.7	21550	929.7
Methanococcoides burtonii DSM6242	2575032	24.0	34876	916.8
Methanospirillum hungatei JF-1	3544738	13.0	18776	958.9
Methylococcus capsulatus str bath	3304561	11.2	16517	884.3
Moorella thermoacetica ATCC 39073	2628784	23.5	21032	989.5
Neorickettsia sennetsu str miyayama	859006	13.7	4996	917.8
Nitrobacter hamburgensis X14	4406967	12.5	20259	935.3
Nitrobacter winogradskyi Nb-255	3402093	11.3	14703	958.4
Nitrosococcus oceanii ATCC 19707	3481691	14.4	20047	970.0
Prochlorococcus marinus sp. NATL2A	1842899	11.0	10561	960.4
Prochlorococcus marinus str. MIT 9312	1709204	13.6	9834	965.9
Prochlorococcus sp. CC9605 (oligotrophic)	2510659	17.0	16269	969.6
Pseudoalteromonas atlantica T6c	5187005	10.0	21681	1035.6
Psychrobacter cryohalolentis K5	3059876	12.2	13397	953.8
Rhodopseudomonas palustris BisB18	5513844	11.1	25015	1023.6
Rhodopseudomonas palustris BisB5	4892717	11.4	20142	1024.8
Rhodopseudomonas palustris HaA2	5331656	11.7	25012	959.5
Rubrobacter xylanophilus DSM 9941	3225748	14.5	14321	1193.9
Shewanella denitrificans OS217	4545906	16.6	27125	967.2
Shewanella frigidimarina NCMB400	4845257	11.9	23999	978.6
Shewanella sp. MR-4	4706287	11.5	21464	1020.4
Streptococcus agalactiae a909	2127839	13.9	12135	914.9
Synechococcus sp. PCC 7942 (elongatus)	2695903	13.0	15863	891.1
Thermobifida fusca YX	3642249	31.1	39988	850.3
Thiomicrospira crunogena, XCL-2	2427734	18.5	18675	955.0
Thiomicrospira denitrificans ATCC 33889	2201561	12.0	9992	958.4
Treponema denticola atcc 35405	2843201	15.6	16935	901.9
Trichodesmium erythraeum	7750108	17.3	49207	966.3

# Learning Models for Aligning Protein Sequences with Predicted Secondary Structure

Eagu Kim\*, Travis Wheeler, and John Kececioglu

Department of Computer Science  
The University of Arizona, Tucson AZ 85721, USA  
`{egkim,twheeler,kece}@cs.arizona.edu`

**Abstract.** Accurately aligning distant protein sequences is notoriously difficult. A recent approach to improving alignment accuracy is to use additional information such as predicted *secondary structure*. We introduce several new models for scoring alignments of protein sequences with predicted secondary structure, which use the predictions and their confidences to modify both the substitution and gap cost functions. We present efficient algorithms for computing optimal pairwise alignments under these models, all of which run in near-quadratic time. We also review an approach to learning the values of the parameters in these models called *inverse alignment*. We then evaluate the accuracy of these models by studying how well an optimal alignment under the model recovers known benchmark reference alignments. Our experiments show that using parameters learned by inverse alignment, these new secondary-structure-based models provide a significant improvement in alignment accuracy for distant sequences. The best model improves upon the accuracy of the standard sequence alignment model for pairwise alignment by as much as 15% for sequences with less than 25% identity, and improves the accuracy of multiple alignment by 20% for difficult benchmarks whose average accuracy under standard tools is less than 40%.

**Keywords:** Sequence alignment, protein secondary structure, inverse parametric alignment, substitution score matrices, affine gap penalties.

## 1 Introduction

While sequence alignment is one of the most basic and well-studied tasks in computational biology, accurate alignment of distantly related protein sequences remains notoriously difficult. Accurately aligning such sequences usually requires multiple sequence alignment, and a succession of ideas have been employed by modern multiple alignment tools to improve their accuracy, including: (a) *hydrophobic gap penalties*, which modify the alignment score to avoid gaps in regions that may be in the structural core, and are employed by CLUSTAL W [34], T-Coffee [28], and MUSCLE [10]; (b) *polishing*, which refines the alignment by

---

\* Corresponding author. Current address: Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison WI 53706. `egkim@biostat.wisc.edu`

realigning subgroups, and is used by MAFFT [18], MUSCLE, ProbCons [7], and Opal [36]; and (c) *consistency*, both in its combinatorial [28] and probabilistic [9,7] settings, which favors matches in the alignment that have high support, and is employed by T-Coffee, MAFFT, ProbCons, SPEM [39], PROMALS [29], and ISPALign [23]. These techniques all operate on the input sequences alone, and do not require additional sources of information.

Recent techniques that recruit additional information beyond the input sequences, and that have afforded significant boosts in accuracy, include: (d) *sequence profiles*, which augments the input residues with profiles of amino acid exchanges from closely related sequences found through database searches, and is used by SPEM, PRALINE [32], PROMALS, and ISPALign; (e) *intermediate sequences*, which adds new sequences to the alignment that link distant input sequences through chains of similar sequences, and is employed by MAFFT and ISPALign; and (f) predicted *secondary structure*, which annotates the input residues with their predicted structural type and modifies the scoring of alignments to take these types into account, used by SPEM, PRALINE, PROMALS, and ISPALign. These latter techniques, including secondary structure prediction, all involve database searching to find sequences that are closely related to the input sequences, which adds considerable overhead to the running time for alignment.

Of these latter techniques, incorporating secondary structure seems to have the greatest scope for further improvement. The scoring models based on secondary structure employed by current tools [23,29,39,32] do not make full use of predicted structural information when modifying the scores of substitutions and gaps. Furthermore, recent advances in single-sequence secondary structure prediction [1] suggest it may eventually be possible to make sufficiently accurate predictions based on the input sequences alone, which would yield improved alignment accuracy without the slowdown caused by sequence database searches.

In this paper, we introduce several new models for protein sequence alignment based on predicted secondary structure, and show they yield a substantial improvement in accuracy for distant sequences. These models improve on those used by current tools in several ways. They explicitly take into account the confidences with which structural types are predicted at a residue, and use these confidences in a rigorous way to modify both the scores for substitutions and the penalties for gaps that disrupt the structure. Furthermore, optimal alignments under these new models can be efficiently computed. Prior models tend to either have a limited number of ad hoc parameters that are set by hand [23,32,39], or have a large number of parameters that are estimated by counting frequencies of observed events in comparatively small sets of reference alignments [24,29,33]. Our new models have multiple parameters whose values must be set, and we show that recently developed techniques for parameter learning [19,21,22] can be used to find values for these parameters that are optimal in a well-defined sense. Finally, experimental results using our models show that our best model improves on the accuracy of the standard model by 15% for pairwise alignment of sequences with less than 25% identity, and by 20% for multiple alignment of difficult benchmarks whose accuracy under standard tools is less than 40%.

**Related work.** Structure is often conserved among related proteins even when sequence similarity is lost [3], so incorporating structure has the potential to improve the accuracy of aligning distant sequences. While deducing full three-dimensional structure from protein sequences remains challenging, accurate tools are available for predicting secondary structure from protein sequences [17]. Predicted secondary structure can be incorporated into the alignment model by encouraging substitutions between residues of the same secondary structure type, and discouraging gaps that disrupt regions of secondary structure.

The first work on incorporating secondary structure into the alignment scoring model appears to be by Lüthy, McLachlan and Eisenberg [24], who applied the log-odds scoring methodology of Dayhoff et al. [6] to derive substitution score matrices that take both the amino acids and the secondary structure types of the aligned residues into account when scoring a substitution. For the three secondary structure types of  $\alpha$ -helix,  $\beta$ -strand, and other, Lüthy et al. derive three log-odds substitution score matrices, where a given matrix applies when both of the aligned residues have the *same* structural type. In particular, their work does not provide matrices that apply to substitutions between residues with differing structural types. Moreover, the log-odds methodology for substitution scores does not provide appropriate gap penalties for these matrices.

