

S. Cenk Sahinalp (Ed.)

LNBI 10229

Research in Computational Molecular Biology

21st Annual International Conference, RECOMB 2017
Hong Kong, China, May 3–7, 2017
Proceedings

 Springer

Subseries of Lecture Notes in Computer Science

LNBI 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

LNBI Editorial Board

Søren Brunak

Technical University of Denmark, Kongens Lyngby, Denmark

Mikhail S. Gelfand

IITP, Research and Training Center on Bioinformatics, Moscow, Russia

Thomas Lengauer

Max Planck Institute for Informatics, Saarbrücken, Germany

Satoru Miyano

University of Tokyo, Tokyo, Japan

Eugene Myers

Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany

Marie-France Sagot

Université Lyon 1, Villeurbanne, France

David Sankoff

University of Ottawa, Ottawa, Canada

Ron Shamir

Tel Aviv University, Ramat Aviv, Tel Aviv, Israel

Terry Speed

Walter and Eliza Hall Institute of Medical Research, Melbourne, VIC, Australia

Martin Vingron

Max Planck Institute for Molecular Genetics, Berlin, Germany

W. Eric Wong

University of Texas at Dallas, Richardson, TX, USA

More information about this series at <http://www.springer.com/series/5381>

S. Cenk Sahinalp (Ed.)

Research in Computational Molecular Biology

21st Annual International Conference, RECOMB 2017
Hong Kong, China, May 3–7, 2017
Proceedings

Editor
S. Cenk Sahinalp
Indiana University Bloomington
Bloomington, IN
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Bioinformatics
ISBN 978-3-319-56969-7 ISBN 978-3-319-56970-3 (eBook)
DOI 10.1007/978-3-319-56970-3

Library of Congress Control Number: 2017936939

LNCS Sublibrary: SL8 – Bioinformatics

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

RECOMB, the Annual International Conference on Research in Computational Molecular Biology started in 1997, under the leadership of Sorin Istrail, Pavel Pevzner, and Michael Waterman. This year marks its 21st anniversary. RECOMB 2017 was hosted by The University of Hong Kong and The Chinese University of Hong Kong during May 3–7, 2017. This volume contains the 38 extended or short abstracts selected for oral presentation at RECOMB 2017 by the Program Committee (PC). Each of the 184 submissions consisted of a full paper, and was assigned to at least three PC members and reviewed with the help of many external reviewers. Following the initial reviews, final decisions were made after an extensive discussion of the submissions among the members of the PC.

Even though RECOMB 2017 did not allow parallel submissions, authors of accepted papers were given the option to publish short abstracts in the proceedings and submit their full papers to a journal. The papers for which the proceedings feature short abstracts had appeared in a journal by the time of the conference and were to be deposited in the preprint server arxiv.org. All other papers that appear as long abstracts in the proceedings were invited for submission to RECOMB 2017 special issues by either *Cell Systems* or the *Journal of Computational Biology*.

In addition to the paper presentations, RECOMB 2017 featured six invited keynote talks by leading scientists worldwide. The keynote speakers were Colin Collins (Vancouver Prostate Centre), Joe Gray (Oregon Health Sciences University), Wang Jun (iCarbonX), Laxmi Parida (IBM T.J. Watson Research Center), Ben Raphael (Princeton University), and Michael Schnell Levin (10X Genomics).

Following the tradition started at RECOMB 2010, RECOMB 2017 also featured highlight talks presenting computational biology papers that were published in journals during the last 18 months. There were 32 highlight submissions, six of which were selected for oral presentation at the main conference.

The success of RECOMB depends on the effort, dedication, and devotion of many colleagues. I especially thank the Organizing Committee chair, Siu Ming Yiu (The University of Hong Kong), and Kevin Yip (The Chinese University of Hong Kong), for hosting RECOMB and RECOMB-Seq 2017, the RECOMB satellite meeting on Massively Parallel Sequencing; Sumaiya Nazeen (MIT) for website design and technical support; the Steering Committee and especially its chair, Bonnie Berger (MIT), for help, advice, and support throughout the process; Mona Singh, the Program Chair of RECOMB 2016 (Princeton), for answering my many questions; Mathieu Blanchette (McGill) for chairing the highlights track; Paul Medvedev (Penn State) for acting as the publications chair; Alex Schoenhuth (CWI) for acting as the publicity chair; Fereydoun Hormozdiari (UC Davis) and Jian Ma (CMU) for chairing RECOMB-Seq; the main conference and RECOMB-Seq PC members and external reviewers for their timely reviews of assigned papers despite their busy schedules; the authors of the papers, highlights, and posters for their scientific contributions; and all the attendees for their

enthusiastic participation in the conference. We also thank the International Society of Computational Biology (ISCB) for student support and the Croucher Foundation for additional sponsorship.

February 2017

S. Cenk Sahinalp

Organization

Program Committee

Max Alekseyev	George Washington University, USA
Rolf Backofen	Albert Ludwigs University of Freiburg, Germany
Vineet Bafna	University of California, San Diego, USA
Chris Bailey-Kellogg	Dartmouth College, USA
Nuno Bandeira	UCSD, USA
Ziv Bar-Joseph	Carnegie Mellon University, USA
Niko Beerenwinkel	ETH Zurich, Switzerland
Bonnie Berger	Massachusetts Institute of Technology, USA
Mathieu Blanchette	McGill University, USA
Sebastian Böcker	Friedrich Schiller University of Jena, Germany
Lenore Cowen	Tufts University, USA
Nadia El-Mabrouk	University of Montreal, Canada
Irit Gat-Viks	Tel Aviv University, Israel
David Gifford	MIT, USA
Raluca Gordan	Duke University, USA
Fereydoun Hormozdiari	University of California, Davis, USA
Trey Ideker	University of California, San Diego, USA
Tao Jiang	University of California, Riverside, USA
Vladimir Jojic	University of North Carolina, USA
John Kececioglu	University of Arizona, USA
Manolis Kellis	MIT, USA
Carl Kingsford	Carnegie Mellon University, USA
Gunnar W. Klau	CWI, The Netherlands
Jens Lagergren	SBC and CSC, KTH
Max Leiserson	Tufts University, USA
Ming Li	University of Waterloo, Canada
Jian Ma	Carnegie Mellon University, USA
Paul Medvedev	Pennsylvania State University, USA
Bernard Moret	EPFL, Switzerland
Veli Mäkinen	University of Helsinki, Finland
William Stafford Noble	University of Washington, USA
Laxmi Parida	IBM T.J. Watson Research Center, USA
Bogdan Pasaniuc	UCLA, USA
Jian Peng	University of Illinois at Urbana-Champaign, USA
Yann Ponty	CNRS/LIX, Polytechnique
Teresa Przytycka	NIH, USA
Ben Raphael	Princeton University, USA
Knut Reinert	FU Berlin, Germany

S. Cenk Sahinalp	Indiana University, Bloomington, USA
Alexander Schoenhuth	Centrum Wiskunde and Informatica, The Netherlands
Russell Schwartz	Carnegie Mellon University, USA
Roded Sharan	Tel Aviv University, Israel
Mona Singh	Princeton University, USA
Donna Slonim	Tufts University, USA
Sagi Snir	Institute of Evolution
Leen Stougie	VU University
Jens Stoye	Bielefeld University, Germany
Fengzhu Sun	University of Southern California, USA
Wing-Kin Sung	Nuational University of Singapore, Singapore
Glenn Tesler	University of California, San Diego, USA
Tamir Tuller	Tel Aviv University, Israel
Alfonso Valencia	Spanish National Cancer Research Centre, Spain
Fabio Vandin	University of Padova, Italy
Martin Vingron	Max Planck Institut für molekulare Genetik, Germany
Jerome Waldispuhl	McGill University, Canada
Sebastian Will	University of Leipzig, Germany
Jinbo Xu	Toyota Technological Institute at Chicago, USA
Noah Zaitlen	University of California San Francisco, USA
Alex Zelikovsky	Georgia State University, USA
Jianyang Zeng	Tsinghua University, China
Louxin Zhang	National University of Singapore, Singapore
Xuegong Zhang	Tsinghua University, China

Additional Reviewers

Aganezov, Sergey	Biran, Hadas	Cunial, Fabio
Aguiar, Derek	Boix, Carles	Dahl, Andy
Alachiotis, Nikolaos	Bonora, Giancarlo	Daniels, Noah
Alexeev, Nikita	Bordewich, Magnus	Dao, Phuong
Alkhnabashi, Omer	Bryant, David	Deblasio, Dan
Andonov, Rumen	Canzar, Stefan	Ding, Jun
Artyomenko, Alexander	Cardner, Mathias	Dirmeier, Simon
Arvestad, Lars	Castillo, Omar	Doerr, Daniel
Avdeyev, Pavel	Chaisson, Mark	Doty, David
Backofen, Rolf	Cho, Hoon	Eetemadi, Ameen
Bankevich, Anton	Cho, Hyunghoon	El-Kebir, Mohammed
Bansal, Vikas	Chor, Benny	Emde, Anne-Katrin
Batu, Tugkan	Cichonska, Anna	Engler, Martin
Berry, Vincent	Cook, Kate	Eslami Rasekh, Marzieh
Bhadra, Sahely	Costa, Fabrizio	Fallmann, Joerg
Bhutani, Kunal	Crawford, Jake	Feijao, Pedro
Bielow, Chris	Csuros, Miklos	Frenkel, Zeev

Frishberg, Amit	Käll, Lukas	Palmer, Cameron
G. Costa, Ivan	Köster, Johannes	Park, Danny
Gagie, Travis	Lafond, Manuel	Park, Yongjin
Gao, Tianxiang	Langmead, Christopher	Peng, Yu
Glynn, Eric	Lee, Heewook	Persi, Erez
Golumbeanu, Monica	Lei, Jinzhi	Persikov, Anton
Gottlieb, Assaf	Lemaitre, Claire	Pham, Son
Gruenewald, Stefan	Lemieux, Sebastien	Pirkli, Martin
Gu, Jin	Levy, Maya	Pisanti, Nadia
Guo, Yuchun	Li, Wenyuan	Pittala, Srivamshi
Hach, Faraz	Li, Yue	Pockrandt, Christopher
Hajirasouliha, Iman	Libbrecht, Max	Pons Mayol, Joan Carles
Halldorsson, Bjarni	Limasset, Antoine	Przytycki, Pawel
Harel, Tom	Lin, Dejun	Pullman, Benjamin
Heller, David	Lin, Yen Yi	Rashid, Sabrina
Herman, Pawel	Liu, Yan	Reinharz, Vladimir
Hescott, Ben	Liu, Yaping	Ren, Jie
Hescott, Benjamin	Loh, Po-Ru	Reyna, Matthew
Hiaminen, Niina	Lu, Yang	Rhrissorakrai, Kahn
Hobolth, Asger	Ludwig, Marcus	Ruffalo, Matthew
Hodzic, Ermin	Luhmann, Nina	Rusch, Doug
Holley, Guillaume	Löytynoja, Ari	Saglam, Mert
Homilius, Max	Ma, Cong	Sahlin, Kristoffer
Hormozdiari, Farhad	Ma, Jianzhu	Salmela, Leena
Howbert, Jeff	Malikic, Salem	Sanguinetti, Guido
Hua, Kui	Mandric, Igor	Sarkar, Abhishek
Huang, Justin	Mann, Martin	Sauerwald, Natalie
Huska, Matt	Marcais, Guillaume	Schreiber, Jacob
Huynh-Thu, Vân Anh	May, Damon	Schöpflin, Robert
Jahn, Katharina	Mazza, Arnon	Scott, Camille
Jain, Siddhartha	Mefford, Joel	Shao, Mingfu
Joseph, Tyler	Meuleman, Wouter	Shi, Alvin
Jünemann, Sebastian	Minkin, Ilia	Shrestha, Raunak
Kamm, John	Mirzaei, Sajad	Shteyman, Alan
Kehr, Birte	Munro, Daniel	Silberberg, Yael
Keich, Uri	Na, Seungjin	Silverbush, Dana
Kelk, Steven	Nadimpalli, Shilpa	Simmons, Sean
Kim, Juho	Navlakha, Saket	Sindi, Suzanne
Kim, Yoo-Ah	Noutahi, Emmanuel	Singer, Jochen
Knyazev, Sergey	Numanagic, Ibrahim	Siragusa, Enrico
Kockan, Can	Nurk, Sergey	Solomon, Brad
Kopp, Wolfgang	Oesper, Layla	Stolzer, Maureen
Koyama, Taka	Ohler, Uwe	Stoye, Jens
Kuipers, Jack	Oren, Yael	Sun, Chen
Kundu, Kousik	Orenstein, Yaron	Sundermann, Linda
Kundu, Ritu	Ouangraoua, Aida	Swenson, Krister

Syed, Tahin	Wall, Timothy	Xin, Hongyi
Tang, Haixu	Wang, Lusheng	Yang, Shuo
Tang, Kujin	Wang, Mingxun	Yanover, Chen
Thankachan, Sharma V.	Wang, Sheng	Yardimci, Galip
Thurnherr, Thomas	Wang, Tina	Ye, Yuzhen
Tremblay-Savard, Olivier	Wang, Weili	Yeo, Grace
Utro, Filippo	Wang, Yijie	Yilmaz, Sule
Uurtio, Viivi	Wang, Ying	You, Xintian
Valenzuela, Daniel	Wetzel, Joshua	Yu, Michael
van Iersel, Leo	White, Tim	Yu, Ning
Varoquaux, Nelle	Wienbrandt, Lars	Zamalloa, Jose
Verma, Deeptak	Wilentzik, Roni	Zekic, Tina
Viduani Martinez, Fábio Henrique	Wise, Aaron	Zeng, Haoyang
von Kleist, Max	Wittler, Roland	Zhang, Mengge
	Wójtowicz, Damian	Zhang, Zhizhuo

Contents

Boosting Alignment Accuracy by Adaptive Local Realignment.	1
<i>Dan DeBlasio and John Kececioglu</i>	
A Concurrent Subtractive Assembly Approach for Identification of Disease Associated Sub-metagenomes	18
<i>Wontack Han, Mingjie Wang, and Yuzhen Ye</i>	
A Flow Procedure for the Linearization of Genome Sequence Graphs	34
<i>David Haussler, Maciej Smuga-Otto, Benedict Paten, Adam M. Novak, Sergei Nikitin, Maria Zueva, and Dmitrii Miagkov</i>	
Dynamic Alignment-Free and Reference-Free Read Compression	50
<i>Guillaume Holley, Roland Wittler, Jens Stoye, and Faraz Hach</i>	
A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases	66
<i>Chirag Jain, Alexander Dilthey, Sergey Koren, Srinivas Aluru, and Adam M. Phillippy</i>	
Determining the Consistency of Resolved Triplets and Fan Triplets.	82
<i>Jesper Jansson, Andrzej Lingas, Ramesh Rajaby, and Wing-Kin Sung</i>	
Progressive Calibration and Averaging for Tandem Mass Spectrometry Statistical Confidence Estimation: Why Settle for a Single Decoy?	99
<i>Uri Keich and William Stafford Noble</i>	
Resolving Multicopy Duplications <i>de novo</i> Using Polyploid Phasing.	117
<i>Mark J. Chaisson, Sudipto Mukherjee, Sreeram Kannan, and Evan E. Eichler</i>	
A Bayesian Active Learning Experimental Design for Inferring Signaling Networks.	134
<i>Robert Osazuwa Ness, Karen Sachs, Parag Mallick, and Olga Vitek</i>	
<i>BBK*</i> (Branch and Bound over K^*): A Provable and Efficient Ensemble-Based Algorithm to Optimize Stability and Binding Affinity over Large Sequence Spaces.	157
<i>Adegoke A. Ojewole, Jonathan D. Jou, Vance G. Fowler, and Bruce R. Donald</i>	
Superbubbles, Ultrabubbles and Cacti	173
<i>Benedict Paten, Adam M. Novak, Erik Garrison, and Glenn Hickey</i>	

EPR-Dictionaries: A Practical and Fast Data Structure for Constant Time Searches in Unidirectional and Bidirectional FM Indices 190
Christopher Pockrandt, Marcel Ehrhardt, and Knut Reinert

A Bayesian Framework for Estimating Cell Type Composition from DNA Methylation Without the Need for Methylation Reference 207
Elior Rahmani, Regev Schweiger, Liat Shenhav, Eleazar Eskin, and Eran Halperin

Towards Recovering Allele-Specific Cancer Genome Graphs 224
Ashok Rajaraman and Jian Ma

Using Stochastic Approximation Techniques to Efficiently Construct Confidence Intervals for Heritability 241
Regev Schweiger, Eyal Fisher, Elior Rahmani, Liat Shenhav, Saharon Rosset, and Eran Halperin

Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees 257
Brad Solomon and Carl Kingsford

AllSome Sequence Bloom Trees 272
Chen Sun, Robert S. Harris, Rayan Chikhi, and Paul Medvedev

Longitudinal Genotype-Phenotype Association Study via Temporal Structure Auto-learning Predictive Model 287
Xiaoqian Wang, Jingwen Yan, Xiaohui Yao, Sungeun Kim, Kwangsik Nho, Shannon L. Risacher, Andrew J. Saykin, Li Shen, Heng Huang, and for the ADNI

Improving Imputation Accuracy by Inferring Causal Variants in Genetic Studies 303
Yue Wu, Farhad Hormozdiari, Jong Wha J. Joo, and Eleazar Eskin

The Copy-Number Tree Mixture Deconvolution Problem and Applications to Multi-sample Bulk Sequencing Tumor Data 318
Simone Zaccaria, Mohammed El-Kebir, Gunnar W. Klau, and Benjamin J. Raphael

Quantifying the Impact of Non-coding Variants on Transcription Factor-DNA Binding 336
Jingkang Zhao, Dongshunyi Li, Jungkyun Seo, Andrew S. Allen, and Raluca Gordân

aBayesQR: A Bayesian Method for Reconstruction of Viral Populations Characterized by Low Diversity 353
Soyeon Ahn and Haris Vikalo

BeWith: A Between-Within Method for Module Discovery in Cancer using Integrated Analysis of Mutual Exclusivity, Co-occurrence and Functional Interactions (Extended Abstract) 370
Phuong Dao, Yoo-Ah Kim, Sanna Madan, Roded Sharan, and Teresa M. Przytycka

K-mer Set Memory (KSM) Motif Representation Enables Accurate Prediction of the Impact of Regulatory Variants 372
Yuchun Guo, Kevin Tian, Haoyang Zeng, and David K. Gifford

Network-Based Coverage of Mutational Profiles Reveals Cancer Genes. 375
Borislav H. Hristov and Mona Singh

Ultra-Accurate Complex Disorder Prediction: Case Study of Neurodevelopmental Disorders 377
Linh Huynh and Fereydoun Hormozdiari

Inference of the Human Polyadenylation Code 379
Michael K.K. Leung, Andrew Delong, and Brendan J. Frey

Folding Membrane Proteins by Deep Transfer Learning. 380
Zhen Li, Sheng Wang, Yizhou Yu, and Jinbo Xu

A Network Integration Approach for Drug-Target Interaction Prediction and Computational Drug Repositioning from Heterogeneous Information 383
Yunan Luo, Xinbin Zhao, Jingtian Zhou, Jinling Yang, Yanqing Zhang, Wenhua Kuang, Jian Peng, Ligong Chen, and Jianyang Zeng

Epistasis in Genomic and Survival Data of Cancer Patients 385
Dariusz Matlak and Ewa Szczurek

Ultra-Fast Identity by Descent Detection in Biobank-Scale Cohorts Using Positional Burrows-Wheeler Transform. 387
Ardalan Naseri, Xiaoming Liu, Shaojie Zhang, and Degui Zhi

Joker de Bruijn: Sequence Libraries to Cover All k -mers Using Joker Characters 389
Yaron Orenstein, Ryan Kim, Polly Fordyce, and Bonnie Berger

GATTACA: Lightweight Metagenomic Binning Using Kmer Counting 391
Victoria Popic, Volodymyr Kuleshov, Michael Snyder, and Serafim Batzoglou

Species Tree Estimation Using ASTRAL: How Many Genes Are Enough? 393
Shubhanshu Shekhar, Sebastien Roch, and Siavash Mirarab

Reconstructing Antibody Repertoires from Error-Prone Immunosequencing Datasets	396
<i>Alexander Shlemov, Sergey Bankevich, Andrey Bzikadze, Yana Safonova, and Pavel A. Pevzner</i>	
NetREX: Network Rewiring Using EXpression - Towards Context Specific Regulatory Networks	398
<i>Yijie Wang, Dong-Yeon Cho, Hangnoh Lee, Brian Oliver, and Teresa M. Przytycka</i>	
E Pluribus Unum: United States of Single Cells	400
<i>Joshua D. Welch, Alexander Hartemink, and Jan F. Prins</i>	
ROSE: A Deep Learning Based Framework for Predicting Ribosome Stalling	402
<i>Sai Zhang, Hailin Hu, Jingtian Zhou, Xuan He, Tao Jiang, and Jianyang Zeng</i>	
Author Index	405

Boosting Alignment Accuracy by Adaptive Local Realignment

Dan DeBlasio¹(✉) and John Kececioglu²

¹ Computational Biology Department, Carnegie Mellon University, Pittsburgh, USA
deblasio@cmu.edu

² Department of Computer Science, The University of Arizona, Tucson, USA
kece@cs.arizona.edu

Abstract. While mutation rates can vary markedly over the residues of a protein, multiple sequence alignment tools typically use the same values for their scoring-function parameters across a protein’s entire length. We present a new approach, called *adaptive local realignment*, that in contrast automatically adapts to the diversity of mutation rates along protein sequences. This builds upon a recent technique known as parameter advising that finds global parameter settings for aligners, to adaptively find local settings. Our approach in essence identifies local regions with low estimated accuracy, constructs a set of candidate realignments using a carefully-chosen collection of parameter settings, and replaces the region if a realignment has higher estimated accuracy. This new method of *local parameter advising*, when combined with prior methods for global advising, boosts alignment accuracy as much as 26% over the best default setting on hard-to-align protein benchmarks, and by 6.4% over global advising alone. Adaptive local realignment, implemented within the *Opal* aligner using the *Facet* accuracy estimator, is available at facet.cs.arizona.edu.

Keywords: Multiple sequence alignment · Iterative refinement · Local mutation rates · Alignment accuracy · Parameter advising

1 Introduction

Ever since the 1960s, it has been known that proteins can have distinct mutation rates at different locations along the molecule [11]. The amino acids at some positions in a protein may stay unmutated for long periods of time, while other regions change a great deal (sometimes called “hypermutable regions”). This has led to methods in phylogeny construction that take variable mutation rates into account when building trees from sequences [26]. In multiple sequence alignment, however, variation in mutation rates across sequences to our knowledge has yet to be successfully exploited to improve alignment accuracy. Multiple sequence alignments are typically computed using a single setting of values for the parameters of the alignment scoring function. This single parameter setting affects

The work of both authors was performed at the University of Arizona.



Fig. 1. *Effect of adaptive local realignment.* Two alignments of the same region of benchmark BB11007 from the BALiBASE suite, where the amino acids highlighted in red uppercase are from the so-called core columns of the reference alignment, which should be aligned in a correct alignment. (a) The alignment computed by Opa1 using its optimal default parameter setting (VTML200, 45, 11, 42, 40) across the sequences, with an accuracy of 89.6%. The regions of the alignment in gray boxes are automatically selected for realignment. (b) The outcome of adaptive local realignment, with an improved accuracy of 99.6%, that uses different parameters settings in each region. The realignments of the three regions use alternate parameter settings (BLOSUM62, 45, 2, 45, 42), (BLOSUM62, 95, 38, 40, 40), and (VTML200, 45, 18, 45, 45), respectively.

how residues across a protein are aligned, and implicitly assumes uniform mutation rates. In contrast, the approach of this paper identifies alignment regions that may be misaligned under a single parameter setting, and finds alternate settings that may more closely match the local mutation rate of the sequences.

We present a method that takes a given alignment and attempts to improve its overall accuracy by replacing sections of it with better subalignments, as demonstrated in Fig. 1. The top alignment of the figure was computed using a single parameter setting; the optimal default setting of the Opa1 aligner [25]. The bottom alignment is obtained by our new method, taking the top alignment, automatically identifying the sections in gray boxes, and realigning them using alternate parameter settings, as described later in Sect. 3. This increases the overall alignment accuracy by 10%, as most of the misaligned core blocks (highlighted in red uppercase) are now corrected.