Modern multiple alignment tools that take predicted secondary structure into account include PRALINE [32], SPEM [30], PROMALS [30], and ISPALign [23]. PRALINE uses the three substitution matrices of Lüthy et al. [24] when aligned residues have the same predicted structure type, plus the BLOSUM62 matrix [16] when they have differing types. PRALINE also employs four pairs of hand-chosen affine gap penalties (a gap open and extension penalty), one pair per matrix. SPEM modifies a standard substitution matrix by adding a bonus of  $x$  to the substitution score when aligned residues have the same type, and a penalty of  $-x$  when they have differing types, where  $x$  is a single hand-chosen constant. No gaps are permitted following a residue predicted to be in an  $\alpha$ -helix or  $\beta$ -strand; otherwise, a fixed gap open and extension penalty is used. PROMALS employs a hidden Markov model approach where match, insert, and delete states also emit secondary structure types. Emission and transition probabilities for each of these states are set by counting frequencies of events in a collection of reference alignments. When scoring an alignment by the logarithm of its emission and transition probabilities, the relative contribution to the alignment score of secondary structure emission probabilities versus amino acid emission probabilities is controlled by a hand-chosen weighting constant. ISPALign uses the hidden Markov model of ProbCons [7], where match states are modified to emit secondary structure types. The probability of emitting a pair of the same type is set to  $x$  for all three types, while the probability for a pair of differing types is  $1-x$ , where  $x$  is a single hand-chosen constant. The transition probabilities of ProbCons that correspond to gap open and extension penalties are used, shifted by factor  $y$  to compensate for the effect of the structure type emission probabilities on the substitution score, where  $y$  is a second hand-chosen constant. In increasing accuracy, these tools are ranked: PRALINE, SPEM, PROMALS, and ISPALign.

In contrast to the above approaches, we develop general scoring models with no ad hoc parameters that modify substitution scores on the basis of the pair of secondary structure states of the aligned residues, and that use a suite of affine gap penalties whose values depend on the degree of secondary structure in the region disrupted by the gap. A unique feature is that at each residue we also take into account the *confidence* of the prediction for the three structure types. (Confidences are output by tools such as PSIPRED [17].) While our models are more complex, especially in how gap costs are determined, we show that optimal pairwise alignments under the models can still be computed efficiently. And though these models have many parameters, we rigorously learn their values using inverse alignment [19][21][22] applied to training sets of reference alignments.

**Overview.** In Section 2, we present several new models for scoring alignments of protein sequences based on their predicted secondary structure. Section 3 develops efficient algorithms for computing optimal alignments of two sequences under these models. Section 4 reviews an approach called inverse alignment that uses these optimal alignment algorithms to learn parameter values for the models from examples of biological reference alignments. Section 5 then presents experimental results that compare the accuracy of these models when used for both pairwise and multiple alignment on standard benchmark reference alignments.

## 2 Scoring Alignments with Predicted Secondary Structure

We now introduce several models for scoring an alignment of two protein sequences  $A$  and  $B$  that make use of predicted secondary structures for  $A$  and  $B$ . These models score an alignment  $\mathcal{A}$  by specifying a cost function  $f(\mathcal{A})$ , where an optimal alignment minimizes  $f$ . The features of an alignment that  $f$  scores are: (i) substitutions of pairs of residues, (ii) internal gaps, and (iii) external gaps. A *gap* is a maximal run of either insertions or deletions. A gap is *external* if it inserts or deletes a prefix or a suffix of  $A$  or  $B$ ; otherwise it is *internal*.

Substitutions and gaps are scored in a position-dependent manner that takes into account the predicted secondary structure. A *substitution* in alignment  $\mathcal{A}$  of residues  $A[i]$  and  $B[j]$  is denoted by a tuple  $(A, i, B, j)$ . An *internal gap* is denoted by a tuple  $(S, i, j, T, k)$ , where substring  $S[i:j]$  is deleted from sequence  $S$  and inserted between positions  $k$  and  $k+1$  of sequence  $T$ . An *external gap* is denoted by a pair  $(i, j)$ , where prefix or suffix  $S[i:j]$  is deleted from  $A$  or  $B$ .

In general, the alignment cost function  $f(\mathcal{A})$  uses two other functions: function  $s$  for the cost of substitutions, and function  $g$  for the cost of internal gaps. External gaps use standard *affine* gap costs [12]. The general form of  $f$  is then

$$\begin{aligned} f(\mathcal{A}) := & \sum_{\substack{\text{substitutions} \\ (A, i, B, j) \in \mathcal{A}}} s(A, i, B, j) + \sum_{\substack{\text{internal gaps} \\ (S, i, j, T, k) \in \mathcal{A}}} g(S, i, j, T, k) + \\ & \sum_{\substack{\text{external gaps} \\ (i, j) \in \mathcal{A}}} \left( \tilde{\gamma} + (j-i+1) \tilde{\lambda} \right), \end{aligned} \quad (1)$$

where  $\tilde{\gamma}, \tilde{\lambda}$  are the respective gap *open* and *extension* penalties for external gaps. We next describe substitution cost function  $s$ , and internal gap cost function  $g$ .

## 2.1 Scoring Substitutions

Consider a substitution of two residues in an alignment, where these residues have amino acids  $a$  and  $b$ , and are involved in secondary structures of types  $c$  and  $d$ . For secondary structures, we consider the standard three types of alpha-helix, beta-strand, and loop, which we represent by the symbols  $\alpha$ ,  $\beta$ , and  $\phi$ , respectively. In the following,  $\Gamma = \{\alpha, \beta, \phi\}$  denotes the alphabet of secondary structure types, and  $\Sigma$  denotes the alphabet of amino acids.

Function  $s$  scores a substitution of amino acids  $a, b \in \Sigma$  with secondary structure types  $c, d \in \Gamma$  using two costs:  $\sigma(a, b)$ , the cost for *substituting* amino acids  $a$  and  $b$ , and  $\mu(c, d)$ , an additive *modifier* for the residues having secondary structure types  $c$  and  $d$ . Values  $\sigma_{ab} = \sigma(a, b)$  and  $\mu_{cd} = \mu(c, d)$  are parameters to our substitution model. In the model, both  $a, b$  and  $c, d$  are unordered pairs. This results in 210 substitution costs  $\sigma_{ab}$ , plus 6 secondary structure modifiers  $\mu_{cd}$ , for a total of 216 parameters that must be specified for the substitution model.

We consider two forms of prediction, which we call *lumped* or *distributed*.

**Lumped prediction.** In lumped prediction, which is a special case of distributed prediction, the prediction at each residue is a *single* secondary structure type. The predicted secondary structure for protein sequence  $A$  can then be represented by a string  $S_A$ , where residue  $i$  of  $A$  has predicted type  $S_A[i] \in \Gamma$ .

For lumped prediction, the substitution cost function  $s$  is

$$s(A, i, B, j) := \sigma(A[i], B[j]) + \mu(S_A[i], S_B[j]). \quad (2)$$

The modifier  $\mu(c, d)$  may be positive or negative. When the residues have the same secondary structure type,  $\mu(c, c) \leq 0$ , which makes it more favorable to align the residues. When the residues have different types,  $\mu(c, d) \geq 0$ , making it less favorable to align them. These constraints on the modifiers can be enforced during parameter learning, as described in Section 4.

**Distributed prediction.** The most accurate tools for secondary structure prediction, such as PSIPRED [17], output a *confidence* that the residue is in each possible type. For residue  $i$  of sequence  $A$ , we denote the predictor's confidence that the residue is in secondary structure type  $c$  by  $P_A(i, c) \geq 0$ . In practice, we normalize the confidences output by the predictor at each residue  $i$  to obtain a *distribution* with  $\sum_{c \in \Gamma} P_A(i, c) = 1$ .

For distributed prediction, the substitution cost function  $s$  is

$$s(A, i, B, j) := \sigma(A[i], B[j]) + \sum_{c, d \in \Gamma} P_A(i, c) P_B(j, d) \mu(c, d). \quad (3)$$

When the predictor puts all its confidence on one structure type at each residue, this reduces to the lumped prediction substitution function.

## 2.2 Scoring Gaps

With standard *affine* gap costs [12], the cost of inserting or deleting a substring of length  $k$  is  $\gamma + \lambda k$ , where  $\gamma$  and  $\lambda$  are respectively the gap open and extension costs. The new gap scoring models generalize this to a suite of gap open and extension costs whose values depend on the secondary structure around the gap. The basic idea is that the gap open cost  $\gamma$  depends on a *global* measure of how disruptive the entire gap is to the secondary structure of the proteins, while the gap extension cost  $\lambda$  charged per residue depends on a *local* measure of disruption at that residue's position. We define these notions more precisely below.

For an internal gap that deletes the substring  $S[i:j]$ , and inserts it after the residue  $T[k]$ , the gap cost function  $g$  has the general form,

$$g(S, i, j, T, k) := \gamma(H(S, i, j, T, k)) + \sum_{i \leq p \leq j} \lambda(h(S, p, T, k)).$$

The first term is a per-gap cost, and the second term is a sum of per-residue costs. Functions  $H$  and  $h$  are respectively the global and local measures of secondary structure disruption. Both  $H$  and  $h$  return integer values in the range  $L = \{1, 2, \dots, \ell\}$  that give the discrete *level* of disruption. The corresponding values for the gap open and extension costs,  $\gamma_i = \gamma(i)$  and  $\lambda_i = \lambda(i)$  for  $i \in L$ , are parameters to our model. For  $\ell$  levels, the internal gap cost model has  $2\ell$  parameters  $\gamma_i$  and  $\lambda_i$  that must be specified.

The gap costs at these levels satisfy  $0 \leq \gamma_1 \leq \dots \leq \gamma_\ell$  and  $0 \leq \lambda_1 \leq \dots \leq \lambda_\ell$ . In other words, a higher level of disruption incurs a greater gap cost. These constraints are enforced during parameter learning, as described in Section 4.

Functions  $H$  and  $h$ , which give the level of secondary structure disruption, depend on two aspects: (i) how strongly a position is involved in secondary structure, and (ii) which positions are considered when determining the level. We call the first aspect the *degree* of secondary structure, and the second aspect the *context* of the gap.

**Measuring the degree of secondary structure.** As described in Section 2.1, the predicted secondary structure for the residues of a protein sequence  $A$  may be represented by string  $S_A$  of structure types in the case of lumped prediction, or vector  $P_A$  of confidences for each type in the case of distributed prediction. We consider three ways of using such predictions to determine the degree of secondary structure at residue position  $i$  in sequence  $A$ . This *degree*  $\Psi_A(i)$  is in the range  $[0, 1]$ , where 0 corresponds to no involvement in secondary structure, and 1 corresponds to full involvement.

(1) The *lumped-binary* approach assumes a lumped prediction, and produces a binary value for the degree:  $\Psi_A(i)$  is 1 if  $S_A[i] \in \{\alpha, \beta\}$ , and 0 otherwise.

(2) The *distributed-binary* approach assumes a distributed prediction, and produces a binary degree:  $\Psi_A(i) = 1$  iff  $P_A(i, \alpha) + P_A(i, \beta) > P_A(i, \phi)$ .

(3) The *distributed-continuous* approach assumes a distributed prediction, and produces the real value  $\Psi_A(i) = P_A(i, \alpha) + P_A(i, \beta)$ .

**Specifying the gap context.** The gap context is specified by the positions that functions  $H$  and  $h$  consider when measuring the global and local secondary structure level. Both functions use the above secondary structure degree  $\Psi(i)$ .

To measure the local level  $h$  at position  $i$  in a sequence  $S$  of length  $n$ , we consider a small window  $W(i, n)$  of consecutive positions centered around  $i$ :

$$h(S, i) := \left\langle \sum_{p \in W(i, n)} \Psi_S(p) \middle/ |W(i, n)| \right\rangle.$$

Here the notation  $\langle x \rangle$  maps real value  $x \in [0, 1]$  to the discrete levels  $1, 2, \dots, \ell$  by  $\langle x \rangle := \lfloor (\ell-1)x \rfloor + 1$ . In words, the local level  $h$  at position  $i$  is the average secondary structure degree  $\Psi$  for the residues in a window around  $i$ . Generally all windows have the same width  $|W(i, n)| = w$ , except if  $i$  is too close to 1 or  $n$  to be centered in a window of width  $w$ , then  $W(i, n)$  shrinks on one side of  $i$ .

In our experiments in Section 5, we consider three ways of specifying the gap context, depending on whether we take an insertion view, a deletion view, or a mixed view of a gap. (The same context applies to all gaps in an alignment.)

(1) The *deletion context* views the disruption caused by the gap in terms of the secondary structure lost by deleting substring  $S[i:j]$ . For the global measure of disruption, this context takes the maximum local level of secondary structure over the positions in the deleted substring, which gives the gap cost function,

$$g(S, i, j, T, k) := \gamma \left( \max_{i \leq p \leq j} h(S, p) \right) + \sum_{i \leq p \leq j} \lambda(h(S, p)). \quad (4)$$

(2) The *insertion context* views the disruption in terms of the secondary structure displaced at residues  $T[k]$  and  $T[k+1]$  where the insertion occurs. For both the global and local measures of disruption this context uses

$$H(T, k) := \left\langle \frac{1}{2} h(T, k) + \frac{1}{2} h(T, k+1) \right\rangle,$$

which gives the gap cost function,

$$g(S, i, j, T, k) := \gamma(H(T, k)) + (j - i + 1) \lambda(H(T, k)). \quad (5)$$

(3) The *mixed context* combines the above global measure  $H$  of the insertion context with the local measure  $h$  of the deletion context, which gives

$$g(S, i, j, T, k) := \gamma(H(T, k)) + \sum_{i \leq p \leq j} \lambda(h(S, p)). \quad (6)$$

To summarize, the *parameters* of the scoring model for protein sequence alignment are the 210 substitution costs  $\sigma_{ab}$ , the 6 substitution modifiers  $\mu_{cd}$ , the  $2\ell$  gap costs  $\gamma_i$  and  $\lambda_i$  for internal gaps, and the two gap costs  $\tilde{\gamma}$  and  $\tilde{\lambda}$  for external gaps. This is a total of  $218 + 2\ell$  parameters. In general, the model depends on the window width  $w$ , the number of levels  $\ell$ , whether the secondary structure prediction is lumped or distributed, the choice of measure  $\Psi$  for the degree of secondary structure, and the choice of gap context.

### 3 Computing Optimal Alignments Efficiently

We can efficiently compute an optimal alignment of sequences  $A$  and  $B$  under scoring function  $f$  given by equation (2) using dynamic programming. Let  $C(i, j)$  be the cost of an optimal alignment of prefixes  $A[1:i]$  and  $B[1:j]$ . This alignment ends with either a *substitution* of residues  $A[i]$  and  $B[j]$ , or a *gap* involving substring  $A[k:i]$  or  $B[k:j]$  for some  $k$ . In each case, the alignment must be preceded by an optimal solution over shorter prefixes. This leads to the recurrence,

$$C(i, j) := \min \begin{cases} C(i-1, j-1) + s(A, i, B, j), \\ \min_{1 \leq k \leq i} \{C(k-1, j) + g(A, k, i, B, j)\}, \\ \min_{1 \leq k \leq j} \{C(i, k-1) + g(B, k, j, A, i)\}. \end{cases} \quad (7)$$

(To simplify the presentation, this ignores boundary conditions and the special case of external gaps.) For two sequences of length  $n$ , the straightforward algorithm that directly evaluates this recurrence in a table, and recovers an optimal alignment using the table, takes  $\Theta(n^3)$  time. (This assumes evaluating  $g$  takes  $O(1)$  time, which can be achieved through preprocessing, as discussed later.)

By studying gap cost function  $g$ , the time to compute an optimal alignment for all three gap contexts can be reduced to nearly  $O(n^2)$ , as discussed next.

**Insertion and mixed contexts.** For the *insertion context*, function  $g$  given by equation (5) is very close to a standard affine gap cost function. The same technique developed by Gotoh [22] for standard affine gap costs, namely (1) keeping track of three separate quantities at each entry  $(i, j)$  of the dynamic programming table, depending on whether the alignment ends by a substitution, insertion, or deletion, and (2) considering the last two columns of an alignment to decide whether or not the gap open cost should be charged, can be used to reduce the total time to compute an optimal alignment to  $O(n^2)$ . For the *mixed context*, given by equation (6), this same approach also leads to an  $O(n^2)$  time algorithm.

**Deletion context.** For the *deletion context*, function  $g$  given by equation (4) involves a maximization to determine the gap open cost, which complicates matters. For this context, the total time can be sped up significantly using the *candidate list technique* originally developed for alignment with convex gap costs by Miller and Myers [26] and Galil and Giancarlo [11]. While our gap cost function  $g$  is not convex in their original sense, their technique still applies. We briefly review its ideas below.