Related Work. Methods that partition a set of sequences to align or realign them can be grouped into two categories, based on the orientation of their partitions. *Vertical* realigners cut the input sequences into substrings, and once these shorter substrings are realigned, they stitch their alignments together. *Horizontal* realigners split an alignment into groups of whole sequences, which are then merged together by realigning between groups, possibly using the induced sub-alignment of each group. Realignment is occasionally called alignment *polishing*.

Crumble and Prune [21] is a pair of algorithms for performing both vertical (Crumble) and horizontal (Prune) splits on an input set of sequences. The objective, however, for splitting sequences both vertically and horizontally within

Crumble and **Prune** is not to improve accuracy, but to reduce running time and memory consumption, making aligning a large number of long sequences feasible.

Gotoh [12] presented several horizontal methods for heuristically aligning two multiple sequence alignments, which he called “group-to-group” alignment. This can be used for alignment construction in a progressive aligner, proceeding bottom-up over the guide tree and applying group-to-group alignment at each node, or for polishing an existing alignment by assigning sequences to two groups and using it to realign the groups.

AlignAlign [16] is unique as a horizontal method in that it implements an exact algorithm for optimally aligning two multiple sequence alignments under the sum-of-pairs scoring function with affine gap costs. This optimal group-to-group alignment algorithm, used for both alignment construction and alignment polishing, forms the basis of the **Opal** aligner [25].

The standard aligners **MUSCLE** [10], **MAFFT** [14], and **ProbCons** [8] also include a polishing step that performs horizontal realignment similar to Gotoh.

While realignment attempts to *correct* errors in existing alignments that were made during the alignment process, several tools attempt to *avoid* making these errors in the first place by adjusting parameter values along the sequences during alignment construction. For example, **PRANK** [17] uses a multi-level HMM that effectively chooses the alignment scoring function at each position. **T-Coffee** [19] uses consistency between pairwise alignments to create position-specific substitution scores. In fact, even the early tool **ClustalW** [23] adjusted positional gap-penalties based on pairwise sequence characteristics. Nevertheless, these tools which adjust positional alignment scores all attain lower accuracies on protein benchmarks than the **Opal** aligner without positional adjustment that we compare against in our experiments.

Adaptive local realignment, in contrast, is a vertical approach that aims to improve alignment accuracy, and a meta-method that can be applied to any aligner with tunable parameters. To our knowledge, this is the first realignment approach that automatically adapts to varying mutation rates along a protein and successfully achieves a demonstrable improvement in accuracy.

2 Background on Parameter Advising

To make the paper self-contained, we briefly review our prior work on parameter advising. We first review the concept of a parameter advisor, which requires an estimator of alignment accuracy and a set of parameter choices for the advisor, and then summarize our prior techniques for learning both an estimator and an advisor set. (An extensive discussion of parameter advising for multiple sequence alignment is in [7].)

We emphasize that while this section describes how to find an accuracy estimator and advisor set based on training examples, in practice a user of parameter advising will simply apply an advisor with a precomputed accuracy estimator and advisor set, and will not invoke the training procedures described here.

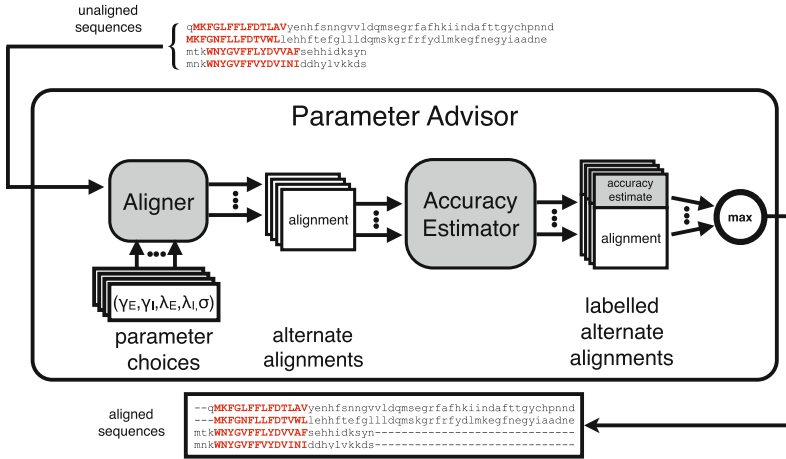


Fig. 2. *The parameter advising process.* For an input set of sequences, a parameter advisor first invokes the aligner for each assignment of parameter values in a collection of parameter choices. Each parameter choice when used with the aligner produces an alternate alignment of the sequences. An accuracy estimator is then used to label each of the alternate alignments with an accuracy estimate. The advisor then returns the alignment with the highest accuracy estimate.

2.1 Global Parameter Advising

The goal of parameter advising is to find the parameter setting for an aligner that yields the most accurate alignment of a given set of input sequences. The accuracy of a computed alignment is measured with respect to the “correct” alignment of the sequences (which often is not known). For special benchmark sets of protein sequences, the gold-standard alignment of the proteins, called their *reference alignment*, is usually obtained through structural alignment by finding the best superposition of the known three-dimensional structures of the proteins. Columns of the reference alignment that contain a residue from every protein in the set (where a *residue* is the amino acid at a particular position in a protein), and for which the residues in the column are all mutually close in space in the superposition of the structures, are called *core columns*. Runs of consecutive core columns are called *core blocks*, which represent the regions of the structural alignment with the highest confidence of being correct. Given such a reference alignment with identified core blocks, the *accuracy* of a different, computed alignment is the fraction of the pairs of residues aligned in the core blocks of the reference alignment that are also aligned in the computed alignment. (So a computed alignment of 100% accuracy completely agrees with the reference on its core blocks, though it may disagree elsewhere.) The best computed alignment is one of highest accuracy, and the task of a parameter advisor is to find a setting of the tunable parameters of an aligner that yields an accurate output alignment.

This setting can be highly input dependent, as the best choice of parameter values for an aligner can vary for different sets of input sequences.

When aligning sequences in practice, a reference alignment is almost never known, in which case the true accuracy of a computed alignment cannot be measured. Instead our parameter advisor relies on an accuracy *estimator* E that for an alignment A , gives a value $E(A)$ in the range $[0, 1]$ that estimates the true accuracy of alignment A . An estimator should be efficiently computable and positively correlated with true accuracy.

To choose a parameter setting, an advisor takes a set of choices P , where each *parameter choice* $p \in P$ is a vector that assigns values to all the tunable parameters of an aligner, and picks the choice that yields a computed alignment of highest estimated accuracy.

Formally, given an accuracy estimator E and a set P of parameter choices, a *parameter advisor* tries each parameter choice $p \in P$, invokes an aligner to compute an alignment A_p using choice p , and then selects the parameter choice p^* that has maximum estimated accuracy: $p^* \in \operatorname{argmax}_{p \in P} \{E(A_p)\}$. Figure 2 shows a diagram of parameter advising. Since the advisor runs the aligner $|P|$ times on a given set of input sequences, a crucial aspect of parameter advising is finding a small set P for which the true accuracy of the output alignment A_{p^*} is high.

To construct a good advisor, we need to find a good estimator E and a good set P . The estimator and advisor set are learned on training data consisting of benchmark sets of protein sequences for which a reference alignment is known. The learning procedure tries to find an estimator E and set P that maximize the true accuracy of the resulting advisor on this training data, which we subsequently assess on separate testing data.

Note that the process of advising is fast: for a set P of k parameter choices, advising involves computing k alignments under these choices, which can be done in parallel, evaluating the estimator on these k alignments, and taking a max. The separate process of training an advisor, by learning an estimator and advisor set as we review next, is done once, off-line, before any advising is done.

2.2 Learning an Accuracy Estimator

Our previous work [6, 15] presented an efficient approach for learning an accuracy estimator that is a linear combination of real-valued alignment feature functions, based on solving a large-scale linear programming problem. This approach resulted in **Facet** (short for “feature-based accuracy estimator” [4]), which is currently the most accurate estimator for parameter advising [5, 15].

This approach assumes we have a collection of d real-valued feature functions $g_1(A), \dots, g_d(A)$ on alignments A , where these functions g_i are positively correlated with true accuracy. The alignment accuracy estimator E is a linear combination of these functions, $E(A) = \sum_{1 \leq i \leq d} c_i g_i(A)$, where the coefficients c_i specify the estimator E . When the feature functions have range $[0, 1]$ and the coefficients form a convex combination, the resulting estimator E will also have range $[0, 1]$. **Facet** uses a collection of five feature functions, many of which

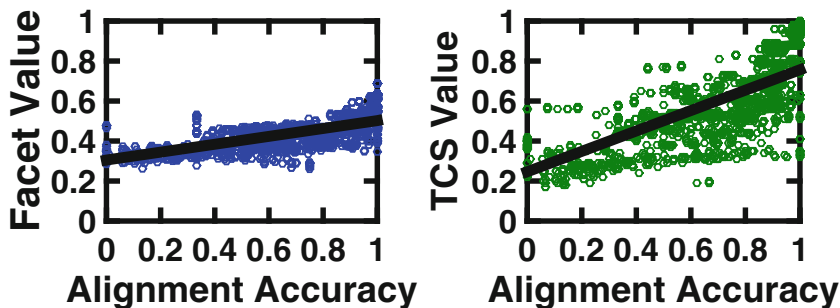


Fig. 3. *Relationship of estimators to true accuracy.* Each point in a scatterplot corresponds to an alignment whose true accuracy is on the horizontal axis, and whose value under a given estimator is on the vertical axis. Both scatterplots show the same set of 3,000 alignments under the accuracy estimators `Facet` [15] and `TCS` [3].

make use of predicted secondary structure for the protein sequences [15]. Figure 3 shows the correlation of `Facet` and `TCS` [3] (the next best estimator in our tests) to true accuracy. To be able to distinguish good from bad alignments an estimator should have a steep slope and very little spread. While the `TCS` estimator has high slope, it has quite a bit of spread. In contrast, the `Facet` estimator has much less spread but a less steep slope, and we have found this to be more effective in ranking alignments for parameter advising.

The features we use in `Facet` are a mixture of canonical measures of alignment quality, such as Amino Acid Identity, and novel non-local features of an alignment that correlate with true accuracy. Many of the most accurate features use predicted protein secondary structure. For instance, the Secondary Structure Blockiness feature finds an optimal packing of blocks of aligned amino acids that have the same predicted structure type. The other feature functions used in the `Facet` estimator are: Secondary Structure Identity, Secondary Structure Agreement, Gap Open Density, and Core Column Percentage. A full description of all features is in [15].

A parameter advisor uses the estimator to effectively rank alignments, so an estimator just needs to be monotonic in true accuracy. The *difference-fitting* approach learns the coefficients of an estimator that is close to monotonic by fitting the estimator to differences in true accuracy for pairs of training alignments. We can formulate the problem of coefficient finding using difference-fitting as a linear program; the details of this approach are in [15].

2.3 Learning an Advisor Set

The size of the parameter set used for advising should be small, since the aligner is run for each parameter setting. We utilize the concept of an oracle [25] (a perfect advisor that has access to the true accuracy of an alignment) to find sets that we use in practice. For a given advisor set P , an *oracle* selects parameter

choice $\operatorname{argmax}_{p \in P} \{F(A_p)\}$, where again function F gives the true accuracy of an alignment. (Equivalently, an oracle is an advisor that uses the perfect estimator F .) An oracle always picks the parameter choice that yields the highest accuracy alignment.

While an oracle is impossible to construct in practice, it gives a theoretical limit on the accuracy achievable by advising with a given set. Furthermore, if we find the optimal advisor set for an oracle for a given cardinality bound k , which we call an *oracle set*, then the performance of an oracle on an oracle set gives a theoretical limit on how well advising can perform for a given bound k on the number of parameter choices. In practice, oracle sets are used with **Facet** to construct an advisor.

We have shown that while finding an optimal oracle set is NP-complete, it can be formulated as an integer linear programming problem [15]. Learning an optimal oracle set of cardinality k , for a universe of u parameter choices and a training set of t benchmarks, involves solving an integer linear program with $\Theta(ut)$ variables and $\Theta(ut)$ inequalities. Using the CPLEX integer linear programming solver, this formulation permits finding optimal oracle sets in practice even for cardinalities up to $k = 25$.

It is possible to use a greedy procedure to find advisor sets tuned to a concrete estimator rather than the oracle [5]. While using these sets on global parameter advising increased advising accuracy over oracle sets, this increase did not transfer to adaptive local realignment. For the results in later sections, we will construct an advisor using the **Facet** accuracy estimator learned using difference fitting, along with oracle sets. Note this is *not* an oracle advisor, since it uses the **Facet** estimator.

This prior work focused on using parameter advising to choose the parameter setting for an entire alignment, which we call here *global* parameter advising. The next section presents adaptive local realignment, which leverages these ideas to in essence achieve *local* parameter advising.

3 Adaptive Local Realignment

To overcome the issue of protein sequences being non-homogeneous and having regions that may require different alignment parameter settings we have developed a method we call *adaptive local realignment*. Adaptive local realignment uses some of the same basic principles that have been shown to work well for global parameter advising. We apply the techniques described in the previous section locally to choose the best alignment parameters over an interval of columns in an alignment.

The adaptive local realignment method can be broken down into two steps: (1) discerning regions of the alignment that are well-aligned, which should be retained; and (2) producing a new alignment for regions that are poorly aligned, using parameter advising.

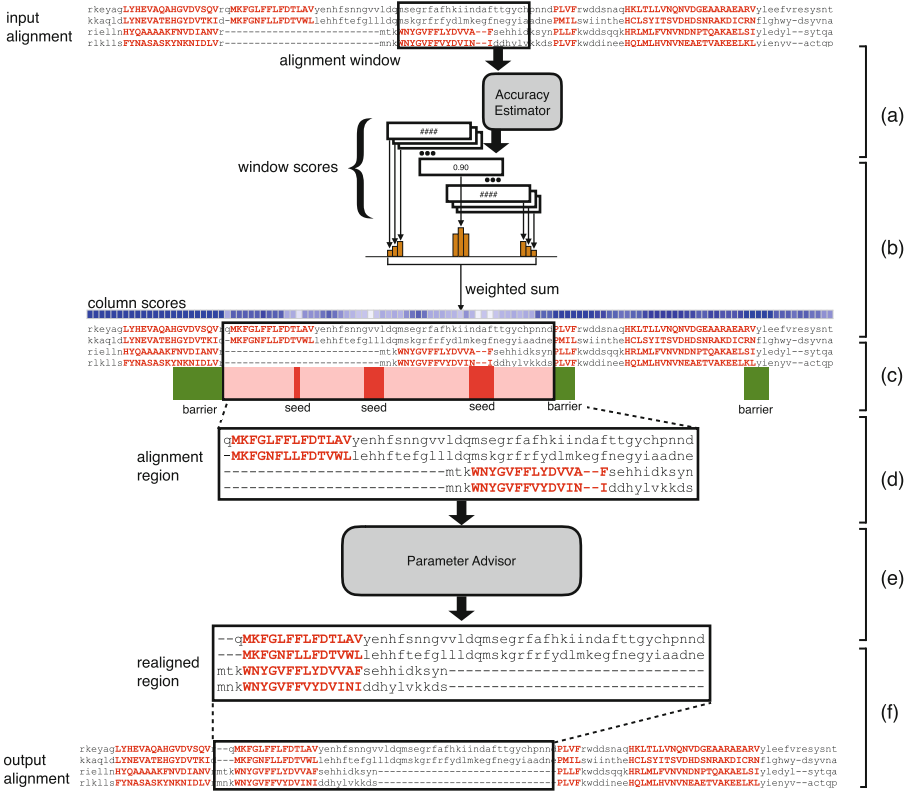


Fig. 4. *The adaptive local realignment process.* (a) Estimate the accuracy for sliding windows across the input alignment using **Facet**. (b) Calculate a score for each column as the weighted sum of the accuracies of all windows that overlap the column. (c) Label columns that are above τ_S or below τ_B as seeds or barriers, respectively. (d) Define realignment regions that will be extracted from the alignment by extending seeds in both directions until they reach a barrier. (e) Use parameter advising to find a new alignment of each realignment region. (f) Replace the original realignment region if the new alignment is more accurate.

3.1 Identifying Local Realignment Regions

When selecting alignment columns that should be saved we cannot simply identify correctly recovered columns in a computed alignment since just as with global alignments we do not have a known reference in practice. But we can identify these regions using an accuracy estimator E which we defined earlier. To partition the input alignment we first calculate the estimated accuracy of a sliding window across the alignment (Fig. 4a). The window size is a fraction of $w \leq 1$ of the total length of the alignment. The value of w must be chosen carefully because the accuracy estimator has features that reflect global properties of an alignment. A larger sliding window will provide more context at each

position and should provide a better estimate of accuracy. At the same time, if the window is too large there will not be fine enough granularity to identify the transition points between correctly- and incorrectly-aligned columns. Additionally, we define upper and lower bounds on the absolute window size to account for very short and very long alignments.

A score is assigned to each column as the sum of the approximately $\frac{1}{w}$ window scores that overlap that column weighted proportionally to the distance to the center column of the window (Fig. 4b). A geometric distribution, with a decay rate $d < 1$, centered on the middle column is used to determine the contribution of a window to the scores of each of the columns it covers. As d approaches 1, a column gets equal weight from all covering windows; as it approaches 0, the score is dependent only on the window centered at that column.

From the column scores we generate a partitioning by labeling columns for which there is the most evidence of being correctly (or incorrectly) aligned. Given the minimum percentage of columns we would like to retain from the input alignment, T_B , and the minimum percentage of columns we would like to replace, T_S , we calculate two threshold values τ_B and τ_S such that the number of columns with score greater than τ_B is at least $\lceil \ell T_B \rceil$, and the number of columns with score less than τ_S is at least $\lceil \ell T_S \rceil$. We can then label all columns with score at least τ_B as *barriers*—these columns are guaranteed to be retained—and those with column score at most τ_S are labeled as *seeds*—these columns are guaranteed to be realigned (Fig. 4c). Finally, we define realignment regions by extending each seed in both directions until a barrier column is reached (or the first or last column of the alignment). Note that a realignment region may contain more than one seed column but will never include one of the barriers. Using this method we ensure that: at least ℓT_B columns from the original alignment will be in the final alignment, there will always be at least one realignment region, and there will never be a realignment region that covers all columns of the input.

3.2 Local Parameter Advising on a Region

The realignment regions defined above identify subalignments that have the potential to be improved and we will use parameter advising to produce better alignments of these regions and replace them in the input alignment. We extract the subalignment from the input identified by the columns in each alignment region (Fig. 4d). Removing the gaps from this subalignment yields a set of unaligned sequences which becomes the input to a slightly modified version of the parameter advising method described earlier (Sect. 2.1, Fig. 2) which considers the location of the realignment region within the alignment scoring scheme (Fig. 4e). The `Opal` aligner scores terminal and internal gaps separately but for the case of adaptive local realignment we only apply terminal gap scores when the terminal column in the context of the subalignment is also the terminal column in the context of the global alignment. As mentioned earlier an alignment region will never include both terminals.

Once we have obtained the new alignment via parameter advising the final step is to replace the original region in the input (Fig. 4f) if the **Facet** score is higher than that of the original subalignment for the realignment region.

After all realignment regions have been updated by local advising we make one last advising decision between the new alignment and the input alignment. The more accurate global alignment of the two is returned.

3.3 Iterative Local Realignment

The adaptive local realignment process corrects misalignments in the input, but after performing the procedure there may still be some regions of the alignment that can be improved. These remaining regions may not have been identified originally because they are subregions within a newly-included alignment, or because the threshold was too low for a seed to be identified due to the very low quality of other another region. In either case, it would be beneficial to repeat adaptive local realignment to further increase accuracy. Therefore, we iterate the whole process (Fig. 4) until a user-defined maximum number of iterations is reached, or no further improvements are made.

3.4 Combining Local and Global Advising

The quality of the alignment input to the adaptive local realignment process is critical since adaptive local realignment is only making local improvements. Therefore, we would like to use global parameter advising, which has been shown to improve accuracy [15], to identify the best initial alignment. Local and global advising can then be combined in two ways:

- (1) local advising on *all* global alignments, using adaptive local realignment on each of the alternate alignments produced within global parameter advising, and then choosing among all $2|P|$ alternate alignments (for $|P|$ unaltered global alignments, and $|P|$ locally-advised alignments); and
- (2) local advising on the *best* global alignment, which chooses the best global alignment, and then uses adaptive local realignment to boost its accuracy.

We compare both ways of combining local and global advising, as well as local advising on the default alignment, in the next section.

4 Assessing Adaptive Local Realignment

We evaluate the performance of adaptive local realignment and its use in combination with global advising through experiments on a collection of protein multiple sequence alignment benchmarks. A full description of the benchmarks and universe of parameters used for parameter advising can be found in [15] and is briefly described here.

The benchmark suites used in our experiments consist of reference alignments of proteins that are largely induced by structurally aligning their known three-dimensional structure. In particular, we use the **BENCH** suite of Edgar [9] (which

is a combination of the BALiBASE [1], PREFAB [10], OxBench [20], and SABRE [24] databases), supplemented by a selection from the PALI suite of Balaji et al. [2]. The full benchmark collection we use consists of 861 reference alignments.

As is common in benchmark suites, easy-to-align benchmarks are highly over-represented in this collection. To correct for this bias towards easy to align benchmarks when evaluating average advising accuracy, we binned the 861 benchmarks by *hardness*, which we measured by the true accuracy of `Opal` using the default parameter setting. We then divided the the full range $[0, 1]$ of accuracies into 10 bins, where bin b for $b = 1, \dots, 10$ contains hardness interval $((b - 1)/10, b/10]$, and has 12, 12, 20, 34, 26, 50, 62, 74, 137, and 434 benchmarks respectively. We report the average accuracy across *bins* rather than across benchmarks. This means that the average accuracy of alignments using the `Opal` default parameter settings is near 50%. Even though the binning is based on the `Opal` default alignments, most other standard aligners have default accuracy near 50% as well: `Clustal Omega` [22], 47.3%; `MUSCLE` [10], 48.4%; `MAFFT` [14], 51.0%. The methodology presented here is general and can be implemented for any other aligner.

We developed a universe of alignment parameter settings by enumerating reasonable values of each of the tunable alignment parameters for the `Opal` aligner. In particular the tunable parameters for `Opal` are represented as a 5-tuple $(\sigma, \lambda_I, \lambda_T, \gamma_I, \gamma_T)$ which represent the replacement matrix (σ) and the internal and terminal gap open (λ) and extension costs (γ). For the substitution matrix we selected 3 matrices from the BLOSUM [13] and VTML [18] families, two choices of terminal gap extension costs, and three choices each of internal gap extension, terminal gap open and internal gap open costs. In total we generate a universe of 162 parameter settings.

We use 12-fold *cross validation* to examine the increase in accuracy gained using adaptive local realignment. We first evenly and randomly distributed benchmarks into twelve groups for each hardness bin; the 12 independent folds are generated by choosing one group from each bin to be in the *testing set*, and the other eleven to be in the *training set*. Finally, we generate an alignment for each benchmark in the training or testing set of each fold using each of the parameters in our universe and the `Opal` aligner. The results we reported are averages over these twelve folds. (Note that across twelve folds, every example is tested on exactly once.)

We trained the estimator coefficients for `Facet` on the training example sets for each fold using the difference fitting method described in Sect. 2.2. We found that there was very little change in coefficients between the training folds so for simplicity we use the estimator coefficients that are release with the newest version `Facet` which were trained on all available benchmarks. We also use the TCS estimator for adaptive local realignment, these results are in Sect. 4.3.