The candidate list technique speeds up the evaluation of the two inner minimizations in equation (7) for  $C(i, j)$ . The minimization involving  $g(B, k, j, A, i)$  can be viewed as computing the function

$$F_i(j) := \min_{1 \leq k \leq j} \{G_i(k, j)\},$$

where  $G_i(k, j) := C(i, k-1) + g(B, k, j, A, i)$ . Similarly, the minimization involving  $g(A, k, i, B, j)$  can be viewed as computing a function  $\tilde{F}_j(i)$  that is the

minimum of another function  $\tilde{G}_j(k, i)$ . At a high level, when filling in row  $i$  of the table for  $C(i, j)$ , the candidate list approach maintains a data structure for row  $i$  that enables fast computation of  $F_i(j)$  across the row for increasing  $j$ . Similarly, it maintains a separate data structure for each column  $j$  that enables fast computation of  $\tilde{F}_j(i)$  down the column. When processing entry  $(i, j)$  of the dynamic programming table, the data structures for row  $i$  and column  $j$  permit evaluation of  $F_i(j)$  and  $\tilde{F}_j(i)$  in  $O(\log n)$  amortized time. Evaluating the recurrence at the  $O(n^2)$  entries of the table then takes a total of  $O(n^2 \log n)$  time. A very readable exposition is given by Gusfield [15, pp. 293–302].

More specifically, for a fixed row  $i$  the candidate list technique computes  $F(j)$  as follows. (When  $i$  is fixed, we drop subscript  $i$  on  $F_i$  and  $G_i$ .) Each index  $k$  in the minimization of  $G(k, j)$  over  $1 \leq k \leq j$  is viewed as a *candidate* for determining the value of  $F(j)$ . Candidate  $k$  contributes the *curve*  $G(k, j)$ , which is viewed as a function of  $j$  for  $j \geq k$ . Geometrically, the set of values of  $F(j)$  for  $1 \leq j \leq n$ , which is the minimum of these curves, is known as their *lower envelope* [4].

When computing  $F(j)$  across the row for each successive  $j$ , a representation of this lower envelope is maintained at all  $j' \geq j$ , only considering curves for candidates  $k$  with  $k \leq j$ . This representation for  $j' \geq j$  is a partition of the interval  $[j, n]$  into maximal subintervals such that across each subinterval the minimum is given by exactly one curve. The process of adding a candidate's curve to the lower envelope exploits the following property of the gap cost function.

**Lemma 1 (Dominance property).** *Consider candidates  $a$  and  $b$  with  $a < b$ . Suppose  $G(a, c) < G(b, c)$  at some  $c \geq b$ . Then  $G(a, d) < G(b, d)$  at all  $d \geq c$ .*

*Proof.* The key is to show that the difference

$$(G(b, d) - G(b, c)) - (G(a, d) - G(a, c)) \quad (8)$$

is nonnegative, as adding  $G(b, c) - G(a, c)$  implies the lemma. For the deletion context, quantity (8) above equals

$$(\gamma(H(S, b, d)) - \gamma(H(S, b, c))) - (\gamma(H(S, a, d)) - \gamma(H(S, a, c))), \quad (9)$$

where function  $H(S, x, y)$  is the maximum of  $h(S, p)$  over the interval  $p \in [x, y]$ . Considering where  $h(S, p)$  attains its maximum on the intervals  $[b, d]$ ,  $[b, c]$ ,  $[a, d]$ ,  $[a, c]$ , and noting that  $\gamma((x))$  is nondecreasing in  $x$ , shows quantity (9) above is nonnegative, which proves the lemma.  $\square$

A key consequence of Lemma 1 is that the curves for any pair of candidates cross at most once. More precisely, for a pair  $a < b$  of candidates and an interval  $x$ , either one candidate dominates the other across  $x$ , or  $x$  can be split into two pieces  $y, z$  such that the later candidate  $b$  dominates on the left piece  $y$ , and the earlier candidate  $a$  dominates on the right piece  $z$ . Using this property, we update the lower envelope as follows.

Given a new candidate  $j$ , we compare it on intervals of the current partition  $p_1, \dots, p_t$ , starting with the leftmost. When comparing against the  $i$ th interval  $x = (p_{i-1}, p_i]$ , we first examine its right endpoint  $c = p_i$ . If  $G(j, c) \leq G(k_i, c)$ ,

then candidate  $j$  dominates across  $x$ , so we *delete* interval  $i$  from the partition (effectively merging it with the next interval), and continue comparing  $j$  against interval  $i+1$ . If  $G(j, c) > G(k_i, c)$ , then by Lemma 11,  $j$  is dominated on intervals  $i+1, \dots, t$  by their corresponding candidates, so those intervals do not change in the partition. Interval  $i$  may change, though if it does, it at worst splits into two pieces with  $j$  dominating in the left piece. In the case of a split, we *insert* a new leftmost interval  $[j, p]$  into the partition with  $j$  as the corresponding candidate. To find the split point  $p$  (if it exists), we can use binary search to identify the rightmost position such that  $G(j, p) < G(k, p)$ , where  $k$  is the candidate corresponding to the current interval. (The proof of Lemma 11 implies that the difference between the curves for candidates  $k$  and  $j$  is nonincreasing.) During the binary search, gap cost function  $g$  is evaluated  $O(\log n)$  times.

In general, updating the partition when considering a new candidate involves a series of deletes, followed by an insert. Assuming  $g$  can be evaluated in  $O(1)$  time, a delete takes  $O(1)$  time while an insert takes  $O(\log n)$  time. While a given update can involve several deletes, each delete removes an earlier candidate, which is never reinserted. Charging the delete to the removed candidate, the total time for deletes is then  $O(n^2)$ . The total time for all inserts is  $O(n^2 \log n)$ .

The final issue is the time to evaluate  $g(S, i, j, T, k)$  for the deletion context. With  $O(n)$  time preprocessing, we can evaluate the sum of gap extension penalties  $\lambda$  in the definition of  $g$  in  $O(1)$  time, by taking the difference of two precomputed prefix sums. With  $O(n^2)$  time and space preprocessing, we can look up the gap open penalty  $\gamma$  in  $O(1)$  time, by precomputing  $H(S, i, j)$  for all  $i, j$ . (Alternately, we can use a *range tree* [4] to find the maximum for  $H(S, i, j)$  online when evaluating the cost of a gap; this only requires  $O(n)$  time and space preprocessing, but it evaluates  $g$  in  $O(\log n)$  time, which is slightly slower.) In short, using  $O(n^2)$  time and space preprocessing, we can evaluate  $g$  in  $O(1)$  time.

We summarize the total time for the alignment algorithm below.

**Theorem 1 (Deletion context running time).** *For the deletion gap context, optimally aligning two sequences of lengths  $m \leq n$  takes  $O(n^2 + mn \log n)$  time.*

This provides a significant speedup in practice as well as in theory.

## 4 Learning Model Parameters by Inverse Alignment

We now show how to learn values for the parameters of the alignment scoring models using the *inverse alignment* approach developed in Kim and Kececioglu [21], modified to incorporate a loss function as introduced by Yu, Joachims, Elber and Pillardy [38]. The goal of parameter learning is to find values for which the optimal scoring alignment is the biologically correct alignment. Inverse alignment takes as input a collection of *examples* of correct reference alignments, and outputs an assignment of values for the parameters that makes the reference alignments score as close as possible to optimal scoring alignments of their sequences. The approach of Kim and Kececioglu [21] finds a parameter assignment that minimizes the average absolute error in score across the examples,

while Yu et al. [38] also incorporate the error in recovery between an optimal scoring alignment and the reference alignment. We briefly review these approaches below.

The reference alignments of protein sequences that are widely available for parameter learning, which are generally based on aligning the known three-dimensional structures of families of related proteins, are actually only partial alignments. In a *partial alignment*, columns are labeled as reliable or unreliable. Reliable columns typically correspond to core blocks, which are gapless regions of the alignment where common three-dimensional structure is shared across the family. At unreliable columns, the alignment is effectively left unspecified. In a *complete alignment*, all columns are reliable. We follow the presentation of Kim and Kececioglu [22], which first develops an algorithm for inverse alignment from examples that are complete alignments, and then extends it to partial examples.

**Complete examples.** In the following, we define inverse alignment under a new optimization criterion that we call *discrepancy*. The discrepancy criterion generalizes the score error approach of Kim and Kececioglu [21] and the recovery error approach of Yu et al. [38]. In our experiments, it is superior to both.

For the alignment scoring function  $f$ , let  $p_1, p_2, \dots, p_t$  be its parameters. We view the entire set of parameters as a vector  $p = (p_1, \dots, p_t)$  drawn from domain  $\mathcal{D}$ . When we want to emphasize the dependence of  $f$  on its parameters  $p$ , we write  $f_p$ . The input to inverse alignment consists of many example alignments  $\mathcal{A}_i$ , where each example aligns a corresponding pair of sequences  $\mathcal{S}_i$ . We denote the average length of the sequences in  $\mathcal{S}_i$  by  $\|\mathcal{S}_i\|$ .

For an alignment  $\mathcal{B}_i$  of sequences  $\mathcal{S}_i$  for example  $\mathcal{A}_i$ , let function  $d(\mathcal{A}_i, \mathcal{B}_i)$  be the fraction of reliable columns of example  $\mathcal{A}_i$  that are *not* present in alignment  $\mathcal{B}_i$ . In other words, function  $d$  measures the error in recovering example  $\mathcal{A}_i$  by alignment  $\mathcal{B}_i$ . Yu et al. [38] call  $d$  a loss function.

**Definition 1 (Inverse alignment with complete examples).** *Inverse Alignment* from complete examples under the discrepancy criterion is the following problem. The input is a collection of complete alignments  $\mathcal{A}_i$  of sequences  $\mathcal{S}_i$  for  $1 \leq i \leq k$ . The output is parameter vector  $x^* := \operatorname{argmin}_{x \in \mathcal{D}} D(x)$ , where

$$D(x) := \frac{1}{k} \sum_{1 \leq i \leq k} \max_{\mathcal{B}_i} \left\{ (1-\alpha) \frac{f_x(\mathcal{A}_i) - f_x(\mathcal{B}_i)}{\|\mathcal{S}_i\|} + \alpha d(\mathcal{A}_i, \mathcal{B}_i) \right\}. \quad (10)$$

In the above, the max is over all alignments  $\mathcal{B}_i$  of  $\mathcal{S}_i$ , and  $\alpha \in [0, 1]$  is a constant that controls the relative weight on score error versus recovery error. Function  $D(x)$  is the *average discrepancy* of the examples under parameters  $x$ .  $\square$

The discrepancy criterion reduces to the approach of Kim and Kececioglu [21] when  $\alpha = 0$ , and to the approach of Yu et al. [38] when  $\alpha = \frac{1}{2}$ , upon removing the length normalization of score error by setting  $\|\mathcal{S}_i\| \equiv 1$ . Intuitively, when minimizing discrepancy  $D(x)$ , the recovery error term  $d(\mathcal{A}_i, \mathcal{B}_i)$  drives the alignments  $\mathcal{B}_i$  that score better than example  $\mathcal{A}_i$  (which have a positive score error term under  $f_x$ ) toward also having low recovery error  $d$ . In other words, the

recovery error term helps make the best scoring alignments agree with the example on its columns. On the other hand, the scale of recovery error  $d \in [0, 1]$  is small, while the score error  $f_x(\mathcal{A}_i) - f_x(\mathcal{B}_i)$  can grow arbitrarily big, especially for long sequences. So correctly tuning the relative contribution of score error and recovery error is impossible for examples  $\mathcal{A}_i$  of varying lengths, unless score error is length normalized, which leads to the above discrepancy formulation.

For the alignment scoring functions  $f$  presented in Section 2, inverse alignment from complete examples can be reduced to linear programming. The parameters of scoring function  $f_x$  are the variables  $x$  of the linear program. The domain  $\mathcal{D}$  of the parameters is described by a set of inequalities that includes the bounds  $(0, -1, 0, 0) \leq (\sigma_{ab}, \mu_{cd}, \gamma_i, \lambda_i) \leq (1, 1, 1, 1)$ . The description of  $\mathcal{D}$  also contains the inequalities  $\sigma_{aa} \leq \sigma_{ab}$ ,  $\mu_{cc} \leq 0$ ,  $\mu_{cd} \geq 0$ ,  $\gamma_i \leq \gamma_{i+1}$ , and  $\lambda_i \leq \lambda_{i+1}$ .

The remaining inequalities measure discrepancy  $D(x)$ . Associated with each example  $\mathcal{A}_i$  is an error variable  $\delta_i$ . For example  $\mathcal{A}_i$ , and *every alignment*  $\mathcal{B}_i$  of sequences  $\mathcal{S}_i$ , the linear program has an inequality

$$(1-\alpha) \frac{f_x(\mathcal{A}_i) - f_x(\mathcal{B}_i)}{\|\mathcal{S}_i\|} + \alpha d(\mathcal{A}_i, \mathcal{B}_i) \leq \delta_i. \quad (11)$$

Note this is a linear inequality in the variables  $x$ , since function  $f_x$  is linear in  $x$ .

The objective function for the linear program is to minimize  $\frac{1}{k} \sum_{1 \leq i \leq k} \delta_i$ . Minimizing this objective forces each  $\delta_i$  to equal the term with index  $i$  in the summation for  $D(x)$  in equation (10). Thus, an optimal solution  $x^*$  to the linear program minimizes the average discrepancy of the examples.

This linear program has an exponential number of inequalities of form (11), since for an example  $\mathcal{A}_i$ , the number of alignments  $\mathcal{B}_i$  of  $\mathcal{S}_i$  is exponential in the lengths of the sequences [13]. Nevertheless, this program can be solved in polynomial time by a far-reaching result from linear programming theory known as the equivalence of separation and optimization [14]. This equivalence result is that a linear program can be solved in polynomial time iff the *separation problem* for the linear program can be solved in polynomial time. The separation problem is, given a possibly infeasible vector  $\tilde{x}$  of values for the variables, to report an inequality from the linear program that is violated by  $\tilde{x}$ , or to report that  $\tilde{x}$  satisfies the linear program if there is no violated inequality.

We can solve the separation problem in polynomial time for the above linear program, given a concrete parameter choice  $\tilde{x}$ , by the following. We consider each example  $\mathcal{A}_i$ , and find an alignment  $\mathcal{B}_i^*$  that maximizes the left-hand side of inequality (11), where for scoring function  $f$  we use parameters  $\tilde{x}$ . If for this alignment  $\mathcal{B}_i^*$  the left-hand side is at most  $\delta_i$ , then all inequalities of form (11) involving  $\mathcal{A}_i$  are satisfied by  $\tilde{x}$ ; if the left-hand side exceeds  $\delta_i$ , then this  $\mathcal{B}_i^*$  gives the requisite violated inequality. Finding this extreme alignment  $\mathcal{B}_i^*$  can be reduced to computing an optimal scoring alignment where scoring function  $f$  is modified slightly to take into account the contribution of substitutions that coincide with the columns of  $\mathcal{A}_i$  to the recovery error  $d(\mathcal{A}_i, \mathcal{B}_i)$ . (More details are provided in [38] and [22].) For an instance of inverse alignment with  $k$  examples, solving the separation problem for the linear program involves computing at most  $k$  optimal scoring alignments, which can be done in polynomial time.

In practice, we solve the linear program using a *cutting plane algorithm* [5]. This approach starts with a small subset  $\mathcal{P}$  of all the inequalities  $\mathcal{L}$ . An optimal solution  $\tilde{x}$  of the linear program restricted to  $\mathcal{P}$  is found, and the separation algorithm for the full linear program  $\mathcal{L}$  is called on  $\tilde{x}$ . If the separation algorithm reports that  $\tilde{x}$  satisfies  $\mathcal{L}$ , then  $\tilde{x}$  is an optimal solution to  $\mathcal{L}$ . Otherwise, the violated inequality that is returned is added to  $\mathcal{P}$ , and the process is repeated. (See [22] for more details.) While such cutting plane algorithms are not guaranteed to terminate in polynomial time, they can be fast in practice. We start with  $\mathcal{P}$  containing just the trivial inequalities that specify parameter domain  $\mathcal{D}$ .