To choose the parameters for adaptive local realignment, we tested the cross product of reasonable values for the tunable parameters. We used the performance on *training* benchmarks described above to find the combination of these settings that gave the highest improvement in accuracy when local advising was

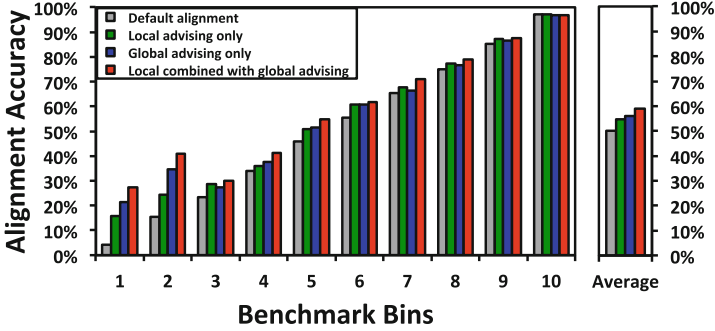


Fig. 5. Accuracy of the default alignment, and different advising methods, within difficulty bins. The horizontal axis shows all ten benchmarks bins. The vertical axis shows the accuracy averaged over just the benchmarks in that bin using default parameter settings, local advising only, global advising only, and the combined advising method using an oracle set of cardinality $k = 10$. The bar chart on the right shows the accuracy uniformly averaged over the bins.

applied to the default alignments from `Opal`. Table 1 summarizes the tunable parameters, the range of values over which we tested and the value we selected for use in our experiments. Details on the iteration count selection are in Sect. 4.4.

4.1 Effect of Local Realignment Within Difficulty Bins

Figure 5 shows the alignment accuracy across difficulty bins for default alignments from `Opal`, local advising on these default alignments, global advising alone, and local combined with global alignment. Here the combination method uses local advising on all alternate alignments within global advising. The oracle set of cardinality $k = 10$ was used for both global and local advising.

The improvement gained by using adaptive local realignment over the default parameter setting is most evident in the two most difficult benchmark bins using local advising increases the average accuracy by 11.5% and 9.1% respectively,

Table 1. Adaptive local realignment parameter selection

Parameter	Range of values	Chosen
Window length fraction, w	0.05, 0.1, 0.2, 0.3, \dots , 0.7	0.3
Window length lower bound	5, 10, 20, 30	10
Window length upper bound	30, 50, 75, 100, 125	30
Barrier label percentages, T_B	5%, 10%, 20%, 30%, \dots , 70%	10%
Seed label percentages, T_B	5%, 10%, 20%, 30%, \dots , 70%	30%
Geometric decay rate, d	0.5, 0.66, 0.9, 0.99	0.9
Iterations	1, \dots , 5, 10, 15, 25	5

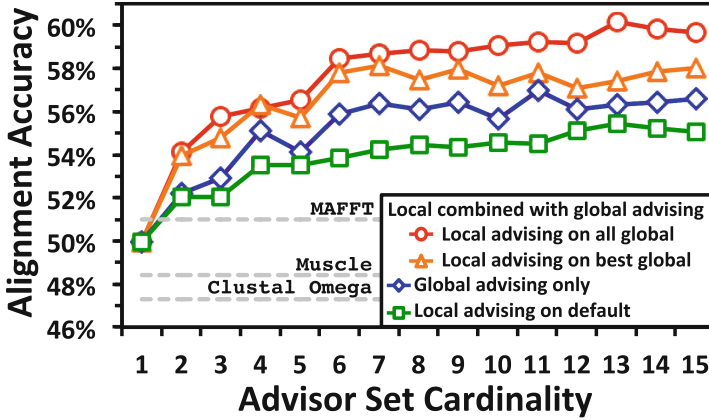


Fig. 6. *Advising accuracy versus advisor set cardinality.* The horizontal axis is the cardinality of the advisor set used by the advising methods. The vertical axis shows the advising accuracy of the default parameter setting, local advising, global advising, and the combined advising method, averaged across difficulty bins.

but the accuracy increases on all bins. Overall using local advising increases the accuracy of the default alignments by an average of 4.5% across bins.

Combining local and global advising substantially improves the accuracy over either of the methods individually. This is most pronounced for the hardest to align benchmarks. For the bottom two bins using both parameter advising and adaptive local realignment increases the accuracy by 23.0% and 25.6% accuracy over using just the default parameter choices. Additionally, using adaptive local realignment increases the accuracy by 5.9% and 6.4% accuracy on the bottom most bins over using parameter advising alone. On average that's an 8.9% increase in accuracy over all bins by using the combined procedure over using just the default parameter choice and a 3.1% increase over using only parameter advising.

4.2 Varying Advisor Set Cardinality

Since an alignment is produced for each region of local realignment for each parameter choice in the advisor set, and the running time is dependent on the size of the advisor set, it may be desirable to use a smaller set than used in the previous section to reduce the running time of local (and global) advising. We produced oracle advisor sets for cardinalities $k = 2, \dots, 15$ and used them to test the effect of local advising both alone and in combination with global advising. Figure 6 shows the average advising accuracies of using advisor sets of increasing cardinalities for adaptive local realignment applied to the `Opal` default alignment, global advising alone, and global combined with local advising. The figure shows the accuracy of both combined local and global advising strategies described in Sect. 3.4: adaptive local realignment applied to best global alignment found through advising and adaptive local realignment applied to all global

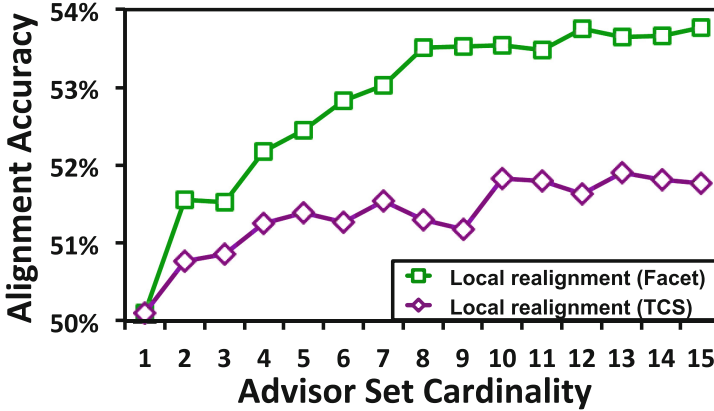


Fig. 7. Accuracy of the default alignment and local realignment using TCS and Facet with various advisor set cardinalities. This figure compares the accuracy of alignments produced by the `Opal` default parameter settings applying local realignment using either the TCS or Facet estimator. The horizontal axis is the cardinality of the oracle advising set used for local realignment. The vertical axis shows the accuracy of the alignments produced by each of the advising methods, averaged across difficulty bins.

alignments. The cardinality of the set used for both global and local advising is shown on the horizontal axis, while the vertical axis shows alignment accuracy uniformly averaged across bins.

The accuracy of alignments produced by all four methods shown eventually reaches a plateau where adding additional parameters to the advisor set no longer increases the alignment accuracy. This plateau is reached at cardinality $k = 10$ when local realignment is applied to the default alignments and at $k = 6$ for parameter advising with and without local realignment, but this plateau is higher for the combined methods. Across all cardinalities using local combined with global advising improves alignment accuracy by nearly 4% on average. Note that when local realignment is applied to all global alignments the advisor is now choosing from a set of alignments which have higher accuracy than their corresponding original alignments.

The results above uniformly average advising accuracy across *bins*. In contrast, if we report advising accuracy uniformly averaged across *benchmarks*, `Opal` on its default parameter choice achieves accuracy 80.4%, local or global advising alone increases this accuracy to 82.1% and 81.8% respectively, and combining both methods increases the accuracy to 83.1% (all at cardinality $k = 10$). Other aligners have accuracies: `Clustal Omega`, 77.3%; `MUSCLE`, 78.1%; `MAFFT`, 79.4%.

4.3 Comparing Estimators for Local Advising

Figure 7 shows the average accuracy of local advising on default alignments using both Facet and TCS (the next-best estimator for advising [5, 15]). These results

used only a single iteration of adaptive local realignment for both estimators, due to the large increase in running time caused by calls to the external TCS program. Using TCS for local advising does increase accuracy over the default alignment, but the increase is less than half that of **Facet**.

4.4 Effect of Iterating Local Realignment

As discussed in Sect. 3.3, iterating local advising should eventually reach a state where the alignment is no longer improving, or even worse, begins deteriorating due to noise in the accuracy estimator. Table 2 shows the average accuracy of using local adaptive realignment on the default alignment as the number of iterations is increased. The training accuracy reaches a plateau at 5 iterations; we use this number of iterations in Sects. 4.1 and 4.2.

4.5 Summarizing the Effect of Adaptive Local Realignment

Table 3 summarizes how adaptive local realignment behaves across difficulty bins during the first iteration of improving **Opal** default alignments. The columns are average values for each of the 10 benchmark bins, and average values across all benchmarks. The first three rows show how many of the 861 benchmarks are in each bin, as well as the number and percentage of those had at least one realignment region in the alignment that was replaced. The last three rows summarize how much of each alignment changed. The fourth row shows the average number of realignment regions found for each benchmark; on average about 2 regions were realigned for each default alignment. The last two rows summarize the percentage of the original columns that were in realignment regions, and how many of the columns from the original alignment were replaced. Notice that while the percentage of columns covered by realignment regions stays roughly the same

Table 2. Accuracy of adaptive local realignment across iterations

Iterations	1	2	3	4	5	10	15	25
Testing	53.5%	53.7%	54.1%	54.4%	54.5%	54.5%	54.5%	54.5%
Training	53.5%	53.9%	54.5%	54.6%	54.8%	54.8%	54.9%	54.9%

Table 3. Summary of adaptive local realignment on default alignments

Bin	1	2	3	4	5	6	7	8	9	10	Overall
Number of benchmarks	12	12	20	34	26	50	61	74	137	434	861
Number modified	8	7	16	27	19	34	46	61	115	352	685
Percentage modified	67%	58%	80%	79%	73%	68%	74%	82%	84%	81%	80%
Regions per benchmark	1.92	2.17	2.50	1.88	2.23	2.14	2.31	2.16	2.48	2.19	2.23
Columns realigned	75%	73%	76%	70%	75%	77%	74%	73%	75%	72%	73%
Columns replaced	64%	60%	68%	60%	66%	72%	65%	63%	64%	47%	57%

in the easiest-to-align benchmark bin, only 47% of the alignment columns were altered, while in the rest of the bins over 60% of the alignment columns improved.

4.6 Running Time

The running time of `Opal` with adaptive local realignment, averaged across all benchmarks, increases to 110 seconds when using an advisor set of cardinality $k=10$, and 5 iterations. This is up from 36 seconds for one iteration, and about 8 seconds for the default parameter settings. This high increase in wall-clock time is mainly due to the fact that, as currently implemented, adaptive local realignment does not exploit parallelism in advising. In contrast, *global* advising has been parallelized, so the average running time of global advising on the same advisor set of size $k=10$ is only around 33 seconds. Note that the number of columns being repeatedly aligned by global advising is about a factor 1.25 more than for local advising. When the two methods are combined, the average running time increases to 68 and 178 seconds for local advising on the best global alignment, and local advising on all global alignments, respectively.

5 Conclusion

We have presented *adaptive local realignment*, the first approach that demonstrably boosts protein multiple sequence alignment accuracy by adaptively realigning regions with local parameter settings. Applying this to alignments initially computed using an optimal default parameter setting significantly improves alignment accuracy; when combined with global parameter advising to select an initial parameter setting, this new approach to local advising boosts accuracy greatly.

Acknowledgements. Research of JK and DD at Arizona was funded by NSF Grant IIS-1217886 to JK. DD was partially supported at Carnegie Mellon by NSF Grant CCF-1256087, NSF Grant CCF-131999, NIH Grant R01HG007104, and Gordon and Betty Moore Foundation Grant GBMF4554, to Carl Kingsford.

References

1. Bahr, A., Thompson, J.D., Thierry, J.C., Poch, O.: **BAl**iBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Res.* **29**(1), 323–326 (2001)
2. Balaji, S., Sujatha, S., Kumar, S., Srinivasan, N.: **PALI**—a database of Phylogeny and ALignment of homologous protein structures. *NAR* **29**(1), 61–65 (2001)
3. Chang, J., Tommaso, P., Notredame, C.: A new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. *Mol. Biol. Evol.* **31**(6), 1625–1637 (2014)
4. DeBlasio, D., Kececioglu, J.: **Facet**: software for accuracy estimation of protein multiple sequence alignments (2014). facet.cs.arizona.edu
5. DeBlasio, D., Kececioglu, J.: Learning parameter-advising sets for multiple sequence alignment. *IEEE/ACM Trans. Comput. Biol. Bioinform.* (2015). doi:[10.1109/TCBB.2015.2430323](https://doi.org/10.1109/TCBB.2015.2430323)

6. DeBlasio, D.F., Wheeler, T.J., Kececioglu, J.D.: Estimating the accuracy of multiple alignments and its use in parameter advising. In: Chor, B. (ed.) RECOMB 2012. LNCS, vol. 7262, pp. 45–59. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-29627-7_5](https://doi.org/10.1007/978-3-642-29627-7_5)
7. DeBlasio, D.F.: Parameter Advising for Multiple Sequence Alignment. Ph.D. dissertation, Department of Computer Science, The University of Arizona, May 2016
8. Do, C., Mahabhashyam, M., Brudno, M., Batzoglou, S.: Probabilistic consistency-based multiple sequence alignment. *Genome Res.* **15**(2), 330–340 (2005)
9. Edgar, R.C.: **BENCH** (2009). drive5.com/bench
10. Edgar, R.: **MUSCLE** multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**(5), 1792–1797 (2004)
11. Fitch, W.M., Margoliash, E.: A method for estimating the number of invariant amino acid coding positions in a gene using cytochrome c as a model case. *Biochem. Genet.* **1**(1), 65–71 (1967)
12. Gotoh, O.: Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput. Appl. Biosci.* **9**(3), 361–370 (1993)
13. Henikoff, S., Henikoff, J.: Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* **89**(22), 10915–10919 (1992)
14. Katoh, K., Kuma, K.I., Toh, H., Miyata, T.: **MAFFT** version: 5 improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.* **33**(2), 511–518 (2005)
15. Kececioglu, J., DeBlasio, D.: Accuracy estimation and parameter advising for protein multiple sequence alignment. *J. Comput. Biol.* **20**(4), 259–279 (2013)
16. Kececioglu, J., Starrett, D.: Aligning alignments exactly. In: Proceedings of the 8th Conference on Research in Computational Molecular Biology (RECOMB), pp. 85–96. ACM (2004)
17. Löytynoja, A., Goldman, N.: Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science* **320**(5883), 1632–1635 (2008)
18. Müller, T., Spang, R., Vingron, M.: Estimating amino acid substitution models: a comparison of Dayhoff’s estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.* **19**(1), 8–13 (2002)
19. Notredame, C., Higgins, D., Heringa, J.: **T-Coffee**: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* **302**(1), 205–217 (2000)
20. Raghava, G., et al.: **OXBench**: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinform.* **4**(1), 1–23 (2003)
21. Roskin, K.M., Paten, B., Haussler, D.: Meta-alignment with **Crumble** and **Prune**: partitioning very large alignment problems for performance and parallelization. *BMC Bioinform.* **12**(1), 1–12 (2011)
22. Sievers, F., et al.: Fast, scalable generation of high-quality protein multiple sequence alignments using **Clustal Omega**. *Mol. Sys. Biol.* **7**(1), 539 (2011)
23. Thompson, J., Higgins, D., Gibson, T.: **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)
24. Van Walle, I., Lasters, I., Wyns, L.: **SABmark**: a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics* **21**(7), 1267–1268 (2005)
25. Wheeler, T.J., Kececioglu, J.D.: Multiple alignment by aligning alignments. *Bioinformatics* **23**(13), i559–i568 (2007)
26. Yang, Z.: Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Mol. Biol. Evol.* **10**(6), 1396–1401 (1993)

A Concurrent Subtractive Assembly Approach for Identification of Disease Associated Sub-metagenomes

Wontack Han, Mingjie Wang, and Yuzhen Ye^(✉)

Indiana University, Bloomington, IN, USA
yye@indiana.edu

Abstract. Comparative analysis of metagenomes can be used to detect sub-metagenomes (species or gene sets) that are associated with specific phenotypes (e.g., host status). The typical workflow is to assemble and annotate metagenomic datasets individually or as a whole, followed by statistical tests to identify differentially abundant species/genes. We previously developed subtractive assembly (SA), a *de novo* assembly approach for comparative metagenomics that first detects differential reads that distinguish between two groups of metagenomes and then only assembles these reads. Application of SA to type 2 diabetes (T2D) microbiomes revealed new microbial genes associated with T2D. Here we further developed a Concurrent Subtractive Assembly (CoSA) approach, which uses a Wilcoxon rank-sum (WRS) test to detect k-mers that are differentially abundant between two groups of microbiomes (by contrast, SA only checks ratios of k-mer counts in one pooled sample versus the other). It then uses identified differential k-mers to extract reads that are likely sequenced from the sub-metagenome with consistent abundance differences between the groups of microbiomes. Further, CoSA attempts to reduce the redundancy of reads (from abundant common species) by excluding reads containing abundant k-mers. Using simulated microbiome datasets and T2D datasets, we show that CoSA achieves strikingly better performance in detecting consistent changes than SA does, and it enables the detection and assembly of genomes and genes with minor abundance difference. A SVM classifier built upon the microbial genes detected by CoSA from the T2D datasets can accurately discriminates patients from healthy controls, with an AUC of 0.94 (10-fold cross-validation), and therefore these differential genes (207 genes) may serve as potential microbial marker genes for T2D.

Keywords: Metagenome · Concurrent Subtractive Assembly · Wilcoxon rank-sum test · Comparative metagenomics

1 Introduction

The human body is host to trillions of bacteria cells, outnumbering human cells by 1.3 to 1 (in contrast to the widely cited 10:1 ratio), according to a recent

W. Han and M. Wang—These authors contributed equally to this work.

estimate [40]. Moreover, the genes encoded by human microbiome are hundreds of times more than the human complement [47]. It has been reported that those microorganisms are involved in $\sim 20\%$ of human malignancies [6]. The gut microbiota has been linked to a variety of conditions including inflammatory bowel disease [23], cardiovascular disease [19], rheumatoid arthritis [38], Parkinson's disease [37], autism spectrum disorder [16], colon cancer [9, 39], and liver cirrhosis [34], among others. However, only 10 microbes are designated to be carcinogenic to human beings by the International Agency for Cancer Research (IACR) [6]. Therefore, it is intriguing to explore microbes that are directly related to the development of human diseases.

The development of next generation sequencing has pushed the advancement of metagenomics, which presents us a great opportunity to identify microorganisms that are enriched or depleted during disease and explore possible mechanisms behind the association. The human microbiome project has shown the association between the shifts in our microbiota and diseases such as obesity [17] and periodontitis [15]. Although the change in identify of the species (or abundance) does not ensure a causal role for the microbes, we can narrow down the set of candidate genomes or genes by such studies. One new trend of microbiome research is microbiome-wide association studies (MWAS), which are analogous to genome-wide association studies (GWAS) [20]. MWAS may take a case-control approach, revealing the association between microbiomes and human diseases. However, the limitation of this approach is that it cannot distinguish whether the microbiome drives the disease, the disease drives the microbiome, or both are modified by confounding factors. On the other hand, longitudinal studies may allow researchers to test whether changes in the microbiome precede or follow the development of disease [5, 11].

In the seeking of disease-associated microbes, we should note the significant compositional variations of microbiota from individual to individual [10]. Regarding this interpatient variability, the correct strategy is to identify conserved microbial community behaviors in microbiota-associated diseases [15]. Microbial marker gene surveys have been used extensively to reveal the association of microbiota with diseases such as diabetes and Crohn's disease [31]. For instance, Qin et al. identified 15 optimal marker genes from the gut microbiome in liver cirrhosis by comparing 98 patients and 83 healthy control individuals [34]. Based on only the 15 biomarkers, they were able to construct a classifier that can discriminate patients with a decent accuracy [34]. Similarly, gut microbiota was explored to detect colorectal cancer and a metagenomic classifier was trained using the taxonomic abundances of 22 marker species [45]. The typical workflow of these marker-gene surveys is to assemble the metagenomes and then predict the genes, potential marker genes can then be identified by detecting significant differences in their distribution across healthy and disease populations. The analysis of differential abundance is critical for these surveys and computational tools have been developed for the analysis, including a recently developed approach that relies on a novel normalization technique and a statistical model accounting for undersampling [31].

Due to the complexity of microbial communities, the de novo partition of metagenomic space into specific biological entities remains to be difficult. To address this problem, researchers have utilized various features, including compositional features such as tetra-nucleotide statistics [13] and coverage signals of genetic sequences [1, 44]. However, the assumptions of those methods are not universally true. For example, the methods relying on abundances of genetic sequences are admittedly weak in segregating taxonomically related organisms [1]. In the process of exploring other features, it has been realized that utilizing co-abundance across multiple samples improves the resolution of genome segregation from metagenomic data sets [2, 29, 43]. Similarly, we should also utilize information from multiple samples for the sake of identifying conserved differential patterns.

We have previously introduced a method called subtractive assembly (SA) [42], which is a de novo method to compare metagenomes by identifying and assembling the differential reads. We have demonstrated that SA can recover the differential genomes by effectively extracting the differential reads based on sequence signatures (frequencies of k-mers). Also, SA can improve the quality of metagenomic assembly when only a subset of closely-related genomes change in their abundances between the groups of samples in comparison. Application of SA to gut metagenomes from women with type 2 diabetes (T2D) [17] reveals compositional features and a large collection of unique or abundant genes in T2D gut metagenomes (some of the genes identified by SA were otherwise missed by direct assembly of the original datasets). SA utilizes both the compositional and coverage features through the composition and frequency of k-mers, contributing to its superior performance. However, the SA method pools the samples for each group before comparison and therefore loses power in detecting minor but consistent changes without using information from individual samples. In addition, SA picks up genes in species which only appear in a few samples but with high abundances, as a result, many of the “differential” genes assembled are not actually consistently abundant across samples in the same group. Therefore additional profiling of gene abundance is required in order to search for genes consistently more abundant in one group versus the other.

In this paper, we further developed the subtractive assembly approach for the detection of consistently differential genomes or genes by using k-mer frequencies in individual samples (co-abundance). We adopted KMC 2 [7] for k-mer counting in our implementation, since KMC 2 is one of the fastest k-mer counting approaches, which was claimed to be twice faster than the strongest competitors such as Jellyfish 2 [26]. Differential reads extracted from individual samples were then pooled for assembly. We call our new method Concurrent Subtractive Assembly approach (CoSA). We observed that some reads are extremely redundant (those sampled from abundant common species across samples). We further developed a strategy to remove redundant reads based on k-mer counts: only some of the reads that contain highly abundant k-mers are retained for assembly. Using simulated datasets, we showed that CoSA achieves much better performance in detecting consistent changes than the original subtractive assembly (SA) approach. Moreover, we applied it to analyzing T2D gut metagenomes

to identify microbial marker genes, based on which we built a classifier that accurately discriminates patients from healthy controls.

2 Materials and Methods

2.1 Overview

Concurrent Subtractive Assembly (CoSA) is designed to identify the short reads that make up the conserved/consistent compositional differences across multiple samples based on sequence signatures (k-mer frequencies), and then to only assemble the differential reads, aiming to reveal the consistent differences between two groups of metagenomic samples (e.g., metagenomes from cancer patients vs. metagenomes from healthy controls).

2.2 k-mer Counting

CoSA is a k-mer-based method, and therefore the first step is the counting of all k-mers in metagenomic samples. For comparative metagenomic studies, the sheer size of the datasets is a fundamental challenge. We employed KMC 2 for k-mer counting. We specified the maximal value of a count (the `cs` flag) as 65,536 instead of 255 by default. On one hand this helps identify the more frequently observed differential k-mers by using a larger cut-off value; on the other hand we can store each count using a 16-bit unsigned integer, which demands a reasonable amount of memory or disk space when dealing with billions of k-mers. Meanwhile, we exclude k-mers occurring less than two times by the `ci` option based on the fact that a large number of singletons are products from sequencing errors, as previously employed by both BFCOUNTER [28] and khmer [46].