**Partial examples.** As mentioned earlier, the examples that are currently available are partial alignments. Given such a partial example  $\mathcal{A}$  for sequences  $\mathcal{S}$ , a *completion*  $\overline{\mathcal{A}}$  of  $\mathcal{A}$  is a complete alignment of  $\mathcal{S}$  that agrees with the reliable columns of  $\mathcal{A}$ . In other words, a completion  $\overline{\mathcal{A}}$  can change  $\mathcal{A}$  on the substrings that are in unreliable columns, but must not alter  $\mathcal{A}$  in reliable columns.

When learning parameters by inverse alignment from partial examples, we treat the unreliable columns as missing information.

**Definition 2 (Inverse alignment with partial examples).** *Inverse Alignment* from partial examples is the following problem. The input is a collection of partial alignments  $\mathcal{A}_i$  for  $1 \leq i \leq k$ . The output is parameter vector

$$x^* := \operatorname{argmin}_{x \in \mathcal{D}} \min_{\overline{\mathcal{A}}_1, \dots, \overline{\mathcal{A}}_k} D(x),$$

where the min is over all possible completions  $\overline{\mathcal{A}}_i$  of the partial examples.  $\square$

Kim and Kececioglu [21] present the following iterative approach to partial examples. Start with an initial completion  $(\overline{\mathcal{A}}_i)_0$  for each partial example  $\mathcal{A}_i$ , which may be formed by computing alignments of the unreliable regions that are optimal with respect to a default parameter choice  $(x)_0$ . Then iterate the following for  $j = 0, 1, 2, \dots$ . Compute an optimal parameter choice  $(x)_{j+1}$  by solving inverse alignment on the complete examples  $(\overline{\mathcal{A}}_i)_j$ . Given  $(x)_{j+1}$ , form a new completion  $(\overline{\mathcal{A}}_i)_{j+1}$  of each example  $\mathcal{A}_i$  by computing an alignment of each unreliable region that is optimal with respect to parameters  $(x)_{j+1}$ , and concatenating these alignments of the unreliable regions alternating with the alignments given by the reliable regions. Such a completion optimally stitches together the reliable regions, using the current parameter estimate.

The discrepancy for successive parameter estimates forms a monotonically decreasing sequence, and hence converges [21]. In practice, we iterate until the improvement is too small, or a limit on the number of iterations is reached [22].

## 5 Experimental Results

To evaluate these scoring models, we studied how well an optimal alignment under the model recovers a biological benchmark alignment, when using the models for both *pairwise alignment* and *multiple alignment*.

**Pairwise alignment.** For the experiments on pairwise alignment, we collected examples from the following suites of reference alignments: BALiBASE [2], HOMSTRAD [27], PALI [3], and SABMARK [35]. From each suite, we selected a subset of 100 pairwise alignments as examples. The corresponding subsets of BALiBASE are denoted by  $B$ ; similarly,  $H$  for HOMSTRAD,  $P$  for PALI, and  $S$  for SABMARK. We also define the union  $U = B \cup H \cup P \cup S$ , and for a set  $X \in \{B, H, P, S\}$  we denote the complement with respect to  $U$  by  $\overline{X} = U - X$ . Gap level  $\ell = 8$ , window width  $w = 7$ , and discrepancy weight  $\alpha = 0.1$  are used throughout. To predict secondary structure, we used PSIPRED [17]; to solve linear programs, we used GLPK [25]. Recovery is measured in terms of the *SPS score* [2]: the percentage of residue pairs in the core blocks of all induced pairwise alignments of the reference alignment that are correctly aligned in the computed alignment.

The first set of experiments, shown in Table 1, study the effect of the substitution models alone (without the new gap models). We compared the recovery rates of the lumped and distributed prediction models for secondary structure modifiers; gap costs use the standard affine model of a gap open and extension penalty for internal gaps, and a separate gap open and extension penalty for terminal gaps. The new substitution models are also compared with the standard model that does not use modifiers, called “none.” These experiments use hold-out cross validation, where parameters learned on training set  $\overline{B}$  are applied to test set  $B$ , and so on for each suite of examples. As the table shows, the highest recovery is achieved using the *distributed prediction* model.

Table 2 studies the effect of the gap models alone (without substitution modifiers). The table compares each of the three models of gap context, combined with each of the three measures of the degree of secondary structure. Due to page limits, only results where the test and training sets are both  $U$  are shown. As the table shows, the highest recovery is achieved using the *mixed context* combined with the *distributed-continuous degree* measure of secondary structure.

**Table 1.** Recovery rates comparing only the new substitution models

Training set	Test set	Modifier model		
		none	lumped	distributed
$\overline{B}$	$B$	73.52	80.99	81.57
$\overline{H}$	$H$	79.92	82.44	83.58
$\overline{P}$	$P$	67.87	74.67	76.60
$\overline{S}$	$S$	68.76	72.58	72.55

**Table 2.** Recovery rates comparing only the new gap models

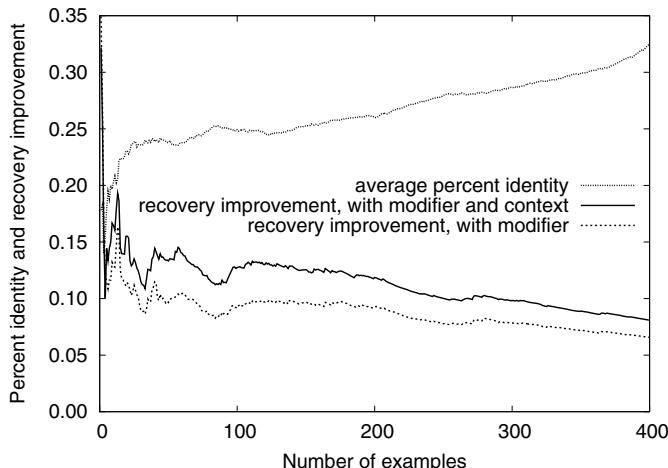
Secondary structure degree model	Gap context model		
	deletion	insertion	mixed
lumped-binary	75.63	75.59	76.51
distributed-binary	75.86	75.90	76.43
distributed-continuous	75.96	76.22	76.63

Table 3 shows the improvement in recovery over the standard model (that does not use the new substitution or gap models), upon first adding the best new substitution model (distributed prediction), and then adding the best new gap model (distributed-continuous degree with mixed context). As can be seen, adding the new substitution model gives a large improvement in recovery; adding the new gap model gives a further, but smaller, improvement.

**Table 3.** Recovery rates comparing both the substitution and gap models

Training set	Test set	Substitution and gap model			
		std sub	new sub	new sub	
		std gap	std gap	new gap	
$\overline{B}$	$B$	73.51	81.57	82.88	
$\overline{H}$	$H$	79.92	83.58	84.62	
$\overline{P}$	$P$	67.87	76.60	77.32	
$\overline{S}$	$S$	68.76	72.55	75.52	

Figure 1 plots the improvement in recovery for pairwise alignment for examples that are ranked according to the dissimilarity of their sequences. We sorted the examples from set  $U$  in order of decreasing normalized alignment cost [36], where their cost is scored according to the standard alignment model using the default parameter values of the multiple alignment tool Opal [37]. Each value of  $n$  along the horizontal axis corresponds to using as a test set the first  $n$  examples in this ranking. The uppermost curve plots the average percent identity for these corresponding test sets, which generally increases as the average normalized alignment cost decreases along the horizontal axis. The lower two curves



**Fig. 1.** Improvement in recovery rate in relation to percent identity

plot the improvement in recovery as measured on these test sets for parameter values learned on training set  $U$ . Improvement is measured with respect to the standard model with no substitution modifiers and no gap context. The lowest curve adds only substitution modifiers to the standard model, and the next higher curve adds both substitution modifiers and gap context. Note that the greatest improvement is generally achieved for examples with the lowest percent identity. For examples with less than 25% identity, the combined improvement in recovery using both substitution modifiers and gap context is as much as 15%.