After k-mer counting with KMC 2, CoSA goes through the outputs of KMC by using the KMC API and stores all observed k-mers in a hash table, implemented using the libcuckoo library (downloaded from <https://github.com/efficient/libcuckoo>). Libcuckoo [25] provides a high-performance concurrent hash table, by which we can efficiently update the hash table using multiple threads. With the k-mers in the hash table, CoSA accesses the outputs of KMC again and writes to disk the counts of the k-mers based on their orders in the hash table for every sample. By storing the counts on the disk, we can load the counts of k-mers in batches and therefore significantly reduce the memory requirement for recording the counts of all k-mers in every sample.

2.3 Identification of Differential k-mers Using Wilcoxon Rank-Sum Test

CoSA by default loads 10^7 k-mers into a two-dimensional array each time and iteratively tests if the frequencies of each k-mer are differential between the two groups of samples. To compare k-mers in different metagenomic samples, we calculate the frequency of each k-mer in each metagenomic sample. In case the

frequency of a rare k-mer is extremely small, we compute the frequency of a k-mer as the number of occurrences per million k-mers. Then the normalized frequencies are used for WRS test (a nonparametric test), for which we employ the “mannwhitneytest” function from ALGLIB (<http://www.alglib.net>). The WRS test is used to detect k-mers that have different frequencies in one group of the samples (e.g., the patient group) than the other group of samples (e.g., the healthy control) with statistical significance. The k-mers that pass the test (p-value cut-off is set to 0.05 by default) are identified as differential k-mers.

We tested different k-mer sizes empirically. Bigger k-mer size increases the memory assumption by CoSA, but has very little impact on the results of extracted reads and downstream application of the reads. We therefore set the default k-mer size to 23.

2.4 Identification of Differential Reads Based on Differential k-mers

Reads that are composed of differential k-mers tend to be from differential genomes. Thus, we extract differential reads in each sample based on the differential k-mers using a voting strategy. With the voting threshold as 0.5, for example, a read is considered to be differential if 50% of its k-mers belong to differential k-mers. We empirically tested the voting threshold and found a value in the range of 0.3–0.8 gives a good balance between the number of extracted reads and efficiency of the differential gene assembly. However, users may change this parameter (`-v`) in their own applications of CoSA.

2.5 Reduction of Reads Redundancy

We noticed that some k-mers are extremely abundant in the extracted reads file (these k-mers are likely from the reads sampled from abundant species that are common across many samples). When the differential reads contain these k-mers, the distribution of k-mers is skewed and this can challenge the assembly algorithm. To address this issue, we reduced the reads redundancy by excluding reads that contain highly abundant k-mers. The reads redundancy removal relies on a list of highly abundant k-mers prepared based on k-mer counts. A read is determined to be redundant if it contains many k-mers on the abundant k-mer list. Specifically, for each read, the fraction of abundant k-mer (over all k-mers) is computed and used for determining the fate of the read: if the fraction is smaller than a random number between 0 and 1 generated by the program, the read is retained; otherwise, it is discarded. In this way, a read that has a higher ratio of abundant k-mers will have a higher chance to be discarded.

2.6 Assembly of Extracted Reads and Downstream Annotations

Following the read extraction, any metagenomic assembler can be employed in subtractive assembly. Here, we used MegaHIT (with meta-large presets option) [24] (version 1.0.2) to assemble the differential reads, to illustrate the usage

of CoSA. For each group (e.g., T2D patients, or healthy controls), differential reads extracted from individual samples were pooled and assembled together by MegaHIT. We note that we only pooled reads from multiple samples in the same group for assembly. We used MegaHIT as it is one of the recently developed assemblers that are memory efficient and fast. But in principle, other assemblers such as IDBA-UD [33] and metaSPAdes [3] can be used as well. In order to identify differential genes, protein coding genes were predicted from the contigs using FragGeneScan [35] (version 1.30).

To estimate the abundance of the genes, all the reads from each sample were aligned against the gene set by using Bowtie 2 [22] (version 2.2.6). We counted a gene’s abundance based on the counts of both uniquely and multiply mapped reads. The contribution of multiply mapped reads to a gene was computed according to the proportion of the multiply mapped read counts divided by the gene’s unique abundance [34]. The read counts were then normalized per kilobase of gene per million of reads in each sample.

2.7 Building Classifiers

After the gene abundance profile was built, we attempted to build a classifier that can discriminate patients from healthy controls. We first used L1-based feature selection method in the “scikit learn” python package [32] to select genes. After the feature selection, we built classifiers using Random Forest (RF) and Support Vector Machine (SVM). We used RF as it has been shown to be a suitable model for exploiting non-normal and dependent data such as metagenomic data [18] and it was used for prediction of T2D in [17]. On the other hand, SVMs are widely used in computational biology due to their high accuracy and their ability to deal with high-dimensional and large datasets [4]. We used the SVM (linear kernel) and RF (10 trees) in the “scikit learn” python package. We evaluated the predictive power of a model as the Area Under Curve (AUC) using a tenfold cross-validation method.

We tested different p-value cut-offs and voting thresholds used in CoSA for evaluating their impact on the accuracy of the classifiers built from genes derived by CoSA.

2.8 Simulated and Real Metagenome Datasets

To test the performance of CoSA in detecting minor effects, we first generated two groups of metagenomic datasets using five bacterial genomes from the FAMEs dataset [27] by MetaSim [36], with each group representing a unique population structure; and for each group, we simulated 10 samples.

As a showcase for CoSA, we further applied our method to the T2D cohort. The T2D cohort was derived from two groups of 70-year-old European women, one group of 50 with T2D and the other a matched group of healthy controls (NGT group; 43 participants). We did not use 3 samples of T2D datasets that were outliers based on neighbor-joining clustering using a d_2^S dissimilarity measure for $k = 9$ [14]. We tested our original SA approach using the T2D cohort,

and in this study, we focused on the comparison of CoSA with SA using the T2D datasets. Table 1 summarizes the simulated datasets and the T2D microbiome datasets we used for testing.

Table 1. Summary of the simulated and T2D datasets.

	Simulated	T2D and healthy
Number of datasets	20	93
Total bps	2.29 Gbp	225.30 Gbp
Number of k-mers	9,112,554	4,121,225,700

2.9 Availability of CoSA

We implemented CoSA in C++. Because CoSA employs k-mer frequencies from individual samples, it introduces a new dimension for different samples and therefore increases the requirement of computational resources, especially for large cohort of datasets such as the T2D datasets. To reduce the running time and memory usage, we implemented CoSA with multiple threading. Also, counts of k-mers are written to disk and then loaded back in batches for the detection of differential k-mers (since it is impossible to load all k-mer counts into the memory at the same time). The software is available for download at sourceforge (<https://sourceforge.net/projects/concurrentsa/>).

3 Results

We first report the results of CoSA using simulated datasets. We then report the comparison of CoSA with our original SA method using the T2D cohort. Finally we report the results of using CoSA for extracting and charactering disease associated sub-microbiome using the T2D datasets.

3.1 Evaluation of CoSA Using Simulated Datasets

Instead of using fold change of k-mers, CoSA detects differential genomes by testing k-mer frequencies with Wilcoxon rank-sum test. Also, it employs k-mer frequencies concurrently from multiple samples for each group in comparison. In theory CoSA has the capability of detecting minor but consistent changes between groups of samples. To test the performance of CoSA in such case, we simulated metagenomic samples using two population (community) structures (Table 2). The *Streptococcus thermophilus* LMD-9 genome is two times more in population one (P1) than in population two (P2) in terms of relative abundance. Similarly, *Prochlorococcus marinus* NATL2A is the differential genome that is two times more abundant in P2 than in P1. Since there is only a fold change of

two for the differential genomes, it is hard to detect the minor effects through fold change of k-mers (as a result SA performed poorly on this simulated dataset; see below).

We evaluated CoSA with different parameters, including p-value cut-off and number of samples for each group in comparison. First, we compared the efficacy of read extraction using either 5 or 10 samples for each population. The results show that CoSA extracted more reads from the differential genomes by using more samples (Fig. 1). For example, using a p-value cut-off of 0.005, CoSA extracted 593,739 (99.98%) out of 593,858 short reads (expected) for the *S. thermophilus* LMD-9 genome when 10 samples were used (see Table 2). When using

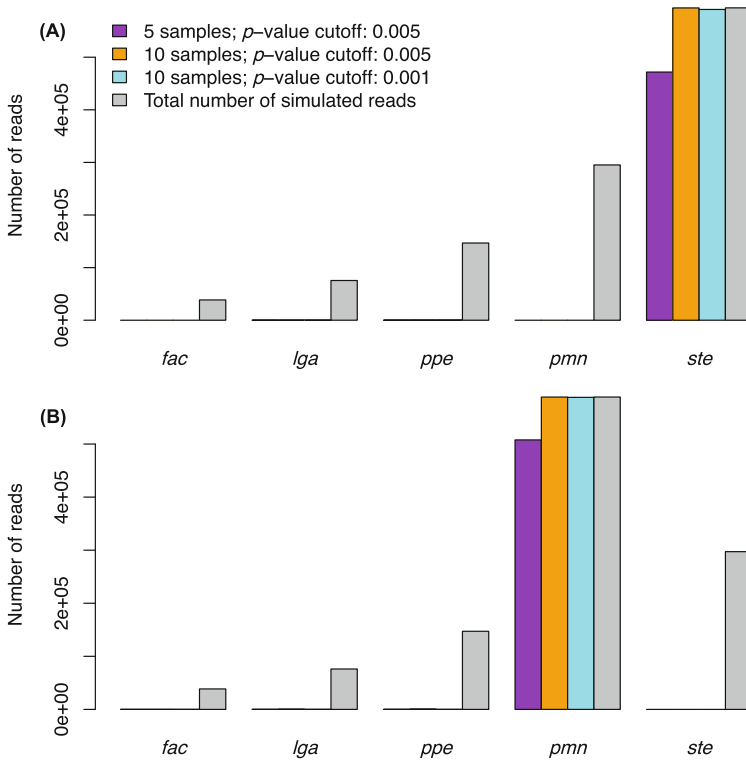


Fig. 1. CoSA effectively extracted reads from differential genomes. The upper and lower subfigures refer to read extraction for one of the samples of population 1 and 2, respectively. The x-axis shows the 5 different species; *fac*: *Ferroplasma acidarmanus* fer1, *lga*: *Lactobacillus gasser* ATCC 33323, *ppe*: *Pediococcus pentosaceus* ATCC 25745, *pmn*: *Prochlorococcus marinus* NATL2A, *ste*: *Streptococcus thermophilus* LMD-9. Bars of different colours (purple, yellow, cyan) indicate separate runs of CoSA using different parameters or different number of samples while the grey bars indicate simulated reads for each genome. The y-axis shows the number of reads extracted (or expected shown in grey bars).

only 5 samples for each population, CoSA could only extract 471,786 (79.44%) reads. Meanwhile, CoSA extracted very few reads from the non-differential genomes in both cases. Using a lower p-value cut-off of 0.001 (see Table 2 for the results) reduced the number of extracted reads from both differential and non-differential genomes. But CoSA still extracted most of the reads from the differential genomes. In conclusion, CoSA effectively extracted reads from differential genomes with a minor fold change whereas a minimal number of reads were extracted from non-differential genomes. We note that a very stringent p-value cut-off (e.g., 0.001) works well for this simulated case; however, for real microbiome datasets that have more complex population structure, a less stringent p-value cut-off might be needed for differential reads extraction (because of the sharing of k-mers among species) as shown in the application of CoSA to the T2D microbiomes (see below).

Table 2. Evaluation of CoSA using simulated datasets: community structure and reads extraction.

	Population		Reads extracted/simulated	
	P1 ^a	P2	P1	P2
<i>Ferroplasma acidarmanus</i> fer1	1 ^b	1	0/38,568 ^c	19/38,569
<i>Lactobacillus gasseri</i> ATCC 33323	2	2	122/75,528	77/76,152
<i>Pediococcus pentosaceus</i> ATCC 25745	4	4	178/146,787	25/147,199
<i>Prochlorococcus marinus</i> NATL2A	8	16	8/295,230	587,980/588,579
<i>Streptococcus thermophilus</i> LMD-9	16	8	590,820/593,858	0/297,227

^a: simulated population 1; ^b: relative abundance of the *F. acidarmanus* genome in population 1; ^c: 0 reads were extracted out of 38,568 reads from the *F. acidarmanus* genome in P1.

We further compared the assembly quality for the differential genomes with different number of samples, with the help of QUAST [12] and MUMer [21]. For the *S. thermophilus* LMD-9 genome in the same sample as above, we recovered 95.76% of the reference genome when 10 samples per population were used; but only 73.32% of the genome were assembled when we used 5 samples for each group (see Fig. 2 for the comparison). Not only we assembled a higher fraction of the genome for the differential genomes, but also we obtained fewer but longer contigs. We produced 84 contigs with N50 of 51,061 using 10 samples and 1,280 contigs with N50 of 1,180 using 5 samples. With more samples, CoSA is capable of better assembling the differential genomes. By contrast, our original SA approach relies on ratios of k-mers to detect differential reads and only a small fraction (19.64%) of the genome can be assembled using the reads it extracted.

3.2 Evaluation of CoSA Using the T2D Microbiomes

As shown in the above, CoSA was able to detect minor, but conserved differential genomes using the simulated datasets. Here we applied CoSA to the T2D microbiome cohort. As shown in Table 3, CoSA has resulted in a greater reduction of

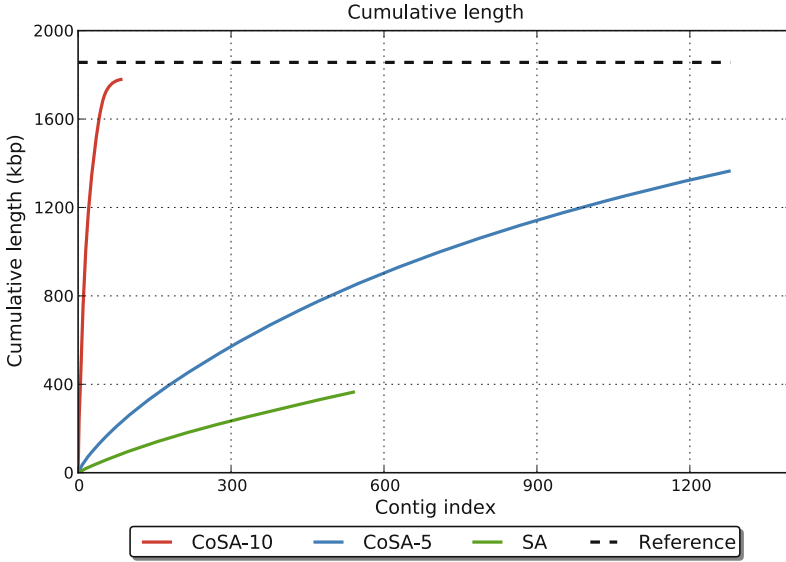


Fig. 2. Evaluation of the assembly quality of the differential genomes. The results indicate that CoSA outperforms SA for detecting minor but consistent effect when multiple samples are used, and that using more samples by CoSA results in better assembly of the differential genomes (CoSA-10, 10 samples were used; CoSA-5, 5 samples were used).

the sequencing data (retaining 8.99% of the total bases) than the original SA reads (which retained 17.59% of the original sequencing data). Extracted reads were then used for assembly and gene annotation. Although reads extraction by CoSA resulted in a smaller collection of microbial genes than the SA approach (since CoSA retained much fewer reads than SA), genes from CoSA tend to be more consistently differential across the samples between the groups. We pooled the genes derived from CoSA (1,008,068 genes) and SA (1,648,016 genes), resulting a collection of 2,656,084 genes, and further quantified the abundances of the genes in this collection. The gene abundance profile was then used for WRS test between the patient and the healthy control groups, with correcting for multiple testing using false discovery rate (q-value) computed by the tail area-based method of the R `fdrtool` package [41]. Table 3 summaries the test results, indicating that CoSA produced more significantly differential genes than SA. We note that none of the genes derived by SA had q-value less than 0.05. Sequences and annotations of the 357,591 genes assembled by CoSA (with $q\text{-value} \leq 0.05$) are available for download at the CoSA sourceforge project page.

3.3 Prediction of T2D Using Microbial Genes

It has been shown that metagenomes can be used for classification and prediction of diabetes status [17]. Karlssons and colleagues trained a Random Forest (RF)

Table 3. Summary of subtractive assembly results of the T2D datasets.

	CoSA*	SA
Total base pair in extracted reads	11.59 Gbp (8.99%)	22.68 Gbp (17.59%)
# of predicted genes ^a	1,008,068	1,648,016
# of significant genes (q-value ≤ 0.07)	563,743	285,666
# of significant genes (q-value ≤ 0.05)	357,591	0

*: p-value = 0.2 and voting threshold = 0.8 were used for reads extraction; ^a: only counted genes assembled from extracted reads from patients (but not healthy individuals).

model based on a training set of the NGT and T2D subjects using the profiles of species and MGCs (megagenomic gene clusters), and evaluated its performance using a tenfold cross-validation approach and calculated the predictive power as the area under the ROC curve (AUC). Their results showed that T2D was identified more accurately with MGCs (highest AUC = 0.83) than with microbial species (highest AUC = 0.71), suggesting that the functional composition of the microbiota determined by MGCs correlates better with diabetes than the species composition. We applied CoSA to T2D datasets (including datasets from patients and healthy individuals) using different settings of parameters and compared the performance of classifiers built from the assembled microbial genes (from both T2D patients and healthy-controls). Table 4 summarizes the results. We used two different classify algorithms, one is SVM with linear kernel and the other is RF whose forest includes 10 trees.

Using p-value of 0.05 and voting threshold of 0.3 (called *Normal* in Table 4) for reads extraction in CoSA followed by assembly and abundance quantification, we derived 296,979 genes. Our collection of genes resulted in a SVM that achieved a prediction accuracy of 0.94 (AUC), a significant improvement in the prediction accuracy as compared to the AUC reported in [17] (AUC = 0.83).

We also tested CoSA using more stringent parameters for reads extraction (p-value = 0.001 and voting threshold = 0.5). The reads extraction only resulted in a small reads file with 19.13 Mbp in total. Not surprisingly we were only able to assemble and predict 249 genes from this small collection of sequencing reads. Interestingly, a RF model (without using feature selection) built from this small set of microbial genes achieved an AUC of 0.79. This accuracy is worse than our best model (AUC = 0.94), and Karlsson’s RF model based on MGC (AUC = 0.83), but it is much better than Karlsson’s RF model based on bacterial species (AUC = 0.71). The advantage of using this setting (we called it *Strict*) is that only a small number of reads were extracted and only a small number of genes need to be quantified and used for building classifiers, and it still achieves reasonable prediction accuracy.

On the other hand, a much larger collection of microbial genes may make feature selection a more serious problem for building predictors, and therefore compromise the accuracy of predictors trained using these microbial genes. For example, we applied CoSA using a looser setting (p-value = 0.2 and voting-

threshold = 0.8; called *Loose* in Table 4), which resulted in the extraction of many more reads. Not surprisingly, many more genes can be assembled. However, more genes to start with doesn't necessarily result in a better classifier for prediction. The best classifier built using this larger collection of genes achieved an AUC of only 0.89. Similarly, using our original subtractive assembly approach (SA), an even greater collection of microbial genes can be assembled. However, the best predictor built using this larger collection of genes only achieved an AUC of 0.85.

Sequences and annotations (by myRAST [30] and hmmscan [8]) of the 207 differential genes that resulted in the highest prediction accuracy (AUC = 0.94) are available for download at the CoSA sourceforge project website. Some of the functions and associated pathways are consistent with what we observed based on SA [42], including murein hydrolases (protein ID: *k87_534_1_134_+*) and multidrug resistance efflux pumps (protein ID: *k87_34893_1_275_-*).

Table 4. Comparison of the accuracy of T2D prediction using microbial genes derived by CoSA and SA.

		CoSA			SA
		<i>Strict</i>	<i>Normal</i>	<i>Loose</i>	
Reads extraction	P-value cut-off	0.001	0.05	0.2	- ^a
	Voting threshold	0.5	0.3	0.8	0.5
	Total base pair	19.13 Mbp	6.08 Gbp	19.23 Gbp	36.26 Gbp
Classification	# of genes	249	296,979	1,741,472	2,098,590
	# of genes selected ^b	249 ^c	207	230	210
	Classifier	RF	SVM	SVM	SVM
	AUC ^d	0.79	0.94	0.89	0.85

^a: SA uses ratios of k-mer counts to determine differential k-mers; ^b: genes were selected using L1-based feature selection method; ^c: no feature selection was applied for this case;

^d: average accuracy using 10-fold cross-validation.

4 Discussion

We developed a pipeline based on CoSA, which efficiently extracts reads that are likely sequenced from differential genes across samples for the identification of conserved microbial marker genes. Considering the heterogeneity nature of the microbiomes across human subjects, it is important to have a method that can detect disease-associated features that are consistent across samples. Tests of our approach using both simulated and real microbiomes show the importance of using multiple samples for such purposes.

The time and space complexity of CoSA is related to the number of datasets and the size of each dataset. The running time and memory cost is small for small datasets such as the simulated microbiome datasets. However, the computational time and memory usage can be substantial for large cohorts of datasets such as

the T2D datasets. The total running time of CoSA for the simulated datasets was 44 min (38 min for k-mer counting and 6 min for the detection of differential k-mers and therefore differential reads), and the peak memory usage was 2G. However, for the large T2D cohort, the running time for k-mer counting was 6.9 h and the next step of detecting differential k-mers and reads took 27.5 h. The peak memory usage for the T2D datasets was also substantial, which was 229 Gb. Considering the increasing capacity of sequencing technologies, we will further investigate other strategies to reduce the memory usage and running time of CoSA.

In the current implementation of CoSA, WRS test is applied to k-mer counts normalized by the total k-mers (which is equivalent to the total reads) in each sample, for the detection of k-mers with differential abundances across healthy-controls and patients. This choice is mostly driven by the practical convenience. Our results showed that this simple strategy of normalization worked well in practice. However, it has been shown that such a normalization approach may have limitations for applications in detecting metagenome-wise marker-gene surveys [31]. We will explore the possibility of using other normalization techniques such as the cumulative-sum scaling approach in CoSA.

Acknowledgement. This work was supported by the NIH grant 1R01AI108888 to Ye.

References

1. Albertsen, M., Hugenholtz, P., Skarshewski, A., Nielsen, K.L., Tyson, G.W., Nielsen, P.H.: Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nat. Biotechnol.* **31**(6), 533–538 (2013)
2. Alneberg, J., Bjarnason, B.S., de Bruijn, I., Schirmer, M., Quick, J., Ijaz, U.Z., Lahti, L., Loman, N.J., Andersson, A.F., Quince, C.: Binning metagenomic contigs by coverage and composition. *Nat. Methods* **11**(11), 1144–1146 (2014)
3. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.A., Dvorkin, M., Kulikov, A.S., Lesin, V.M., Nikolenko, S.I., Pham, S., Prjibelski, A.D., Pyshkin, A.V., Sirotkin, A.V., Vyahhi, N., Tesler, G., Alekseyev, M.A., Pevzner, P.A.: SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.* **19**(5), 455–477 (2012)
4. Ben-Hur, A., Ong, C.S., Sonnenburg, S., Scholkopf, B., Ratsch, G.: Support vector machines and kernels for computational biology. *PLoS Comput. Biol.* **4**(10), e1000173 (2008)
5. Cho, I., Blaser, M.J.: The human microbiome: at the interface of health and disease. *Nat. Rev. Genet.* **13**(4), 260–270 (2012)
6. de Martel, C., Ferlay, J., Franceschi, S., Vignat, J., Bray, F., Forman, D., Plummer, M.: Global burden of cancers attributable to infections in 2008: a review and synthetic analysis. *Lancet Oncol.* **13**(6), 607–615 (2012)
7. Deorowicz, S., Kokot, M., Grabowski, S., Debudaj-Grabysz, A.: KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics* **31**(10), 1569–1576 (2015)
8. Finn, R.D., Clements, J., Eddy, S.R.: HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.* **39**(Web Server issue), 29–37 (2011)

9. Garrett, W.S.: Cancer and the microbiota. *Science* **348**(6230), 80–86 (2015)
10. Ge, X., Rodriguez, R., Trinh, M., Gunsolley, J., Xu, P.: Oral microbiome of deep and shallow dental pockets in chronic periodontitis. *PLoS One* **8**(6), e65520 (2013)
11. Gilbert, J.A., Quinn, R.A., Debelius, J., Xu, Z.Z., Morton, J., Garg, N., Jansson, J.K., Dorrestein, P.C., Knight, R.: Microbiome-wide association studies link dynamic microbial consortia to disease. *Nature* **535**(7610), 94–103 (2016)
12. Gurevich, A., Saveliev, V., Vyahhi, N., Tesler, G.: QUASt: quality assessment tool for genome assemblies. *Bioinformatics* **29**(8), 1072–1075 (2013)
13. Iverson, V., Morris, R.M., Frazar, C.D., Berthiaume, C.T., Morales, R.L., Armbrust, E.V.: Untangling genomes from metagenomes: revealing an uncultured class of marine Euryarchaeota. *Science* **335**(6068), 587–590 (2012)
14. Jiang, B., Song, K., Ren, J., Deng, M., Sun, F., Zhang, X.: Comparison of metagenomic samples using sequence signatures. *BMC Genomics* **13**, 730 (2012)
15. Jorth, P., Turner, K.H., Gumus, P., Nizam, N., Buduneli, N., Whiteley, M.: Metatranscriptomics of the human oral microbiome during health and disease. *MBio* **5**(2), e01012–e01014 (2014)
16. Kang, D.W., Park, J.G., Ilhan, Z.E., Wallstrom, G., Labaer, J., Adams, J.B., Krajmalnik-Brown, R.: Reduced incidence of *Prevotella* and other fermenters in intestinal microflora of autistic children. *PLoS One* **8**(7), e68322 (2013)
17. Karlsson, F.H., Tremaroli, V., Nookaew, I., Bergstrom, G., Behre, C.J., Fagerberg, B., Nielsen, J., Backhed, F.: Gut metagenome in European women with normal, impaired and diabetic glucose control. *Nature* **498**(7452), 99–103 (2013)
18. Knights, D., Costello, E.K., Knight, R.: Supervised classification of human microbiota. *FEMS Microbiol. Rev.* **35**(2), 343–359 (2011)
19. Koeth, R.A., Wang, Z., Levison, B.S., Buffa, J.A., Org, E., Sheehy, B.T., Britt, E.B., Fu, X., Wu, Y., Li, L., Smith, J.D., DiDonato, J.A., Chen, J., Li, H., Wu, G.D., Lewis, J.D., Warrier, M., Brown, J.M., Krauss, R.M., Tang, W.H., Bushman, F.D., Lusis, A.J., Hazen, S.L.: Intestinal microbiota metabolism of L-carnitine, a nutrient in red meat, promotes atherosclerosis. *Nat. Med.* **19**(5), 576–585 (2013)
20. Kostic, A.D., Howitt, M.R., Garrett, W.S.: Exploring host-microbiota interactions in animal models and humans. *Genes Dev.* **27**(7), 701–718 (2013)
21. Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.L.: Versatile and open software for comparing large genomes. *Genome Biol.* **5**(2), R12 (2004)
22. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**(4), 357–359 (2012)
23. Lewis, J.D., Chen, E.Z., Baldassano, R.N., Otley, A.R., Griffiths, A.M., Lee, D., Bittinger, K., Bailey, A., Friedman, E.S., Hoffmann, C., Albenberg, L., Sinha, R., Compher, C., Gilroy, E., Nessel, L., Grant, A., Chehoud, C., Li, H., Wu, G.D., Bushman, F.D.: Inflammation, antibiotics, and diet as environmental stressors of the gut microbiome in pediatric Crohn’s disease. *Cell Host Microbe* **18**(4), 489–500 (2015)
24. Li, D., Luo, R., Liu, C.M., Leung, C.M., Ting, H.F., Sadakane, K., Yamashita, H., Lam, T.W.: Megahit v1.0: a fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods* **102**, 3–11 (2016)
25. Li, X., Andersen, D.G., Kaminsky, M., Freedman, M.J.: Algorithmic improvements for fast concurrent cuckoo hashing. In: Proceedings of the 9th ACM European Conference on Computer Systems (EuroSys), April 2014
26. Marcais, G., Kingsford, C.: A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* **27**(6), 764–770 (2011)

27. Mavromatis, K., Ivanova, N., Barry, K., Shapiro, H., Goltsman, E., McHardy, A.C., Rigoutsos, I., Salamov, A., Korzeniewski, F., Land, M., Lapidus, A., Grigoriev, I., Richardson, P., Hugenholtz, P., Kyrpides, N.C.: Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nat. Methods* **4**(6), 495–500 (2007)
28. Melsted, P., Pritchard, J.K.: Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinform.* **12**, 333 (2011)
29. Nielsen, H.B., Almeida, M., Juncker, A.S., Rasmussen, S., Li, J., Sunagawa, S., Plichta, D.R., Gautier, L., Pedersen, A.G., Le Chatelier, E., et al.: Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nat. Biotechnol.* **32**(8), 822–828 (2014)
30. Overbeek, R., Olson, R., Pusch, G.D., Olsen, G.J., Davis, J.J., Disz, T., Edwards, R.A., Gerdes, S., Parrello, B., Shukla, M., Vonstein, V., Wattam, A.R., Xia, F., Stevens, R.: The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST). *Nucleic Acids Res.* **42**(Database issue), D206–D214 (2014)
31. Paulson, J.N., Stine, O.C., Bravo, H.C., Pop, M.: Differential abundance analysis for microbial marker-gene surveys. *Nat. Methods* **10**(12), 1200–1202 (2013)
32. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
33. Peng, Y., Leung, H.C., Yiu, S.M., Chin, F.Y.: IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* **28**(11), 1420–1428 (2012)
34. Qin, N., Yang, F., Li, A., Prifti, E., Chen, Y., Shao, L., Guo, J., Le Chatelier, E., Yao, J., Wu, L., Zhou, J., Ni, S., Liu, L., Pons, N., Batto, J.M., Kennedy, S.P., Leonard, P., Yuan, C., Ding, W., Chen, Y., Hu, X., Zheng, B., Qian, G., Xu, W., Ehrlich, S.D., Zheng, S., Li, L.: Alterations of the human gut microbiome in liver cirrhosis. *Nature* **513**(7516), 59–64 (2014)
35. Rho, M., Tang, H., Ye, Y.: FragGeneScan: predicting genes in short and error-prone reads. *Nucleic Acids Res.* **38**(20), e191 (2010)
36. Richter, D.C., Ott, F., Auch, A.F., Schmid, R., Huson, D.H.: MetaSim: a sequencing simulator for genomics and metagenomics. *PLoS One* **3**(10), e3373 (2008)
37. Scheperjans, F., Aho, V., Pereira, P.A., Koskinen, K., Paulin, L., Pekkonen, E., Haapaniemi, E., Kaakola, S., Eerola-Rautio, J., Pohja, M., Kinnunen, E., Murros, K., Auvinen, P.: Gut microbiota are related to Parkinson’s disease and clinical phenotype. *Mov. Disord.* **30**(3), 350–358 (2015)
38. Scher, J.U., Sczesnak, A., Longman, R.S., Segata, N., Ubeda, C., Bielski, C., Rosstron, T., Cerundolo, V., Pamer, E.G., Abramson, S.B., Huttenhower, C., Littman, D.R.: Expansion of intestinal *Prevotella copri* correlates with enhanced susceptibility to arthritis. *Elife* **2**, e01202 (2013)
39. Sears, C.L., Garrett, W.S.: Microbes, microbiota, and colon cancer. *Cell Host Microbe* **15**(3), 317–328 (2014)
40. Sender, R., Fuchs, S., Milo, R.: Revised estimates for the number of human and bacteria cells in the body. *PLoS Biol.* **14**(8), e1002533 (2016)
41. Strimmer, K.: fdrtool: a versatile R package for estimating local and tail area-based false discovery rates. *Bioinformatics* **24**(12), 1461–1462 (2008)
42. Wang, M., Doak, T.G., Ye, Y.: Subtractive assembly for comparative metagenomics, and its application to type 2 diabetes metagenomes. *Genome Biol.* **16**, 243 (2015)

43. Wu, Y.W., Simmons, B.A., Singer, S.W.: MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics* **32**(4), 605–607 (2016)
44. Wu, Y.W., Ye, Y.: A novel abundance-based algorithm for binning metagenomic sequences using l-tuples. *J. Comput. Biol.* **18**(3), 523–534 (2011)
45. Zeller, G., Tap, J., Voigt, A.Y., Sunagawa, S., Kultima, J.R., Costea, P.I., Amiot, A., Bohm, J., Brunetti, F., Habermann, N., Hercog, R., Koch, M., Luciani, A., Mende, D.R., Schneider, M.A., Schrotz-King, P., Tournigand, C., Tran Van Nhieu, J., Yamada, T., Zimmermann, J., Benes, V., Kloor, M., Ulrich, C.M., von Knebel Doeberitz, M., Sobhani, I., Bork, P.: Potential of fecal microbiota for early-stage detection of colorectal cancer. *Mol. Syst. Biol.* **10**, 766 (2014)
46. Zhang, Q., Pell, J., Canino-Koning, R., Howe, A.C., Brown, C.T.: These are not the k-mers you are looking for: efficient online k-mer counting using a probabilistic data structure. *PLoS One* **9**(7), e101271 (2014)
47. Zhu, B., Wang, X., Li, L.: Human gut microbiome: the second genome of human body. *Protein Cell* **1**(8), 718–725 (2010)

A Flow Procedure for the Linearization of Genome Sequence Graphs

David Haussler¹, Maciej Smuga-Otto¹, Benedict Paten¹(✉),
Adam M. Novak¹, Sergei Nikitin², Maria Zueva²,
and Dmitrii Miagkov²

¹ UC Santa Cruz Genomics Institute, University of California,
Santa Cruz, CA, USA

benedict@soe.ucsc.edu

² Life Sciences Business Unit, EPAM Systems, Inc., Newtown, PA, USA

dmitrii.miagkov@epam.com

Abstract. Efforts to incorporate human genetic variation into the reference human genome have converged on the idea of a graph representation of genetic variation within a species, a genome sequence graph. A sequence graph represents a set of individual haploid reference genomes as paths in a single graph. When that set of reference genomes is sufficiently diverse, the sequence graph implicitly contains all frequent human genetic variations, including translocations, inversions, deletions, and insertions.

In representing a set of genomes as a sequence graph one encounters certain challenges. One of the most important is the problem of graph linearization, essential both for efficiency of storage and access, as well as for natural graph visualization and compatibility with other tools. The goal of graph linearization is to order nodes of the graph in such a way that operations such as access, traversal and visualization are as efficient and effective as possible.

A new algorithm for the linearization of sequence graphs, called the flow procedure, is proposed in this paper. Comparative experimental evaluation of the flow procedure against other algorithms shows that it outperforms its rivals in the metrics most relevant to sequence graphs.

Keywords: Sequence graph · Linearization · Flow procedure · Feedback arcs · Cut width · Backbone · Grooming

1 Motivation

The current human reference genome consists essentially of a single representative of each of the human chromosomes. In essence, an arbitrary person's genome is chosen to represent all of humanity. This leads to loss of information and bias. Efforts to incorporate human genetic variation into the reference human genome have converged on the idea of a graph representation of genetic variation within a species, a genome sequence graph [1].

In its mathematically most simple form, each node of a sequence graph contains a single DNA base that occurs at an orthologous locus in one or more of the haploid

genomes represented in the graph. Each arc represents an adjacency (chemically, a covalent bond) that occurs between consecutive instances of bases in those genomes. Excluding reversing joins (see below) each arc is directed according to the default strand direction of the DNA sequence used to build the graph, connecting the 3' side of the previous base (the tail of the edge) with the 5' side of the next base (the head of the edge). At points where the individual genomes differ to the right (i.e., 3', downstream) of an orthologous base, the node representing that base will have two or more outgoing arcs. For example, the graph in Fig. 2 was built from the DNA strands *CATCGCT*, *CATCGT*, *CAGCGAT* and *CATCGAGAGAGCT* aligned at orthologous bases as shown in Fig. 1.

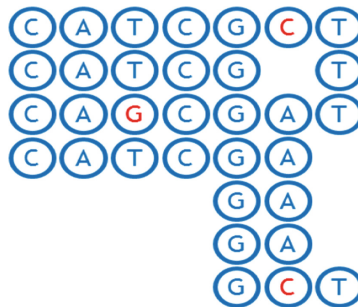


Fig. 1. Alignment of *CATCGCT*, *CAGCGAT*, *CATCGAGAGAGCT* DNA strands.

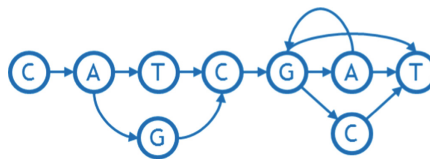


Fig. 2. An example of a sequence graph reflecting SNP (T/G), tandem duplication GA->GAGAGA, and deletion/SNP (A/C/-).

Representing a genome as a graph requires building a number of tools to work with it efficiently. In particular, one needs to linearize the graph, that is order the nodes from left to right in a straight line. Linearization facilitates visual perception of a graph, allows software to index the nodes in a familiar manner, and imposes natural order of bases useful in storage, search and analysis, for example enabling traversal from left to right with minimum feedback runs. Figures 3 and 4 show examples of a linearized graph and individual genomes in it. Here we have also added a new feature to the graphs in the form of **arc weights**. An arc weight is used to signify the importance of an arc in typical applications running on the graph. Normally we set the arc weight to the number of times that the arc is traversed in the reference genomes used to build the graph, under the assumption that arcs used frequently in the reference genomes will

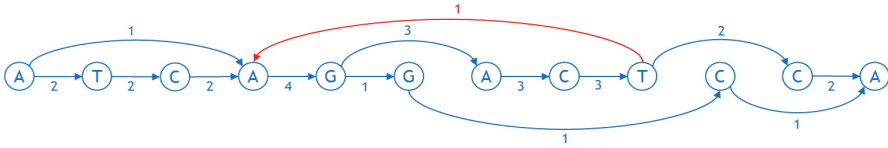


Fig. 3. An example of a linearized sequence graph with weighted arcs. The arc in red is directed from the right to the left and is called feedback arc.

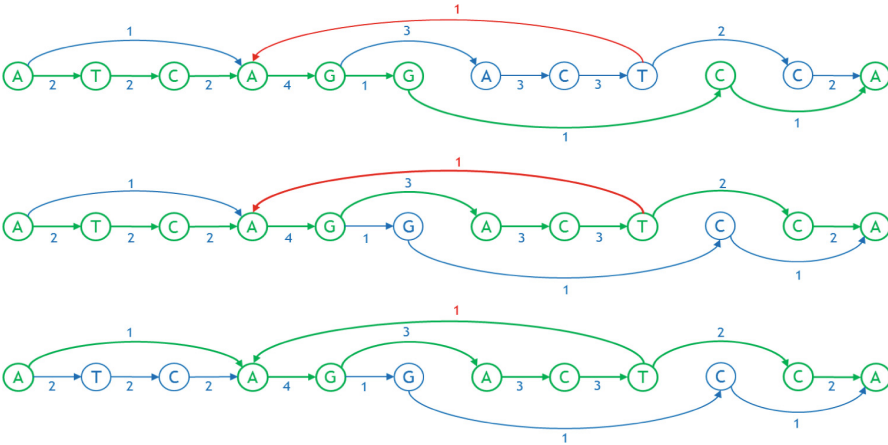


Fig. 4. Three individual genomes as paths in the linearized graph, shown in green. The first and second genomes are ATCAGGCA and ATCAGACTCA, respectively. The third one is AAGACTAGACTCA where the arc between T and A is a feedback arc.

also be used frequently in the applications of the sequence graph built from them. Some reference genomes may be weighted more than others.

2 Problem Statement

A linearization of a sequence graph aims to make the total weight of all feedback arcs, called the **weighted feedback** (see Fig. 3), small, along with the “width” (number of arcs) crossing any vertical line in the layout (called a “cut”, see Fig. 5). Unnecessary

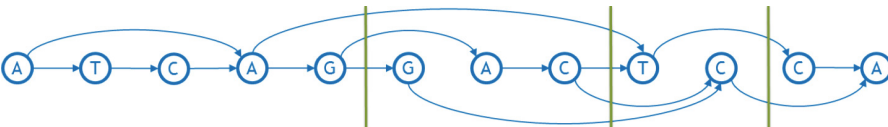


Fig. 5. Examples of cuts and cut widths. Three cuts are shown with green vertical lines. Their widths are, from left to right, 3, 4, and 2 respectively.

feedback arcs make many types of genetic analysis more inefficient, as these typically proceed left-to-right on a conventional reference genome. An arc crossing a cut is considered to be part of an allele spanning that cut (see [2]), so a graph with smaller cut width at a cut has fewer alleles at that cut. The mean of the cut width over all cuts in the graph is called the **average cut width**.

In light of their importance for genetic analysis, we will evaluate a linearization based on its average cut width and either the weighted feedback or the number of feedback arcs it contains.

Unfortunately, the problems of minimizing weighted feedback, and of minimizing the average cut width, are each separately difficult. The simpler problem of minimizing the number of feedback arcs is known in literature as the *feedback arc set problem*, or FAS for short [3]. This is an NP-hard problem [4], but there are a number of various heuristic approaches to approximating a solution to it [5]. The problem of minimizing the average cut width [6] is also known to be NP-hard. A good heuristic [7] is necessary for good results. In our case, starting our procedure with the “primary path” taken by the reference genome is a natural choice. This is the first time to our knowledge that a heuristic algorithm to minimize both metrics at the same time has been proposed.

3 Algorithm Description

We propose here a simple heuristic divide-and-conquer approach to linearly order the bases of a graph that tries to achieve either small weighted feedback (or small number of feedback arcs) and small average cut width. The key algorithmic tool is max-flow/min-cut in a directed graph [8], so we call it the **flow procedure**. Prior to applying the flow procedure, the sequence graph is “groomed” as described at the end of this section.

3.1 First Step: Determine the Backbone

Following grooming, the flow procedure starts with a connected graph with directed arcs and a designated linear ordering of a subset of the bases called the **backbone**. Arcs leading from the backbone to nodes not on the backbone are called **out-arcs**, and arcs directed into the backbone from nodes not on the backbone are called **in-arcs**. Grooming guarantees such the first base of the backbone has no in-arcs and the last has no out-arcs. Extra dummy bases are added at either end if necessary. Normally the initial backbone is a biologically determined primary path of the graph, e.g. from a selected haploid reference genome in the set of genomes used to build the graph, perhaps the existing haploid reference human genome. The flow procedure creates a backbone using its internal heuristics if none is given *a priori*. In linearizing the sequence graph by creating a total ordering of the bases, the relative order of the bases in backbone will not change. The rest of the bases in the graph will be inserted either between bases of the backbone, before the backbone, or after the backbone. Thus, any feedback arcs already in the backbone ordering, called **initial feedback arcs**, will remain as feedback arcs in the final ordering.

Consider the graph depicted in Fig. 6. The backbone is CGATC horizontally across the middle (highlighted by dark blue color). The three out-arcs of the backbone are shown with thick green arrows. The two in-arcs are shown with thick purple arrows. The weights are assumed to reflect usage, but we also assume the usage statistics may be partial. Hence the weight coming into a base does not always match the weight coming out.

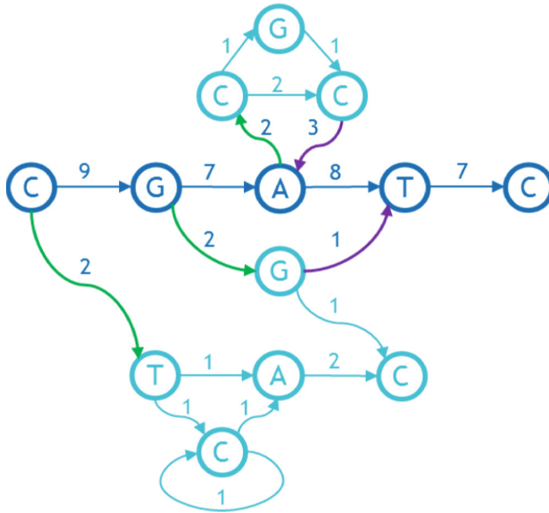


Fig. 6. An initial directed graph with weights on the arcs.

3.2 Second Step: Add Source and Sink

We set up a max flow/min cut network as follows. The nodes and arcs of the network are nodes and arcs of the graph. The capacity of the arc is its weight. In addition, there is a special source node and a special sink node. The network arcs for these are defined as follows: we let N be the maximum of the sum of the weights of the outgoing arcs for any node in the graph and we add an arc of capacity $N + 1$ from the source node to each base on the backbone that has an out-arc. Then each in-arc on the backbone is redirected to the sink node with no change in capacity.

3.3 Third Step: Determine the Minimum Cut and Delete It

The maximum flow in our example is easily seen to be 3, and it maximizes the capacity of the two arcs that are cut by the green bars (Fig. 7). Therefore, these form a minimum cut. Since the capacities of the arcs connecting the source to the backbone are too high to be achieved by any flow, none of these are in the cut. Therefore, the cut must split the flow network into an **out-component** containing the source and its outgoing arcs, and an **in-component** containing the sink. In Fig. 7 the in-component consists of the uppermost nodes C, G, C and the sink, and the out-component is the remainder.

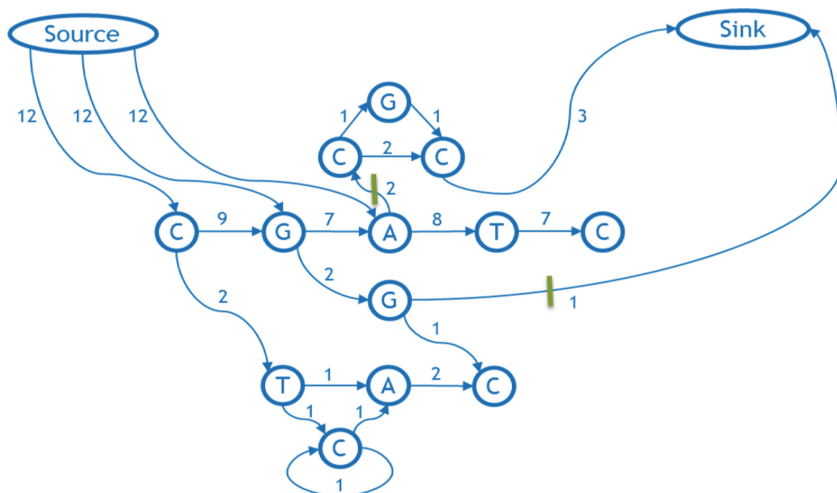


Fig. 7. The flow network corresponding to the above weighted graph and its designated backbone.

We remove the cut arcs from the graph. Essentially, in doing this we decide to give up worrying about these arcs and try to minimize the weighted feedback and average cut width as if they were not there. Since capacity equals weight, by choosing a minimum capacity cut, we ignore the arcs that cost us the least in weight.

Excluding any initial feedback arcs on the backbone itself, we classify all bases not on the backbone into a sequence of **out-growths** and **in-growths** as follows. Starting at the last base on the backbone that has an out-arc, we define its out-growth to be all bases reachable by a forward directed path from that base. Then we move backwards along the backbone, defining out-growths consisting of all the bases reachable by a forward directed path from the base on the backbone with an out-arc that were not already included in previously defined out-growths. We define in-growths in a similar fashion, moving forward from the start of the backbone and using backward-directed paths. Figure 8 shows in-growth and out-growths for the example under consideration. There are two outgrowths, the first from the node G on the backbone (contains nodes G, G, C), and the second from its predecessor, the first node on the backbone (labeled “C”, contains nodes C, T, C, A). The dotted green arc is not a proper part of either outgrowth; it is discovered when exploring the outgrowth from C, and found to lead into the previous outgrowth found from G. The one in-growth enters the node A of backbone and contains nodes C, G, C, A, shown in purple.