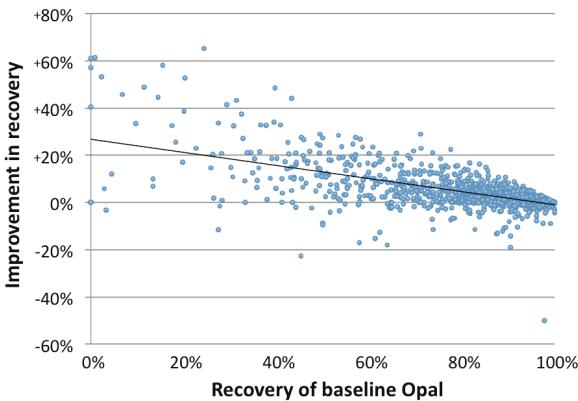
To give an idea of the running time, solving an inverse alignment instance with 300 examples took at most 55 minutes on a Pentium IV running at 3 GHz with 1 Gb of RAM, and involved around 18,600 cutting planes.

**Multiple alignment.** To evaluate these models when used for multiple sequence alignment, we incorporated them into the alignment tool **Opal** [3]. **Opal** is unique in that it constructs multiple sequence alignments using an algorithm for optimally aligning two multiple alignments under the sum-of-pairs scoring function [20,36]. Since its subalignments have optimal score in this sense, it is a good testbed for comparing scoring models, as effects due to suboptimality with respect to the scoring function are reduced. The baseline version of **Opal** uses the BLOSUM62 substitution matrix [16] with affine gap penalties (specifically, a gap open and extension penalty for internal gaps, and another open and extension penalty for terminal gaps). We modified **Opal** to incorporate predicted secondary structure, still using the BLOSUM62 substitution matrix, but now combined with parameters learned using our best scoring model (namely, the distributed prediction model for substitution modifiers, and the distributed-continuous measure with mixed context for gap costs). In these experiments, we rank benchmarks according their *hardness*: their average recovery rate under MAFFT, ProbCons, and baseline **Opal** (which all have comparable recovery rates [3]).

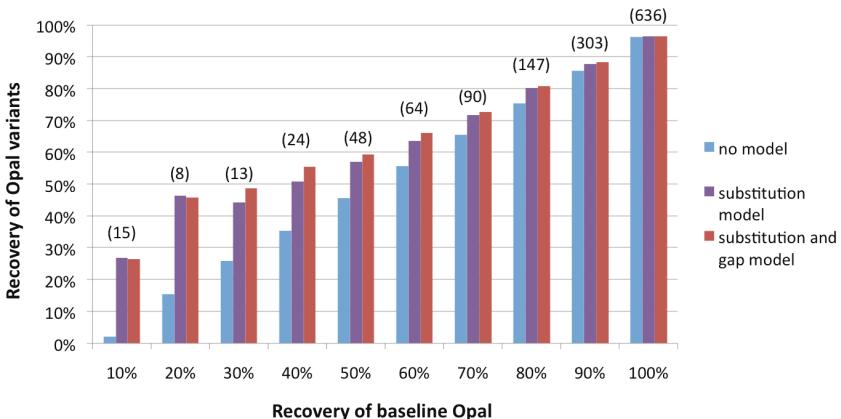
Figure 2 shows a scatter plot of the improvement in recovery of **Opal** when using secondary structure, over all benchmarks in the BALiBASE [2], PALI [3], and HOMSTRAD [27] suites. Note that for the more difficult benchmarks (with lower baseline recovery), the boost in recovery using secondary structure tends to be greater. Notice also that using the secondary structure scoring model occasionally worsens the recovery. (This is to be expected, as structure prediction is imperfect, and no universal parameter choice will improve all benchmarks).

Figure 3 shows the recovery rates on these same benchmarks for three variants of **Opal**: (1) the baseline version with no secondary structure model, (2) using the new substitution model with secondary structure but the baseline gap model, and (3) using the new substitution model and the new gap model with secondary structure. Benchmarks are binned according to their baseline recovery in **Opal**; at the top of each bar in parentheses is the number of benchmarks in its bin. Note that the substitution model provides the largest boost in recovery, which is generally greatest for the hardest examples.

Finally, Figure 4 compares the improvement in recovery achieved using secondary structure in two multiple alignment tools: ISPAlign [23] and **Opal**. (ISPAlign has the best recovery among the competing tools PRALINE, SPEM,

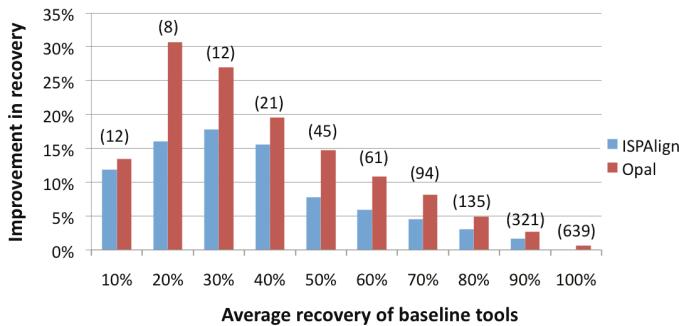


**Fig. 2.** Improvement in recovery using secondary structure within **Opal**



**Fig. 3.** Average recovery of model variants in **Opal** versus binned baseline recovery

and PROMALS that use secondary structure.) ISPAlign was modified to not add sequence profiles or intermediate sequences (which are not used by **Opal**), so the effect of its secondary structure scoring model could be isolated. The recovery boost in ISPAlign is measured with respect to ProbCons, since without secondary structure ISPAlign uses ProbCons to compute alignments. (ProbCons and baseline **Opal** have comparable accuracy [36].) The same benchmarks are used, binned by hardness. The boost in recovery for **Opal** is much greater than for ISPAlign, suggesting that the more involved secondary structure scoring model used in **Opal** may be more effective than the simpler model used in ISPAlign.



**Fig. 4.** Improvement in recovery using secondary structure within alignment tools

## 6 Conclusion

We have presented new models for protein sequence alignment that incorporate predicted secondary structure, and have shown through experimental results on reference alignments that when model parameters are learned using inverse alignment, the models significantly boost the accuracy of both pairwise and multiple alignment of distant protein sequences. Incorporating secondary structure into the substitution scoring function provides the largest benefit, with distributed prediction giving the most accurate substitution scores, while the new gap penalty functions provide a lesser yet still substantial benefit. Comparing with other multiple alignment tools that incorporate secondary structure shows our models provide a larger increase in accuracy compared to not using secondary structure, which suggests that the additional complexity of our models is offset by their correspondingly greater increase in accuracy.

There remain many avenues for further research. In particular, given that improved substitution scores provided the largest boost in accuracy, it would be interesting to learn a model with a substitution scoring function  $\sigma(a, b, c, d)$  that directly scores each pairing  $\{(a, c), (b, d)\}$  of amino acids  $a, b$  with secondary structure types  $c, d$  respectively. Such a model has 1,830 substitution parameters  $\sigma_{abcd}$  alone, and will require a very large training set, combined with a careful procedure for fitting default values for unobserved parameters.

**Acknowledgements.** We thank Matt Cordes and Chuong Do for helpful discussions, and the reviewers for their suggestions. This research was supported by the US National Science Foundation through Grant DBI-0317498.

## References

1. Aydin, Z., Altunbasak, Y., Borodovsky, M.: Protein secondary structure prediction for a single-sequence using hidden semi-Markov models. *BMC Bioinformatics* 7(178), 1–15 (2006)