3.4 Fourth Step: Repeat Procedure for In- and Out-Growths

Finally, we apply the entire procedure recursively to each out-growth and in-growth, using a heuristic that uses the backbone’s base as the first base for an out-growth or as the last base for an in-growth, respectively. When the recursive call completes, the bases from it are inserted into the backbone of the calling procedure in the specified

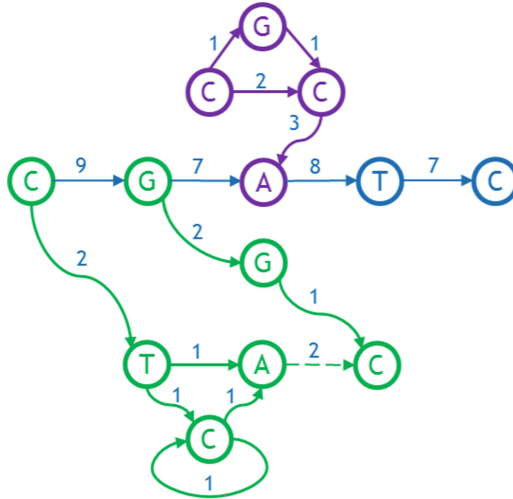


Fig. 8. Out-growths (green) and in-growth (purple) of the graph.

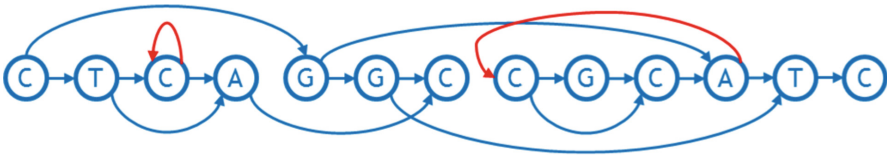


Fig. 9. The final sorted graph with the bases totally ordered.

order immediately following an out-growth or immediately preceding an in-growth, respectively. The final ordering for the example above is shown in Fig. 9.

Normally the out-growths and in-growths together comprise the whole. However, if not, the entire procedure can just be repeated, each time using the linear order established from the previous cycle as a new backbone. Grooming a connected graph (see below) assures that these repetitions will eventually reach every node in the graph.

It remains to specify a heuristic for determining the backbone when it is not explicitly given. When the first base of the backbone is given, we extend it into a path using a greedy algorithm: in each step, we add to the existing path the base with the highest forward directed arc weight, breaking ties arbitrarily, and we do so until no more bases can be added. A complementary procedure is run in reverse if we are instead given the last base of the backbone.

3.5 Grooming

Finally, we explain the preprocessing step of grooming the graph. The base in each node in a sequence graph has two sides (3' and 5'). The directed arcs we have been using are edges of the sequence graph that connect the 3' side of one node to the 5' side

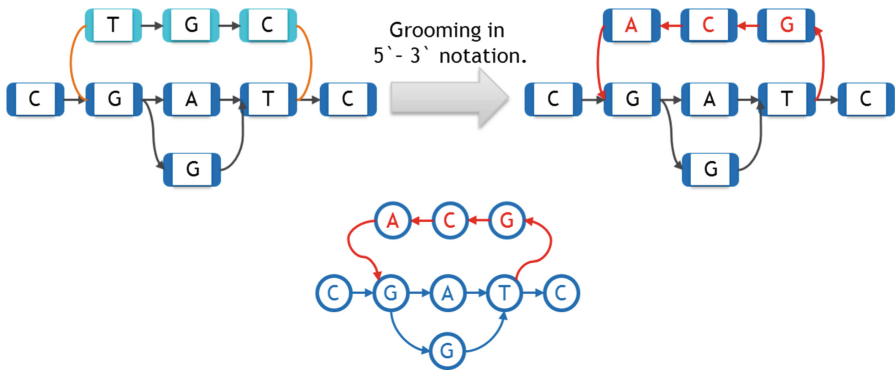


Fig. 10. Grooming procedure. Reversing joins are shown in brown on the upper left. Bottom one shows the groom on the upper right as directed graph.

of another (possibly the same) node. The arcs are directed in the 3' to 5' side direction. There are also additional edges in a sequence graph that here we will refer to as **reversing joins**, which we have not discussed up until this point (see [1 and 9] for an introduction). A reversing join is an undirected edge connecting the 3' sides of two nodes, or connecting the 5' sides of two nodes. As a preprocessing step to the flow procedure, and all other heuristic algorithms we examine for linearization of a directed graph, we first eliminate as many of the reversing joins as possible by replacing the graph with an equivalent graph that has fewer reversing joins. Then if there are still reversing joins left we just ignore them. This way we are always working with graphs that only contain directed arcs. The process we use to minimize the number of reversing joins is called **grooming** (Fig. 10).

Grooming works as follows. A given connected component (a set of nodes such that one can travel between any two nodes in it along the standard arcs, in both direction, and reversing joins) may fall apart if the reversing joins are removed. This indicates that some of reversing joins were unnecessary. Let one connected component be called the **primary component** (shown in dark blue in Fig. 10), and the others be called the **secondary components** (shown in light blue in Fig. 10). We obtain an isomorphic graph that will have fewer components after removal of reversing joins by simply **reverse-complementing** the secondary component, i.e., reverse-complementing every base in it and inverting the direction of the arcs between these secondary bases. The 3' reversing joins connecting this secondary component to the main graph are replaced by directed arcs pointing into the secondary component, and the 5' reversing joins are replaced by directed arcs back to the primary component. This has the effect of changing every 3' side in the secondary component into a 5' side, and vice versa. On the right side of Fig. 10 nodes and arcs which were changed during grooming are shown in red. By repeating this procedure on any connected sequence graph we eventually reach an isomorphic graph that has just one connected component even after removing the reversing joins.

Since we will not be reducing the number of reversing joins further, and being undirected they cannot be considered feedback arcs, from here on, we will stop paying attention to the reversing joins and consider only graphs that are fully directed.

4 Complexity Estimation

The max flow/min cut sorting algorithm described above can be broken into four steps:

1. Find the backbone (if it is not given)
2. Create the flow graph by adding the source and sink and connecting them to the graph
3. Find the maximum flow and minimal cut and delete the minimal cut, using the Ford-Fulkerson algorithm
4. Find the in-growth and out-growth and repeat steps 1 through 4 for them

Let us consider the complexity of each step separately.

In the preparation step, we perform a greedy depth-first search to find the backbone if it is not given (in practice we only find the backbone on recursive calls, as the whole graph's backbone is given). We do not visit any arc more than once, so the time complexity is $O(|A|)$, where A is the number of arcs.

In creating the flow, we add 2 nodes to the graph (the source and sink) and draw several arcs from the source to those nodes in the backbone that have an outgoing arc, and also reroute the arcs going into the backbone to the sink. We do not examine any arc more than once, so the time complexity is $O(|A|)$.

The Ford-Fulkerson algorithm works in $O(|A| * |\text{max-flow}|)$ [10]. In the worst case, $|\text{max-flow}| \sim O(|A|)$, in which case the time complexity becomes $O(|A|^2)$.

Every recursive call decreases the number of nodes in the graph, so the number of recursive calls is $O(|V|)$.

These estimations are shown in the Table 1.

Table 1. Complexity estimation of the flow procedure algorithm.

Step	Complexity
1. Find the backbone	$O(A)$
2. Add source and sink, draw the arcs with required weights	$ V + \text{out- and in-arcs} = O(A)$ assuming at least one arc per node
3. Determine the maximum flow and remove the arcs from the minimum cut	$O(A * \text{max flow})$
4. Repeat steps 1 through 4	recursion depth $\leq V $

The final complexity estimate is thus $O(|A| * |\text{max-flow}| * |\text{recursion depth}|)$.

The best-case complexity (if the max flow is constant and there are a constant number of recursive calls) is $O(|A|)$, again, assuming at least one arc per node. In the worst case, it is $O(|A|^2 * |V|)$ as the max flow may be proportional to $|A|$ and the recursion depth proportional to $|V|$.

The Ford-Fulkerson algorithm and the recursion contribute the most to the final algorithm’s complexity. Depth of recursion is not a problem in practice. However, the maximum flow will often be approximately $O(|A|)$ due to how the flow is constructed: each variation increases it by creating a new path from the source to the sink. Because the flow becomes so large, the Ford-Fulkerson algorithm will work in quadratic time ($O(|A|^2)$). Improvements to the algorithm that reduce the typical max flow so that it is polylogarithmic in $|A|$ would improve its speed.

5 Experimental Evaluation

5.1 Data Modeling

The flow procedure was tested on data that was artificially generated by taking a 37 kilobase piece of the GRCh38 assembly and adding artificial structural variations to it using the RSVSim [11] from Bioconductor. This package lets one simulate any given set of structural variations to a reference, producing a modified FASTA file. The positions of the variations were distributed uniformly, while their lengths were fixed. After fixing a specified set of variations, a series of FASTA files were created and passed on to a vg [12] tool, which generated the graph using a multiple sequence graph alignment algorithm.

Four types of structural variation were simulated: insertions, deletions, duplications, and inversions. Tandem duplications were limited to one copy.

Table 2. The lengths of the variations in the testing data.

	Deletion	Insertion	Inversion	Duplication
Length	20	20	200	500

Two different test data sets, each consisting of a series of graphs, were generated in this way. The first was created to investigate the effect of the overall amount of variation on the number of feedback arcs and the cut width achievable by each algorithm. This data set—consisted of graphs each having equal numbers of all four kinds of variations (i.e., there were as many insertions as there were deletions, inversions and duplications), with only the total number of variations changing between graphs. The variations’ sizes are given in Table 2. More details on data modelling could be found in the Appendix.

The second set of graphs was created to investigate the relationship between the relative frequencies of each type of variation and the number of feedback arcs and the cut width achievable by each algorithm ([13], Tables 1–4).

The third data set was created in order to test algorithm’s time performance. The graphs in this data set were created from scratch and have structure similar to those graphs above with doubling of the number of nodes from one to another.

5.2 Results and Discussion

In order to comparatively analyze the quality and speed of our algorithm, we took Kahn’s well-known topological sorting algorithm [14], as well as Eades’ [15] modified version thereof that guarantees a low number of feedback arcs while still working in linear time. We were not able to find competitor for cut width minimization problem to measure against, because all the algorithms we have found are focused on exact solution of the problem, thus having high complexity (cubic and higher) and thus working only with graphs of 200 nodes or less. Note that neither Kahn’s nor Eades’ algorithm uses the backbone as a heuristic. All three algorithms were tested on the same data (see [13] and Sect. 5.1 for details on the modeling). Their outputs were compared on two metrics – the number of feedback arcs and the average cut width. Kahn’s algorithm, Eades’ algorithm, and flow procedure are all implemented in the vg tool [12].

Figures 11 and 12 depict the main quantitative outputs of the three algorithms, Kahn, Eades, and the flow procedure (FP), for the first set of testing data. The same results for the second set are given in [13]. From Fig. 11 it is fairly obvious that in terms of the number of feedback arcs, the FP algorithm vastly outperforms Kahn’s, doing only slightly worse than Eades’ algorithm. The difference between FP and Eades is much smaller than between FP and Kahn. It is clear from Fig. 12 that FP’s average cut width is an order of magnitude lower than that of Eades [15] or Kahn [14].

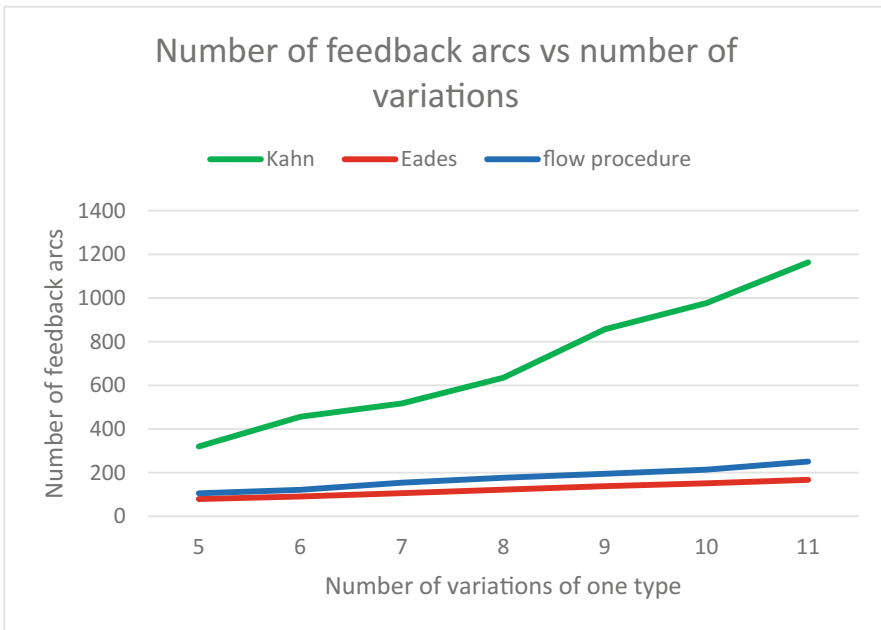


Fig. 11. The relationship between the number of feedback arcs and variations.

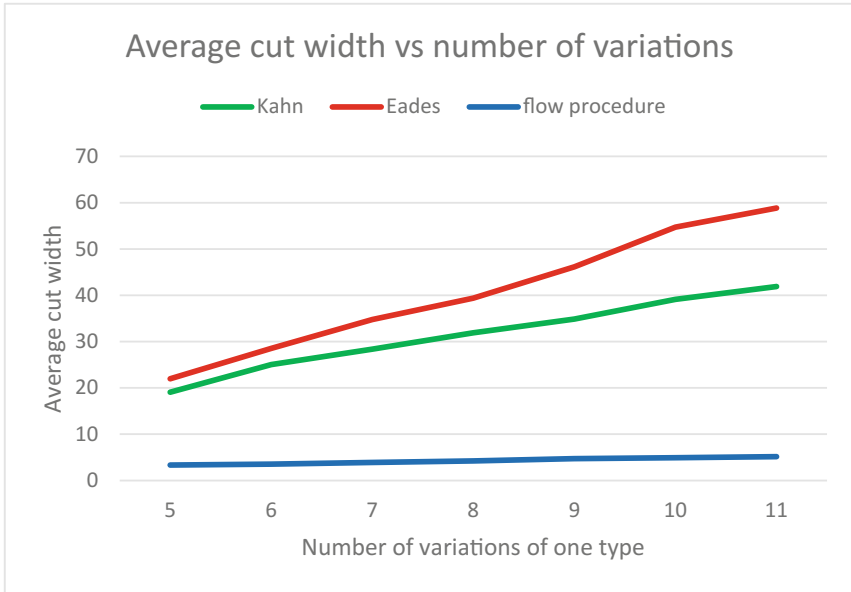


Fig. 12. The relationship between the ACW and number of variations.

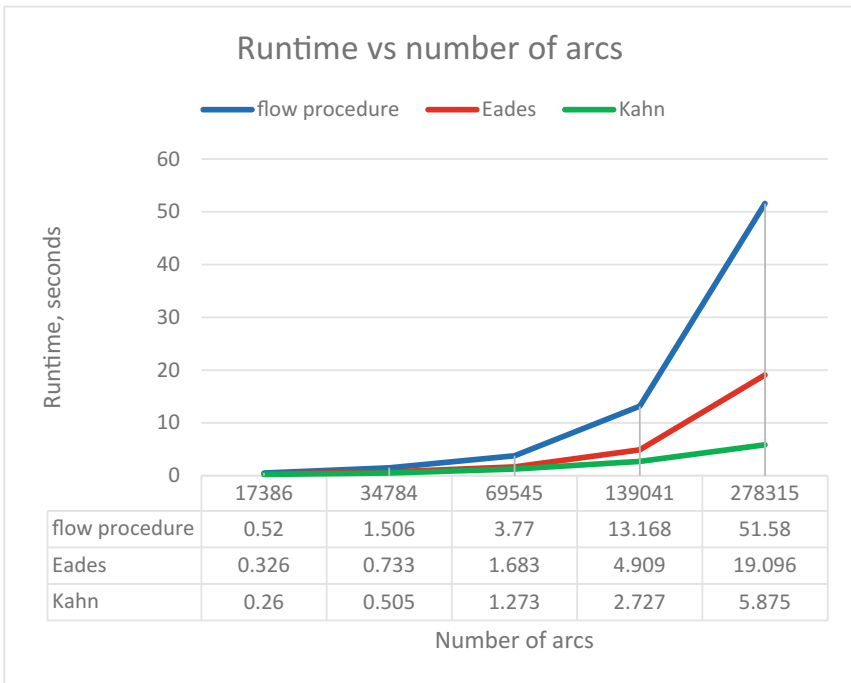


Fig. 13. The relationship between the runtime and the number of arcs in the graph.

To use the FP algorithm in practice, one must estimate the time it takes the algorithm to run on large amounts of data. In order to use the algorithm on large graphs in practice, we would split the graph into pieces using a graph decomposition scheme as described in [2]. In practice the time complexity of the FP is $O(|A|^2)$. On the other hand, both Kahn's and Eades' algorithms have complexity $O(|A| + |V|)$, since they do not pass any node more than twice. The relationship between the number of nodes in the graph and the algorithms' runtimes on our test data is shown in Fig. 13.

The relationship matches the one predicted theoretically. It shows that despite quadratic complexity estimation, it is clearly seen from Fig. 13 the algorithm can be used on big graphs.

In addition to these tests on synthetic data, the algorithms were tested on a graph created from MHC region of chromosome 6 with 251297 nodes. Flow procedure running time was about 40 min.

6 Conclusion

We have proposed a new sequence graph linearization algorithm that outperforms standard methods on the criteria that are important for storing, traversing, analyzing and visualizing genome sequence graphs. The quantitative results thus obtained suggest that this algorithm will prove useful in genome exploration. Earlier work on sequence graph linearization [16] focused on minimizing feedback arcs, here we additionally introduce cut-width as an important measure of a linearization that effectively measures contiguity between elements that are connected. Future effort to lower the computational complexity of the algorithm using graph decomposition (see [2]) will allow us to apply a modified form of the presented algorithm to complete human scale sequence graphs of hundreds of millions of nodes.

Acknowledgements. We'd like to thank Erik Garrison and Glenn Hickey for helpful conversations. This work was supported by the National Human Genome Research Institute of the National Institutes of Health under Award Number 5U54HG007990 and grants from the W.M. Keck foundation and the Simons Foundation. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Appendix

Some Details of the FP Algorithm

Noteworthy is the order in which we find the in- and outgrowths. First, we traverse the backbone from end to start, finding the outgrowth for each node, then we traverse it from start to end, finding the ingrowth. We include in the in- and outgrowths only those nodes that did not end up in any of the previous out- or ingrowths (see Fig. 14).

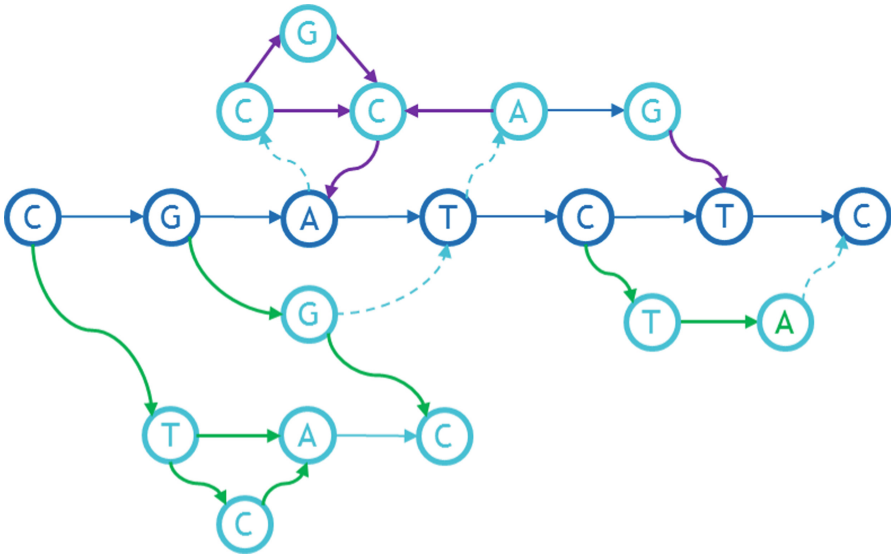


Fig. 14. An example of the in-growths and out-growths for a graph.

Step-by-Step Algorithm Run

Let's start from the moment when we have already found and removed the minimum cut. We go from the beginning to the end over the backbone (CGATC) and find the in-growth CCGA (upper 3 nodes and A from the backbone). For this in-growth we run the entire flow procedure recursively. Looking for the backbone, we start from A and search for incoming max weight arcs. We get CCA, then run the min cut search and remove the CG arc. Then we recursively go to the CCA backbone from the beginning to the end; we are looking for the in-growth. We find GC. For it we run the procedure, which arranges these two nodes in the obvious way. We insert the result into the backbone CCA with the G before the second C (the one that had the in-arc). Thus, we get CGCA. All nodes of this part are sorted, so the recursion is finished and we insert the resulting in-growth into the backbone of the source graph. Inserting to the backbone we get CGCGCATC. There are no other in-growths, so we turn to search for out-growths. We go from the end to the beginning. We find the GGC out-growth. It includes 3 consecutive nodes, so the recursive procedure for it throws out a natural GGC order. We insert to the backbone and get CGGCCGCATC. Then we look for the next out-growth. We find the CTCA starting from the first node of the backbone. For it, we run the procedure recursively. It finds the backbone CTA, then removes the min cut, finds the in-growth CA and inserts its C before the A: CTCA. There are no other in- or out-growths, so this part of the algorithm is finished and we insert nodes to the original backbone, finally getting CTCAGGCCGCATC .

Test Data Set Modeling

In order to simulate the test data, we used the RSVSim package (version 1.14.0) from the Bioconductor software (Release 3.4). As a reference genome, we took BSgenome.Hsapiens.UCSC.hg38 (version 1.4.1), alternative branch chr13_KI270842v1_alt, which is 37287 nucleotides long. Using the simulateSV command of the RSVSim package, we modeled genome fragments of 10 individuals with a given set of variations. Resulting FASTA files were submitted to the entry of the msga command of the vg utility [12]. As a result, we got a sequence graph (*.gfa format). This graph is an input to the commands vg sort-f (Eades) and vg sort (Flow procedure) of the vg utility [12]. Finally, we got text files with graph nodes ordered by linearization using the Kahn, Eades, and flow procedure algorithms respectively. To analyze the algorithm, we created the original software to get the number of feedback arcs and the cut width in abovementioned sorts. To reduce the impact of accidents, we repeated the procedure 20 times for each set of variations and average the results.

We created variation sets as follows. In the modelled genome fragments, we added 5 variation types: insertions, deletions, duplications, inversions, and translocations. The positions of all variations were uniformly distributed over the simulation section of the genome. Twenty percent of the insertions were duplicating sections of the DNA. Translocations were modelled using the shoulder exchange mechanism. The lengths of insertions and deletions were 20 nucleotides; the length of inversion was 200 nucleotides; the length of duplications was 500. The number of variations of each type was equal to 5 in the first set, 6 in the second, 7 in the third, and so on up to 11 in the latest set of variations. The [13] provides a dependence of the number of feedback arcs and cut widths of number of variations of the same type. For this study, the number of variations of all types, except the examined, were fixed at level 7, and the number of investigated variations were changing according to the following list: 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, and 31.

References

1. Paten, B., Novak, A., Haussler, D.: Mapping to a Reference Genome Structure eprint [arXiv:1404.5010](https://arxiv.org/abs/1404.5010)
2. Paten, B., Novak, A.M., Garrison, E., Hickey, G.: Superbubbles, ultrabubbles and cacti. In: Proceedings of RECOMB 2017 (2017)
3. Baharev, A., Schichl, H., Neumaier, A., Achterberg, T.: An exact method for the minimum feedback arc set problem (2016)
4. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer US, New York (1972)
5. Brandenburg, F., Hanauer, K.: Sorting heuristics for the feedback arc set problem. Technical report. Number MIP-1104 (2011)
6. Gavril, F.: Some NP-complete problems on graphs. In: Proceedings of the 11th conference on Information Sciences and Systems, pp. 91–95 (1977)

7. Martí, R., Pantrigo, J., Duarte, A., Pardo, E.: Branch and bound for the cutwidth minimization problem. *Comput. Oper. Res.* **40**, 137–149 (2013). doi:[10.1016/j.cor.2012.05.016](https://doi.org/10.1016/j.cor.2012.05.016)
8. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: *Introduction to algorithms*. Mit Press, Cambridge (Inglaterra) (2009)
9. Medvedev, P., Brudno, M.: Maximum likelihood genome assembly. *J. Comput. Biol.* **16**, 1101–1116 (2009). doi:[10.1089/cmb.2009.0047](https://doi.org/10.1089/cmb.2009.0047)
10. Ford, L.R., Fulkerson, D.R.: *Flows in Networks*. Princeton University Press, Princeton (1962)
11. <https://www.bioconductor.org/packages/release/bioc/html/RSVSim.html>
12. <https://github.com/vgteam/vg>
13. <http://biorxiv.org/content/early/2017/01/18/101501>
14. Kahn, A.: Topological sorting of large networks. *Commun. ACM* **5**, 558–562 (1962). doi:[10.1145/368996.369025](https://doi.org/10.1145/368996.369025)
15. Eades, P., Lin, X., Smyth, W.: A fast and effective heuristic for the feedback arc set problem. *Inf. Process. Lett.* **47**, 319–323 (1993). doi:[10.1016/0020-0190\(93\)90079-O](https://doi.org/10.1016/0020-0190(93)90079-O)
16. Nguyen, N., Hickey, G., Zerbino, D., Raney, B., Earl, D., Armstrong, J., Kent, W., Haussler, D., Paten, B.: Building a pan-genome reference for a population. *J. Comput. Biol.* **22**, 387–401 (2015). doi:[10.1089/cmb.2014.0146](https://doi.org/10.1089/cmb.2014.0146)

Dynamic Alignment-Free and Reference-Free Read Compression

Guillaume Holley^{1,2}(✉), Roland Wittler^{1,2}, Jens Stoye¹, and Faraz Hach^{3,4,5}(✉)

¹ Genome Informatics, Faculty of Technology and Center for Biotechnology,
Bielefeld University, Bielefeld, Germany

guillaume.holley@gmail.com

² International Research Training Group 1906

“Computational Methods for the Analysis of the Diversity
and Dynamics of Genomes”, Bielefeld University, Bielefeld, Germany

³ School of Computing Science, Simon Fraser University, Burnaby, Canada
fhach@sfu.ca

⁴ Department of Urologic Sciences, University of British Columbia,
Vancouver, Canada

⁵ Vancouver Prostate Centre, Vancouver, Canada

Abstract. The advent of High Throughput Sequencing (HTS) technologies raises a major concern about storage and transmission of data produced by these technologies. In particular, large-scale sequencing projects generate an unprecedented volume of genomic sequences ranging from tens to several thousands of genomes per species. These collections contain highly similar and redundant sequences, also known as pan-genomes. The ideal way to represent and transfer pan-genomes is through compression. A number of HTS-specific compression tools have been developed to reduce the storage and communication costs of HTS data, yet none of them is designed to process a pan-genome. In this paper, we present DARRC, a new alignment-free and reference-free compression method. It addresses the problem of pan-genome compression by encoding the sequences of a pan-genome as a guided de Bruijn graph. The novelty of this method is its ability to incrementally update DARRC archives with new genome sequences without full decompression of the archive. DARRC can compress both single-end and paired-end read sequences of any length using all symbols of the IUPAC nucleotide code. On a large *P. aeruginosa* dataset, our method outperforms all other tested tools. It provides a 30% compression ratio improvement in single-end mode compared to the best performing state-of-the-art HTS-specific compression method in our experiments.

Availability. DARRC is available at <https://github.com/GuillaumeHolley/DARRC>.

1 Introduction

Motivation. High Throughput Sequencing (HTS) technologies are constantly improving and making sequencing of genomes more affordable. The second generation of HTS technologies was introduced to the sequencing market in 2007,

enabling higher throughput and drastically reducing the cost of sequencing per genome [20]. As a result, the number of sequenced genomes is growing exponentially [19], making storage and access to these data a problem of main importance. For example, the Sequence Read Archive (SRA) public database was endangered in 2011 because of budgetary constraints [30]. In order to reduce storage and transmission costs, raw sequencing data are often compressed using general purpose compression tools such as gzip (based on Lempel-Ziv-77 [36]) or bzip (based on the Burrows-Wheeler Transform [4]). Although these classic tools compressed most of the public data, they are not optimized for HTS compression [8, 10, 13, 15, 21]. In FASTQ format, each record has three major components: (i) unique identifier, (ii) read sequence and (iii) quality scores. A large variety of HTS-specific compression tools were proposed [1, 3, 11, 12, 17, 18, 22, 25–27] to compress either FASTQ files or only the read sequences. While these tools are very efficient, they are not adapted to the context of large-scale sequencing projects that produce tens to several thousands of genomes per species. A *pan-genome*, a set of genomes belonging to different strains of the same species, is characterized by a high degree of similarity and redundancy between the genomes [31]. All HTS-specific compression tools can only consider redundancy and similarity within a single genome and not in a collection of genomes. Furthermore, large-scale sequencing projects such as the 1000 Genomes Project [7] may take years to complete, making the compression of continually growing pan-genomes a challenging process.

Existing Approaches. HTS-specific compression tools are divided into two categories: *reference-based* and *de novo*. Reference-based methods generally provide high compression ratio by encoding similarities between the read sequences (*reads*) and a reference sequence (*reference*) usually by mapping the reads to the reference. These tools require that the reference used for compression is provided with the compressed archive for decompression, adding extra storage and transmission costs. Note that only a small fraction of sequenced species that are accessible in public databases have such a reference available. On the other hand, *de novo* compression tools perform similarity search within a set of reads in order to exploit its redundancy. BARCODE [26] is a reference-based method that makes use of cascading Bloom filters [29] to compress reads. It inserts reads perfectly matching to a reference into a Bloom filter [2] that can generate false positives. To reduce the number of false positives, BARCODE subsequently inserts them into cascading Bloom filters to tell apart false positives from true positives. Kpath [18] constructs a de Bruijn graph from the reference and encodes each read as a path within the graph. The paths within the graph are then encoded via arithmetic coding [33]. The beginnings of such paths are stored separately in a trie and encoded with LZ-77. QUIP [17] uses a lossless compression algorithm based on adaptive arithmetic encoding of the identifier, read and quality score streams of the FASTQ format. A reference sequence and a sequence alignment of the reads can be used to improve compression of the reads. QUIP can also perform assembly-based compression. Similar methods are used

in FASTQZ and FQZCOMP [3]. SCALCE [12] uses *core substrings* as a measure of similarity in order to cluster similar reads together. Those core substrings are generated via Locally Consistent Parsing (LCP) [28]. SCALCE compresses the reads in each cluster with gzip. ORCOM [11] re-orders reads by similarity as well: it creates clusters of reads that share the same minimizer [24], i.e. the lexicographically smallest p -mer of each read with p usually between 8 to 15. Reads of the same cluster are then merged and compressed. Similar to ORCOM, Mince [22] uses the minimizer approach for clustering. For each read to process, a set of candidate clusters is first established from the k -mers it is composed of. The read is then assigned to the candidate cluster that maximizes the number of q -mers they share. If the read has no candidate cluster, it is assigned to a new cluster corresponding to its minimizer of length k . DSRC 2 [25] compresses the different streams of FASTQ files with different methods: arithmetic coding, Huffman coding [16], as well as 2 bits per base in the case of the DNA sequence stream. Finally, LEON [1] encodes the reads as paths of a de Bruijn graph represented with a Bloom filter. The de Bruijn graph is built from *solid* k -mers of the reads, i.e. k -mers occurring multiple times in the reads. A read is anchored in the graph if it contains at least one solid k -mer and encoded as a list of graph bifurcations from this anchor.

Contributions. In this paper, we present a new alignment-free and reference-free method, DARRC, that compresses the sequencing reads dynamically. The main contribution of this work is the guided de Bruijn graph (gdBG) which allows a unique traversal to reconstruct the reads it was build from. The gdBG is indexed using a colored de Bruijn graph succinct data structure, the Bloom Filter Trie (BFT) [14] that enables the update of the gdBG with reads of other similar genomes. Additional methods are presented to optimize the encoding of the reads. On a large *P. aeruginosa* dataset, DARRC outperforms all other tested tools. It provides a 30% compression ratio improvement in single-end mode compared to the best performing state-of-the-art HTS-specific compression method in our experiments.

2 Methods

A *string* x is a sequence of symbols drawn from a finite, non-empty set, called the *alphabet* \mathcal{A} . Its *length* is denoted by $|x|$. Strings are concatenated by juxtaposition. If $x = ps$ for (potentially empty) strings p and s , then p is a *prefix* and s is a *suffix* of x . The symbol at position i is denoted by $x[i]$, the suffix starting at position i by $x(i)$, the substring starting at position i and having length l by $x(i, l)$.

2.1 The de Bruijn Graph

A de Bruijn graph (dBG) is a directed graph $G = (V_G, E_G)$ in which each vertex $v \in V_G$ represents a k -mer, a string of length k over \mathcal{A} . A directed edge $e \in E_G$

from vertex v to vertex v' representing k -mers x and x' , respectively, exists if and only if $x(2, k - 1) = x'(1, k - 1)$. Each k -mer x has $|\mathcal{A}|$ possible successors $x(2, k - 1)c$ and $|\mathcal{A}|$ possible predecessors $cx(1, k - 1)$ with $c \in \mathcal{A}$. A colored de Bruijn graph (cdBG) is a dBG $G = (V_G, E_G, C_G)$ in which C_G is a set of colors such that each $v \in V_G$ contains a subset of C_G . A lightweight representation of dBGs and cdBGs does not store edges since they are implicitly given by vertices overlapping on $k - 1$ symbols. However, implicit edges can falsely connect vertices that share an overlap of $k - 1$ but do not overlap in the sequences the graph was built from.

The dBG is a long-studied abstract data structure used in computational biology. It is particularly useful for the problem of *read assembly* [6] in which the goal is to reconstruct a genome as a single sequence from a set of reads. For this purpose, it is necessary to find a Hamiltonian cycle in the graph, a path starting and ending on the same vertex that visits each vertex exactly once. Although heuristics exist to extract Hamiltonian cycles from a graph, the read assembly problem is yet to be solved because a Hamiltonian cycle is only one possible reconstruction of the original genome the graph was built from.

2.2 The Guided de Bruijn Graph

The read assembly problem shows that different traversals of dBGs are possible. In the worst case, the number of possible paths between two vertices in a graph is infinite if the graph is cyclic, and exponential otherwise. Given a dBG built from a sequence and a starting vertex for the traversal, the dBG must be supplemented with information to guide its traversal in order to reconstruct the sequence it was built from.

Definition 1. *Given a de Bruijn graph G built from a sequence S , a partition $\text{part}(G, S)$ is a subgraph G' of G such that G' is a path graph that reconstructs a subsequence of S .*

A guided de Bruijn graph (gdBG) built from a sequence S is a cdBG $G = (V_G, E_G, P_G)$ in which the set of colors, now denoted as P_G , represents partitions guiding the traversal of G to reconstruct S . Self-overlapping k -mers, for which the prefix of length $k - 1$ is equal to the suffix of length $k - 1$, require a special treatment to avoid looping on themselves within the same partition. Algorithm 1 creates a gdBG G from a sequence S using vertices of length k . It returns all information necessary to reconstruct S : the gdBG encoding S and the $k - 1$ length prefix of the first k -mer of S starting the graph traversal for decoding. Note that self-overlapping k -mers terminate their partition such that the next inserted k -mers start a new one (line 9). The algorithm requires $\mathcal{O}(|S|)$ time and $\mathcal{O}(|G|)$ space where $|G| = |V_G| + |P_G|$ if the gdBG uses an implicit representation of edges.

Algorithm 2 decodes a sequence S from a gdBG G using vertices of length k starting with k -mer prefix x . Algorithm 1 guarantees that for any k -mer and one of its partitions, this k -mer can only have zero or one successor in the graph with

Algorithm 1. Encode(S, k)

```

1:  $p \leftarrow 1$  ▷ partition index
2:  $G \leftarrow$  the empty graph
3: for  $i \leftarrow 1, \dots, |S| - k + 1$  do
4:    $x \leftarrow S(i, k)$ 
5:    $Y \leftarrow \{y \mid y \text{ successor of } x \text{ in } G \text{ with } p \in G[y]\}$ 
6:   if  $Y \neq \emptyset$  then  $p \leftarrow p + 1$ 
7:   if  $x \in G$  then  $G[x].add(p)$  ▷ add  $p$  to vertex  $x$  in  $G$ 
8:   else  $G.add(x, p)$  ▷ insert vertex  $x$  with  $p$  in  $G$ 
9:   if  $x(2, k - 1) = x(1, k - 1)$  then  $p \leftarrow p + 1$ 
10: return  $(G, S(1, k - 1))$ 

```

the same partition. Therefore, Algorithm 2 traverses the graph by searching, for each traversed vertex, the successor with the same partition. If it is not found, the partition index is incremented and the traversal continues. As for Algorithm 1, the algorithm requires $\mathcal{O}(|S|)$ time and $\mathcal{O}(|G|)$ space.

Algorithm 2. Decode(G, x, k)

```

1:  $p \leftarrow 1$ 
2:  $z \leftarrow k$ -mer  $y$  in  $G$  with  $y(1, k - 1) = x$  and  $p \in G[y]$ 
3:  $x \leftarrow z$ 
4:  $S \leftarrow z$ 
5:  $Z \leftarrow \{z\}$ 
6: if  $z(2, k - 1) = z(1, k - 1)$  then  $p \leftarrow p + 1$ 
7: while  $Z \neq \emptyset$  and  $p \in P_G$  do
8:    $Z \leftarrow \{z \mid z \text{ successor of } x \text{ in } G \text{ with } p \in G[z]\}$ 
9:   if  $Z$  contains exactly one element  $z$  then
10:      $S \leftarrow Sz[k]$ 
11:      $x \leftarrow z$ 
12:     if  $x(2, k - 1) = x(1, k - 1)$  then  $p \leftarrow p + 1$ 
13:   else
14:      $p \leftarrow p + 1$ 
15:      $Z \leftarrow \{x\}$ 
16: return  $S$ 

```

Figure 1 represents a simple cyclic dBG built from a sequence containing a repetition. An infinite number of sequences could be extracted from the dBG because of the cycle. However, by augmenting the dBG with partitions, Algorithm 2 will traverse the cycle only once during the reconstruction of the sequence. Indeed, when Algorithm 1 tries to insert k -mer *agt* with partition 1, a successor with the same partition is found. Therefore, k -mer *agt* is inserted with partition 2 such that the cycle is not contained in one partition.

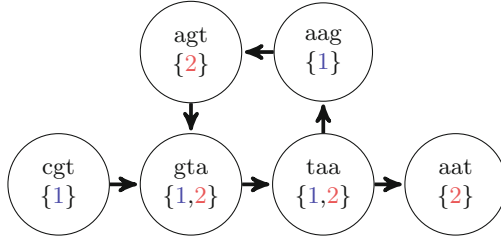


Fig. 1. The guided de Bruijn graph of sequence $S = \text{cgtaagtaat}$ as constructed by Algorithm 1 with $k = 3$.

An important property of gdBGs using implicit edges is that no false implicit edge can be traversed during the decoding.

Proposition 1. *Let G be a gdBG built from a sequence S using an implicit representation of edges. An edge between vertices v and v' corresponding to k -mers x and x' respectively, such that $x(2, k-1) = x'(1, k-1)$ but $xx'[k]$ is not a substring of S is called a false implicit edge. Algorithm 2 does not consider any false implicit edge when traversing G to reconstruct S .*

Proof. If a false implicit edge connects vertices not sharing a partition, Algorithm 2 will not consider this edge as only successors with the same partition are traversed. If a false implicit edge connects vertices v and v' which share a partition, the edge out-degree of v is at least 2 and the edge in-degree of v' is at least 2: one true implicit edge each and at least one false implicit edge each. As these vertices are branching, Definition 1 guarantees that v and v' are not in the same partition. \square

Algorithm 1 does not distinguish true implicit edges from false implicit edges, ensuring that Definition 1 is always respected during the encoding.

Furthermore, partitions allow to apply the following generalized definition of edges in dBGs to gdBGs:

Definition 2. *In a de Bruijn graph, a directed edge from vertex v to vertex v' representing k -mers x and x' , respectively, exists if and only if $x(l+1, k-l) = x'(1, k-l)$ with $l \geq 1$.*

For a sequence S to encode in a gdBG and $l > 1$, $\lfloor \frac{|S|-k+1}{l} \rfloor$ k -mers will be inserted instead of $|S| - k + 1$. However, the graph can contain more partitions as each vertex has now $|\mathcal{A}|^l$ possible successors and predecessors. Figure 2 gives the gdBG encoding the same sequence as in Fig. 1 using a k -mer overlap of $k-2$ instead of $k-1$. The resulting gdBG contains only half the number of vertices than the one in Fig. 1.

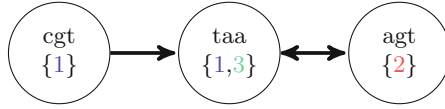


Fig. 2. The guided de Bruijn graph of sequence $S = \text{cgtaagtaat}$ using 3-mers overlapping on $k - l = 1$. The last symbol of S is not encoded in the gdBG as it cannot be part of a k -mer.

3 Compression

Section 2 presented methods to encode a sequence as a gdBG and to decode it. In this section, we describe how to use this methodology to compress reads. To improve compression efficiency, we preprocess the reads.

3.1 Read Clustering and Merging

A simple form of read assembly extended from ORCOM [11] is performed to reduce the input data. It clusters reads according to their minimizer, then merges reads sharing an overlap within each cluster and finally merges reads sharing an overlap but originating from different clusters. These three steps are described in the following.

Clustering. The minimizer [24] of a read r is the lexicographically smallest of its p -mers with $p \ll |r|$. The canonical minimizer of r is the lexicographically smallest minimizer of r and its reverse-complement \bar{r} . The following method is based on the simple assumption that reads sharing a minimizer are likely to share a longer overlap and therefore be similar. Thus, the canonical minimizer m is computed for each read r such that r or \bar{r} is assigned to its cluster m .

Intra-cluster Merging. Within each cluster, the reads are sorted by decreasing position of their minimizer, in which reads sharing the same minimizer position are sorted lexicographically. For each read r and its minimizer m at position p_m , all reads r' with minimizers at positions $p'_m \leq p_m$ are considered for merging, in decreasing order of positions p'_m to maximize the overlap lengths. To merge reads r and r' , they are first anchored at the position of their minimizers such that they overlap on $o = p'_m + \min(|r| - p_m, |r'| - p'_m)$ symbols. Reads are merged into a *super read* [35] if $r(p_m - p'_m, o) = r'(1, o)$ with at most d mismatches. The same process is applied to the created super read in order to merge it with the remaining reads of the cluster. For each super read, we encode all of its read meta data in separate streams: position, length, reverse-complement information and mismatches.

Inter-cluster Merging. As an extension of the previous steps used by ORCOM, we additionally perform a process similar to the intra-cluster read merging described previously to merge super reads from multiple clusters. For each super read sr and its minimizer m at position p_m , a new minimizer m' is computed in $sr(p_m + 1)$ and \overline{sr} . All super reads of cluster m' are considered for a merging with sr or \overline{sr} . Merging two or more super reads creates a *spanning super read* (SSR). The same process is applied to the created SSR until no super reads can merge with it.

Paired-End Reads. Each mate of a pair is considered as a single read that is clustered and merged using the previously described methods. However, the clustering and merging steps keep track of the position of the mates in the SSRs. This information is used afterwards to store in each read meta data whether the read is the first mate of its pair. In such case, the position of its corresponding mate in the SSRs is stored as well.

3.2 Spanning Super Read Encoding

Encoding a set of SSRs using a gDBG requires to extract k -mers from the SSRs. If edges represent overlaps of length $k - 1$, all k -mers of the SSRs are extracted. If edges represent overlaps of length $k - l$ with $l > 1$, k -mers are extracted every l positions. As a consequence, similar SSRs can have different sets of k -mers. An example is given in Fig. 3, in which two similar SSRs, ssr_1 and ssr_2 , do not share any k -mers because they are extracted every $l = 2$ positions from the first position of each SSR. By shifting the k -mer extraction start position by one position in the second SSR, as shown with ssr_2' , two extracted k -mers are shared with the first SSR.

```

ssr1 = a c g t c c t g a a t      ssr2 = g a c g t c c g g a a      ssr'2 = g a c g t c c g g a a

      a c g t                g a c g                a c g t
        g t c c                c g t c                g t c c
          c c t g                t c c g                c c g g
            t g a a                c g g a                g g a a

```

Fig. 3. Extraction of 4-mers overlapping on $k - l = 2$ from two similar SSRs, ssr_1 and ssr_2 .

In order to keep the growth of the gDBG small when inserting a new SSR, we determine the k -mer extraction start position, called *start position* in the following, that maximizes the number of k -mers already stored in the gDBG. To this end, we maintain in memory a k -mer index recording all k -mers extracted. As the cost in time and memory of such an index is prohibitive, we use a Bloom filter instead.

Introduced by Bloom, a *Bloom filter* (BF) [2] records the presence of elements in a set. Based on the hash table principle, look-up and insertion times are constant. The BF is composed of an array B of b bits, initialized with 0s, in which the presence of n elements is recorded. A set of f hash functions h_1, \dots, h_f is used, such that for an element e , $h_i(e) : e \rightarrow \{1, \dots, b\}$. Inserting an element into B and testing for its presence are then

$$\text{Insert}(e, B) : B[h_i(e)] \leftarrow 1 \text{ for all } i = 1, \dots, f$$

and

$$\text{MayContain}(e, B) : \bigwedge_{i=1}^f B[h_i(e)],$$

respectively, in which \bigwedge is the logical conjunction operator. The BF does not generate false negatives but may generate false positives, as `MayContain` can report the presence of elements which are not present but a result of independent insertions.

We propose a greedy approach making use of the BF to iteratively detect for each SSR of a set its optimal start position and updating the BF with all novel k -mers. The optimal start position of an SSR is a position from 1 to l maximizing the number of k -mers extracted that are already present in the BF compared to the other possible start positions. Once the optimal start position of an SSR is determined, the BF is updated with the k -mers extracted and the next SSR of the set is processed. To encode all SSRs completely, this approach does not only return the k -mers to insert into a gDBG, because these do not necessarily cover the entire SSRs. It also returns the *head* and *tail* of each SSR, which are the uncovered prefix and suffix, respectively, not encoded in the gDBG. Additionally, to provide an entry point into the gDBG for the decoding, it returns the *starting overlap* of each SSR, which is the $k - l$ length prefix of the first k -mer. More precisely, we denote by x and y the first and last k -mers extracted, respectively, from an SSR ssr with pos_x and pos_y as their respective occurrence positions in ssr . Then, the head of ssr is the prefix $ssr(1, pos_x - 1)$, the tail of ssr is the suffix $ssr(pos_y + k)$, and the starting overlap of ssr is $ssr(pos_x, k - l)$. SSR heads, tails and starting overlaps are encoded in separate streams and compressed separately from the gDBG.

3.3 Partition Encoding

Encoding. Partition sets associated with k -mers in gDBGs are represented as lists of sorted integers. A naive way to store a partition set is to use a fixed number of bytes for each partition. For example, 4 bytes is a standard size for integers on current computer architectures. In order to decrease the memory footprint while keeping the lists indexed, partitions are first delta encoded by storing the difference between each integer and its predecessor in the list (or 0 if the integer is in first position). The resulting values are called *deltas*. However, it only decreases the minimum number of bits necessary to encode the partitions

but not their final representation. Consequently, deltas are Vbyte encoded [32]: each byte used to encode a delta has one bit indicating whether the byte starts a new delta or not, allowing to remove unnecessary bytes from each delta. Thus, partitions use a variable number of bytes proportional to the minimum number of bits necessary to encode their deltas.

Recycling. As a small delta produces a small encoding, partition integers are recycled instead of naively using the next higher integer for every new partition as, for the sake of convenience, described in Algorithm 1. Partition sets a and b can share the same partition integer if they are not neighbors in the graph, i.e., no k -mer suffix or prefix of a overlaps a k -mer prefix or suffix of b , for suffixes and prefixes of length $k - l$. A trivial example is provided in Fig. 4 in which k -mer $cttc$ uses the same partition integer as k -mer $acgt$ because they are not neighbors in the graph.