2. Bahr, A., Thompson, J.D., Thierry, J.C., Poch, O.: BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research* 29(1), 323–326 (2001)
3. Balaji, S., Sujatha, S., Kumar, S.S.C., Srinivasan, N.: PALI: a database of alignments and phylogeny of homologous protein structures. *Nucleic Acids Research* 29(1), 61–65 (2001)
4. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer, Berlin (2000)
5. Cook, W., Cunningham, W., Pulleyblank, W., Schrijver, A.: Combinatorial Optimization. John Wiley and Sons, New York (1998)
6. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In: Dayhoff, M.O. (ed.) *Atlas of Protein Sequence and Structure*, vol. 5(3), pp. 345–352. National Biomedical Research Foundation, Washington DC (1978)
7. Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S.: ProbCons: probabilistic consistency based multiple sequence alignment. *Genome Research* 15, 330–340 (2005)
8. Do, C.B., Gross, S., Batzoglou, S.: CONTRAlign: discriminative training for protein sequence alignment. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2006. LNCS (LNBI), vol. 3909, pp. 160–174. Springer, Heidelberg (2006)
9. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge (1998)
10. Edgar, R.C.: MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* 32, 1792–1797 (2004)
11. Galil, Z., Giancarlo, R.: Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science* 64, 107–118 (1989)
12. Gotoh, O.: An improved algorithm for matching biological sequences. *Journal of Molecular Biology* 162, 705–708 (1982)
13. Griggs, J.R., Hanlon, P., Odlyzko, A.M., Waterman, M.S.: On the number of alignments of  $k$  sequences. *Graphs and Combinatorics* 6, 133–146 (1990)
14. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)
15. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York (1997)
16. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA* 89, 10915–10919 (1992)
17. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology* 292, 195–202 (1999)
18. Katoh, K., Kuma, K.I., Toh, H., Miyata, T.: MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research* 33, 511–518 (2005)
19. Kececioglu, J., Kim, E.: Simple and fast inverse alignment. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2006. LNCS (LNBI), vol. 3909, pp. 441–455. Springer, Heidelberg (2006)
20. Kececioglu, J., Starrett, D.: Aligning alignments exactly. In: Proceedings of the 8th ACM Conference on Research in Computational Molecular Biology, pp. 85–96 (2004)
21. Kim, E., Kececioglu, J.: Inverse sequence alignment from partial examples. In: Giancarlo, R., Hannenhalli, S. (eds.) WABI 2007. LNCS (LNBI), vol. 4645, pp. 359–370. Springer, Heidelberg (2007)

22. Kim, E., Kececioglu, J.: Learning scoring schemes for sequence alignment from partial examples. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 5(4), 546–556 (2008)
23. Lu, Y., Sze, S.-H.: Multiple sequence alignment based on profile alignment of intermediate sequences. *Journal of Computational Biology* 15(7), 676–777 (2008)
24. Lüthy, R., McLachlan, A.D., Eisenberg, D.: Secondary structure-based profiles: use of structure-conserving scoring tables in searching protein sequence databases for structural similarities. *Proteins: Structure, Function, and Genetics* 10, 229–239 (1991)
25. Makhorin, A.: GNU Linear Programming Kit, release 4.8 (2005),  
<http://www.gnu.org/software/glpk>
26. Miller, W., Myers, E.W.: Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology* 50, 97–120 (1988)
27. Mizuguchi, K., Deane, C.M., Blundell, T.L., Overington, J.P.: HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Science* 7, 2469–2471 (1998)
28. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology* 302, 205–217 (2000)
29. Pei, J., Grishin, N.V.: MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Research* 34, 4364–4374 (2006)
30. Pei, J., Grishin, N.V.: PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics* 23(7), 802–808 (2007)
31. Sander, C., Schneider, R.: Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Genetics* 9, 56–68 (1991)
32. Simossis, V.A., Heringa, J.: PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. *Nucleic Acids Research* 33, W289–W294 (2005)
33. Söding, J.: Protein homology detection by HMM-HMM comparison. *Bioinformatics* 21(7), 951–960 (2005)
34. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research* 22, 4673–4680 (1994)
35. Van Walle, I., Lasters, I., Wyns, L.: Align-m: A new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics* 20, 1428–1435 (2004)
36. Wheeler, T.J., Kececioglu, J.D.: Multiple alignment by aligning alignments. In: *Proceedings of the 15th ISCB Conference on Intelligent Systems for Molecular Biology (ISMB)*, *Bioinformatics*, vol. 23, pp. i559–i568 (2007)
37. Wheeler, T.J., Kececioglu, J.D.: Opal: software for aligning multiple biological sequences. Version 0.3.7 (2007), <http://opal.cs.arizona.edu>
38. Yu, C.-N., Joachims, T., Elber, R., Pillardy, J.: Support vector training of protein alignment models. *Journal of Computational Biology* 15(7), 867–880 (2008)
39. Zhou, H., Zhou, Y.: SPEM: improving multiple sequence alignment with sequence profiles and predicted secondary structures. *Bioinformatics* 21, 3615–3621 (2005)

# Author Index

- Agarwal, Pratul K. 138  
Alipour, Masoud 184  
Alkan, Can 218  
  
Backofen, Rolf 285, 339  
Bafna, Vineet 220, 356  
Bailey-Kellogg, Chris 321  
Bandeira, Nuno 356  
Bar-Joseph, Ziv 90  
Bashir, Ali 220  
Beerenwinkel, Niko 271  
Berger, Bonnie 339  
Bilmes, Jeff A. 434  
Bininda-Emonds, Olaf R.P. 184  
Blanchette, Mathieu 302  
Borgwardt, Karsten M. 201  
Briggs, Steve 356  
Bruckner, Sharon 74  
  
Carson, Dennis 220  
Chauve, Cedric 46  
Clermont, Gilles 155  
Coulombe, Benoit 302  
Cowen, Lenore J. 372  
  
Daskalakis, Constantinos 451  
DeBartolo, Joe 59  
Denby, Katherine 201  
Devadas, Srinivas 339  
Donzé, Alexandre 155  
Dost, Banu 356  
  
Eichler, Evan E. 218  
El-Mabrouk, Nadia 46  
Eskin, Eleazar 466, 482  
  
Freed, Karl F. 59  
  
Geyrhofer, Lukas 271  
Ghahramani, Zoubin 201  
Griswold, Karl E. 321  
Gusfield, Dan 236  
  
Halperin, Eran 108  
Han, Buhm 482  
  
He, Zengyou 16  
Hescott, Benjamin J. 372  
Hormozdiari, Fereydoun 218  
Hüffner, Falk 74  
  
Jaschek, Rami 170  
  
Kang, Eun Yong 466  
Kang, Hyun Min 482  
Karp, Richard M. 74, 108  
Kececioglu, John 512  
Kim, Eagu 512  
Kingsford, Carl 400  
Kirkpatrick, Bonnie 108  
Kurnikova, Maria 138  
  
Langmead, Christopher J. 138, 155  
Lavallée-Adam, Mathieu 302  
Legay, Axel 155  
Leiserson, Maek D.M. 372  
Levo, Michal 217  
Li, Xiangqian 356  
Lin, Yu 386  
Liu, Yu-Tseng 220  
Lu, Qing 220  
Lu, Yong 90  
  
Mäkinen, Veli 121  
Markowitz, Victor M. 496  
Möhl, Mathias 285  
Moret, Bernard M.E. 184, 386  
Mossel, Elchanan 451  
  
Nagarajan, Niranjan 400  
Nau, Gerard J. 90  
Navarro, Gonzalo 121  
Navlakha, Saket 400  
Noble, William Stafford 434  
  
O'Donnell, Charles W. 339  
  
Pan, Feng 253  
Pati, Amrita 496  
Pattengale, Nicholas D. 184  
Pe'er, Itsik 270  
Peng, Jian 31, 59  
Pinter, Ron Y. 496

- Pop, Mihai 400  
Prabhu, Snehit 270
- Qi, Robert Z. 16
- Rajan, Vaibhav 386  
Ramanathan, Arvind 138  
Raphael, Benjamin J. 220, 418  
Raveh-Sadka, Tali 217  
Reynolds, Sheila M. 434  
Roch, Sébastien 451  
Rosa, Javier 108  
Rosenfeld, Roni 90  
Roth, Volker 271
- Sahinalp, S. Cenk 218  
Segal, Eran 217  
Shamir, Ron 74  
Sharan, Roded 74  
Sharon, Itai 496  
Shen, Zhouxin 356  
Shibuya, Tetsuo 1  
Shpitser, Ilya 466  
Sindi, Suzanne S. 418  
Sirén, Jouni 121  
Slonim, Donna K. 372  
Sosnick, Tobin R. 59  
Stamatakis, Alexandros 184
- Stegle, Oliver 201  
Swenson, Krister M. 386
- Tam, Jason Po-Ming 16  
Tanay, Amos 170
- Välimäki, Niko 121
- Waldspühl, Jérôme 339  
Wang, Wei 253  
Wheeler, Travis 512  
White, James 400  
Wild, David L. 201  
Will, Sebastian 285, 339
- Xie, Yuying 253  
Xu, Jinbo 31, 59
- Yang, Can 16  
Yang, Chao 16  
Ye, Chun 466  
Yu, Weichuan 16
- Zagordi, Osvaldo 271  
Zaitlen, Noah A. 482  
Zhang, Xiang 253  
Zhao, Feng 59  
Zheng, Wei 321  
Zou, Fei 253