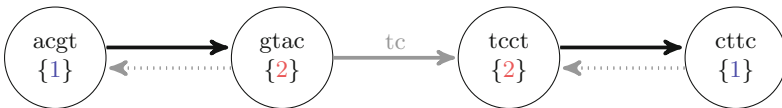


Fig. 4. The guided de Bruijn graph of SSRs $ssr_1 = acgtac$ and $ssr_2 = tccttc$ using 4-mers ($l = 2$). Dotted gray edges are false implicit edges. The solid gray edge exists by using the starting overlap of ssr_2 after the traversal of ssr_1 , as described in Sect. 3.2.

As there can be a large number of partitions in the graph, verifying the connectivity of one partition to all other partitions is often impractical. We propose instead a heuristic that verifies the connectivity only to the last t partitions inserted, t being a user-defined threshold, such that these t partitions are the only candidates for recycling. Using partition recycling requires to save the partitions traversal order which cannot be incremental anymore as shown in Algorithms 1 and 2.

3.4 Meta Data and GDBG Compression

Steps described previously generate meta data specific to one input file such as read lengths and positions in SSRs. These meta data are first encoded in separate streams and are then compressed using an LZ-type algorithm, LZMA [23]. After all k -mers and partitions are inserted in the gDBG, the latter is written to disk. As it must be loaded in memory for every update and decompression, the gDBG is compressed with Zstd [5], a compression method based on Huffman coding and Asymmetric Numeral Systems [9] that favors compression and decompression speed over compression ratio.

4 Update and Decompression

In order to update a compressed archive with a new input file, only the gdBG previously created is decompressed and loaded in memory, as meta data are not used for the update. A fast procedure iterates over all k -mers of the gdBG and inserts them into the BF proposed in Sect. 3.2 instead of starting with an empty BF in order to optimize the choice of the k -mer extraction start positions in the SSRs. The gdBG is then updated with the new k -mers and partitions. The starting partition index is greater than the partition indexes already present in the gdBG, ensuring that each input file is encoded with a unique set of partitions.

Decompressing a read file starts with decompressing its meta data and the gdBG it is encoded in. The gdBG is then loaded in memory and Algorithm 2 is used to traverse the gdBG, but only following those partitions that are specific to the read file to decompress. This way, single files can be decompressed separately. As Algorithm 2 decodes SSRs, meta data are used afterwards to extract the actual reads. If reads are paired-end, meta data are also used to reorganize them such that corresponding mates of the same pair are together in the decompressed file.

5 Results

DARRC is implemented in C and uses the Bloom Filter Trie (BFT) library [14] for its gdBG. The BFT provides time and space efficient functionalities that are required by DARRC. These functionalities include: (i) the ability to update the BFT with new k -mers and colors without recomputing the index, (ii) k -mers extraction from the BFT and (iii) prefix search over the set of k -mers within the BFT. The software is available at <https://github.com/GuillaumeHolley/DARRC>. We compared DARRC to three state-of-the-art *de novo* DNA sequence compression tools: ORCOM [11], LEON [1] and Mince [22]. DARRC was also compared to the same LZ-type algorithm used to compress its meta data, LZMA [23]. Experiments were carried out on a server with 378 GB of RAM and two 8-core Intel Xeon E5-2630 v3 2.4 GHz processors. All input files were placed on mechanical hard drives. Compressed archives and decompressed files, during compression and decompression respectively, together with temporary files such as read clusters were written to a RAM-based partition when the tools allowed to specify an output directory. As the current version of DARRC does not take advantage of parallelism, all software were run using a single thread, except Mince which requires a minimum of four threads. All *de novo* DNA sequence compression tools were run using their default parameters. LZMA was run with the same compression level as the one used to compress DARRC meta data. DARRC default parameters are minimizers of length 9 for the clustering, 5 mismatches allowed per read merging and 36-mers overlapping on 11 symbols for the gdBG. ORCOM, LEON, Mince and LZMA compressed all files in separate archives while DARRC updated the same archive iteratively with the files to compress: each iteration decompressed and reloaded the necessary data

from the data written to disk in the previous iteration. The dataset used for the experiment consists of 473 clinical isolates of *Pseudomonas aeruginosa* sampled from 34 patients (NCBI BioProject PRJEB5438), resulting in 338.61 Gbp of high coverage sequences. Reads are 100 bp paired-end reads generated by Illumina HiSeq 2000. Pair mates were placed in different files for every isolate. The experiment was run in single-end mode and paired-end mode for all tools such that in the single-end mode, every mate file is considered as a single-end read file. The appropriate single-end and paired-end modes were used for DARRC and Mince. The mates were concatenated for the paired-end experiment of ORCOM as the tool neither preserves the order of the reads nor stores the paired-end information. LEON and LZMA do not have an explicit paired-end mode but keep the original order of the reads, thus for the paired-end experiment of LEON and LZMA, the mate files of every isolate were concatenated.

Compression ratios in paired-end mode and single-end mode are shown in Fig. 5. DARRC clearly outperforms all the other tested tools in both modes. In paired-end mode and single-end mode, DARRC uses about 0.261 bits per base and 0.204 bits per base, corresponding to a 57% and 30% compression ratio improvement compared to the second best results, respectively. The paired-end compression ratio of ORCOM compared to its single-end compression ratio shows that the tool is not adapted to paired-end read compression. The gdBG represents about 10% and 13% of the data written to disk by DARRC in paired-end mode and single-end mode, respectively.

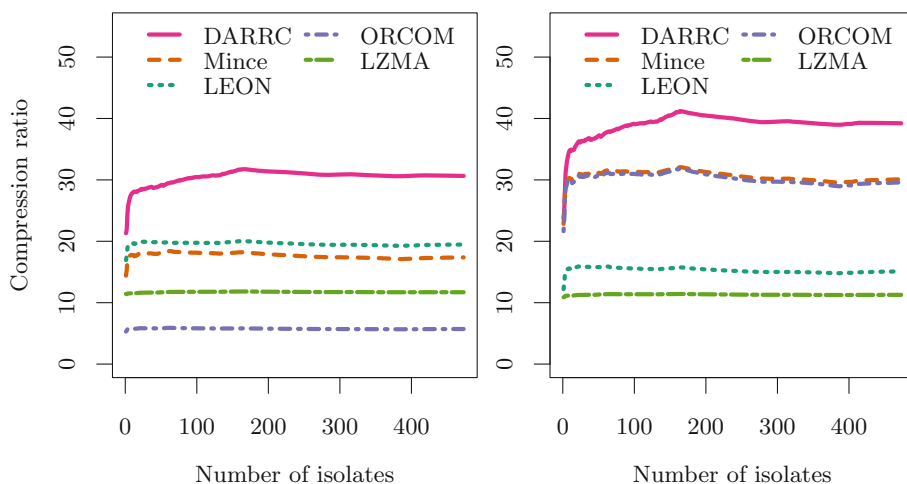


Fig. 5. Compression ratios in paired-end mode (left) and single-end mode (right).

DARRC compressed more than two times faster than LZMA but used the most time to decompress, as shown in Figs. 6 and 7, respectively. DARRC's compression time overhead is explained by the fact that at each iteration, the

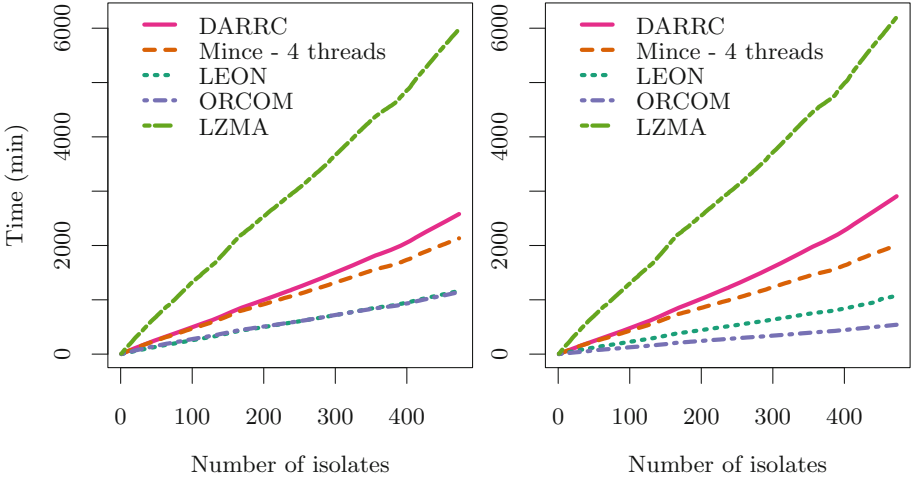


Fig. 6. Compression times in paired-end mode (left) and single-end mode (right).

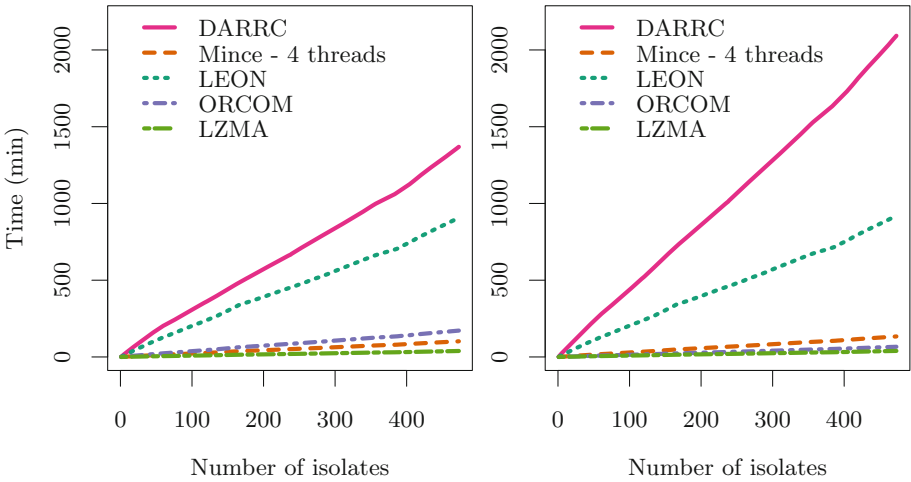


Fig. 7. Decompression times in paired-end mode (left) and single-end mode (right).

gdBG must be decompressed, loaded in memory and updated with new k -mers and partitions.

All tools performed compression and decompression using a maximum of four gigabytes of main memory, an amount nowadays available on most desktop computers and laptops. Even by updating the same archive iteratively, DARRC compression used less than two gigabytes of main memory.

6 Conclusions and Future Work

We presented DARRC, a dynamic alignment-free and reference-free read compression method that can incrementally update compressed archives with new genome sequences without full decompression of the archives. DARRC uses a new abstract data structure, the guided de Bruijn graph, that allows a unique traversal of the de Bruijn graph to reconstruct the sequences it is built from. We showed that, on a large pan-genome dataset, our method outperforms several state-of-the-art DNA sequence compression methods and a general purpose compression tool regarding the compression ratio while achieving reasonable running time and main memory usage. Furthermore, we showed that the compression ratio of DARRC is attractive even with only few files compressed. Future work concerns the parallelization of the software, particularly the read clustering and merging phase which offers a lot of potential for multi-threading. Additionally, a logical evolution of DARRC is the introduction of a pattern matching functionality within the compressed data as in [34], leading to large scale complex methods such as read alignment and variant calling using multiple genomes.

Acknowledgments. This research is funded by the International DFG Research Training Group GRK 1906/1 for GH and RW, the NSERC Discovery Frontiers grant on “Cancer Genome Collaboratory” to FH.

References

1. Benoit, G., Lemaitre, C., Lavenier, D., Drezen, E., Dayris, T., Uricaru, R., Rizk, G.: Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph. *BMC Bioinform.* **16**, 288 (2015)
2. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Comm. ACM* **13**(7), 422–426 (1970)
3. Bonfield, J.K., Mahoney, M.V.: Compression of FASTQ and SAM format sequencing data. *PloS One* **8**(3), e59190 (2013)
4. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Digital SRC Research Report 124 (1994)
5. Collet, Y.: ZSTD. <https://github.com/facebook/zstd>, 20 December 2016
6. Compeau, P.E.C., Pevzner, P.A., Tesler, G.: How to apply de Bruijn graphs to genome assembly. *Nat. Biotechnol.* **29**(11), 987–991 (2011)
7. 1000 Genomes Project Consortium: A global reference for human genetic variation. *Nature* **526**(7571), 68–74 (2015)
8. Deorowicz, S., Grabowski, S.: Data compression for sequencing data. *Algorithms Mol. Biol.* **8**, 25 (2013)
9. Duda, J.: Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding (2013). [arXiv:1311.2540](https://arxiv.org/abs/1311.2540)
10. Giancarlo, R., Rombo, S.E., Utro, F.: Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Brief. Bioinform.* **15**(3), 390–406 (2014)
11. Grabowski, S., Deorowicz, S., Roguski, L.: Disk-based compression of data from genome sequencing. *Bioinformatics* **31**(9), 1389–1395 (2014)

12. Hach, F., Numanagić, I., Alkan, C., Sahinalp, S.C.: SCALCE: boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics* **28**(23), 3051–3057 (2012)
13. Holland, R.C.G., Nick, L.: Sequence squeeze: an open contest for sequence compression. *GigaScience* **2**(1), 5 (2013)
14. Holley, G., Roland, W., Stoye, J.: Bloom Filter Trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms Mol. Biol.* **11**, 3 (2016)
15. Hosseini, M., Pratas, D., Pinho, A.J.: A survey on data compression methods for biological sequences. *Information* **7**(4), 56 (2016)
16. Huffman, D.A.: A method for the construction of minimum-redundancy codes. In: *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101 (1952)
17. Jones, D.C., Ruzzo, W.L., Peng, X., Katze, M.G.: Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* **40**(22), e171 (2012)
18. Kingsford, C., Patro, R.: Reference-based compression of short-read sequences using path encoding. *Bioinformatics* **31**(12), 1920–1928 (2015)
19. Land, M., Hauser, L., Jun, S.-R., Nookaew, I., Leuze, M.R., Ahn, T.-H., Karpinets, T., Lund, O., Kora, G., Wassenaar, T., et al.: Insights from 20 years of bacterial genome sequencing. *Funct. Integr. Genomics* **15**(2), 141–161 (2015)
20. Loh, P.-R., Baym, M., Berger, B.: Compressive genomics. *Nat. Biotechnol.* **30**, 627–630 (2012)
21. Numanagić, I., Bonfield, J.K., Hach, F., Voges, J., Ostermann, J., Alberti, C., Mattavelli, M., Sahinalp, S.C.: Comparison of high-throughput sequencing data compression tools. *Nat. Methods* **13**(12), 1005–1008 (2016)
22. Patro, R., Kingsford, C.: Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics* **31**(17), 2770–2777 (2015)
23. Pavlov, I.: LZMA. <http://www.7-zip.org>, 20 December 2016
24. Roberts, M., Hayes, W., Hunt, B.R., Mount, S.M., Yorke, J.A.: Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**(18), 3363–3369 (2004)
25. Roguski, L., Deorowicz, S.: DSRC 2-Industry-oriented compression of FASTQ files. *Bioinformatics* **30**(15), 2213–2215 (2014)
26. Rozov, R., Shamir, R., Halperin, E.: Fast lossless compression via cascading Bloom filters. *BMC Bioinform.* **15**(9), S7 (2014)
27. Saha, S., Rajasekaran, S.: Efficient algorithms for the compression of FASTQ files. In: *Proceedings of the International Conference on Bioinformatics and Biomedicine (BIBM 2014)*, pp. 82–85 (2014)
28. Sahinalp, S.C., Vishkin, U.: Efficient approximate and dynamic matching of patterns using a labeling paradigm. In: *FOCS*, pp. 320–328 (1996)
29. Salikhov, K., Sacomoto, G., Kucherov, G.: Using cascading Bloom filters to improve the memory usage for de Bruijn graphs. *Algorithm. Mol. Biol.* **9**(1), 2 (2014)
30. Genome Biology Editorial Team: Closure of the NCBI SRA and implications for the long-term future of genomics data storage. *Genome Biol.* **12**(3), 402 (2011)
31. Tettelin, H., Masignani, V., Cieslewicz, M.J., Donati, C., Medini, D., Ward, N.L., Angiuoli, S.V., Crabtree, J., Jones, A.L., Durkin, A.S., et al.: Genome analysis of multiple pathogenic isolates of *Streptococcus agalactiae*: implications for the microbial pan-genome. *Proc. Natl. Acad. Sci. USA* **102**(39), 13950–13955 (2005)
32. Williams, H.E., Zobel, J.: Compressing integers for fast file access. *Comput. J.* **42**(3), 193–201 (1999)

33. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. *Commun. ACM* **30**(6), 520–540 (1987)
34. Yu, Y.W., Daniels, N.M., Danko, D.C., Berger, B.: Entropy-scaling search of massive biological data. *Cell Syst.* **1**(2), 130–140 (2015)
35. Zimin, A.V., Marçais, G., Puiu, D., Roberts, M., Salzberg, S.L., Yorke, J.A.: The MaSuRCA genome assembler. *Bioinformatics* **29**(21), 2669–2677 (2013)
36. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **23**(3), 337–343 (1977)

A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases

Chirag Jain^{1,2}, Alexander Dilthey², Sergey Koren², Srinivas Aluru¹,
and Adam M. Phillippy²(✉)

¹ Georgia Institute of Technology, Atlanta, GA 30332, USA
aluru@cc.gatech.edu

² National Institutes of Health, Bethesda, MD 20894, USA
adam.phillippy@nih.gov

Abstract. Emerging single-molecule sequencing technologies from Pacific Biosciences and Oxford Nanopore have revived interest in long read mapping algorithms. Alignment-based seed-and-extend methods demonstrate good accuracy, but face limited scalability, while faster alignment-free methods typically trade decreased precision for efficiency. In this paper, we combine a fast approximate read mapping algorithm based on minimizers with a novel MinHash identity estimation technique to achieve both scalability and precision. In contrast to prior methods, we develop a mathematical framework that defines the types of mapping targets we uncover, establish probabilistic estimates of p-value and sensitivity, and demonstrate tolerance for alignment error rates up to 20%. With this framework, our algorithm automatically adapts to different minimum length and identity requirements and provides both positional and identity estimates for each mapping reported. For mapping human PacBio reads to the hg38 reference, our method is 290x faster than BWA-MEM with a lower memory footprint and recall rate of 96%. We further demonstrate the scalability of our method by mapping noisy PacBio reads (each ≥ 5 kbp in length) to the complete NCBI RefSeq database containing 838 Gbp of sequence and $> 60,000$ genomes.

Keywords: Long read mapping · Jaccard · MinHash · Winnowing · Minimizers · Sketching · Nanopore · PacBio

1 Introduction

Mapping reads generated by high-throughput DNA sequencers to reference genomes is a fundamental and widely studied problem [16, 24]. The problem is particularly well studied for short read sequences, for which effective mapping algorithms and widely used software such as BWA [15] and Bowtie [12] have been developed. The increasing popularity of single-molecule sequencers from Pacific Biosciences and Oxford Nanopore, and their continually improving read lengths (10 kbp and up), is generating renewed interest in long read mapping algorithms. However, the benefit of long read lengths is currently accompanied

The rights of this work are transferred to the extent transferable according to title 17 § 105 U.S.C.

by much higher error rates (up to 15–20%). Despite their high error rates, long reads have proved advantageous in many applications including *de novo* genome assembly [7, 10] and real time pathogen identification [2, 22].

Sequence data from nanopore devices is available just minutes after introducing the sample. This can enable real-time genomic analysis when coupled with fast computational methods that can map the data stream against large reference databases. However, mapping raw sequences continues to be a bottleneck for many applications. The problem is only going to worsen as Oxford Nanopore’s PromethION is projected to generate multiple tera-bases of sequence per day. In parallel, reference databases are continually growing in size, with the non-redundant NCBI RefSeq database close to exceeding a tera-base in size. The high error rate of raw single-molecule sequences further adds to the computational complexity.

Read mapping problems can be solved exactly by designing appropriate variants of the Smith-Waterman alignment algorithm [27]; however, it is computationally prohibitive when mapping reads from a high throughput sequencer to large reference genomes. Seed-and-extend mapping heuristics address this limitation for both long and short reads by limiting the search to locations where exact short word or maximal common substring matches occur before executing an alignment algorithm at these locations [1, 5, 8]. Accurate alignment-based long read mappers include BLASR [5] and BWA-MEM [13]. However, repetitive seeds that do not translate to correct mappings combined with high sequencing error rates limit their scalability. Additionally, alignment-based mapping algorithms preserve the complete reference sequence in the index, and hence, cannot scale to tera-base scale reference databases. Many genomics applications do not require detailed base-to-base alignment information, and instead use only the alignment boundary and identity summaries. Such applications include depth-of-coverage analysis, metagenomic read assignment, structural variant detection, and selective sequencing [18]. Efficient algorithms for these problems, combined with nanopore sequencing, could enable the real-time genomic analysis of patients, pathogens, cancers, and microbiomes.

One class of algorithms for fast, approximate mapping relies on ideas originally developed for finding similarities between web documents. Broder [4] proved that an unbiased estimate of the Jaccard similarity coefficient between two sets can be computed efficiently using a subset of hashed elements called a sketch. Schleimer *et al.* [25] proposed the winnowing algorithm, which picks a minimum hashed item (also known as a minimizer [23]) from each consecutive window of text as a means to more quickly estimate local similarity between web documents. These ideas have been used to develop new mapping and assembly algorithms for long reads such as the MinHash Alignment Process [3], minimap [14], and BALAUR [21]. To date, the effectiveness of these approaches has only been demonstrated empirically.

In this paper, we propose a fast approximate algorithm for mapping long reads that scales to large reference databases with sufficient theoretical guarantees and practical validation on the quality of results reported. We propose a

problem formulation that mathematically characterizes desired mapping targets by linking the Jaccard coefficient between the k -mer spectra of the read and its mapping region to a sequence error rate assuming a Poisson error model. We then provide an efficient algorithm to estimate the Jaccard coefficient through a combination of MinHash and winnowing techniques that characterizes and guarantees the types of mapping regions we find. On the quality side, we provide probabilistic bounds on sensitivity. We present techniques for choosing algorithmic parameters as a function of error rate and sequence lengths that guarantees the desired statistical significance. The theory is validated using PacBio and MinION datasets, and we demonstrate the scalability of our approach by mapping PacBio metagenomic reads to the entire RefSeq database. The speed and space efficiency of our algorithm enables real-time mapping, and compared to minimap, our method maintains high sensitivity with better precision for large, repetitive genomes. The implementation is available at github.com/MarBL/MashMap.

2 Preliminaries

Read Error Model: We assume errors occur independently at the read positions, and use a Poisson error model as in previous works [9, 19]. A binomial model would also be appropriate, but is not discussed here for brevity. Let $\epsilon \in [0, 1]$ be the per-base error rate. The expected number of errors in a k -mer is $k \cdot \epsilon$, and the probability of no errors within each k -mer, assumed independent, is $e^{-\epsilon k}$. We assume the statement is valid irrespective of error type.

Jaccard Similarity: Assuming \mathcal{X}, \mathcal{Y} are the sets of k -mers in sequences X and Y respectively, their Jaccard similarity is defined as $J(X, Y) = |\mathcal{X} \cap \mathcal{Y}| / |\mathcal{X} \cup \mathcal{Y}|$. The Poisson error model is used to compute the relationship between Jaccard similarity and alignment error rate [19]. We approximate the length of a read alignment to be the read length. Let A be a read derived from B_i , where B_i denotes the length $|A|$ substring of reference B starting at position i . If c and n denote the number of error-free and total k -mers in A , respectively, then the expected value of c/n , termed *k -mer survival probability*, is $e^{-\epsilon k}$. This equation assumes k is large enough such that k -mers in A or B_i are unique. Because $|A| = |B_i|$, $J(A, B_i)$, abbreviated as J , equals $c/(2n - c)$. Using the two equations, we derive the following functions \mathcal{G} and \mathcal{F} to estimate J and ϵ :

$$\mathcal{G}(\epsilon, k) = \frac{1}{2e^{\epsilon k} - 1} \quad \text{and} \quad \mathcal{F}(J, k) = \frac{-1}{k} \times \log \left(\frac{2J}{1 + J} \right), \quad (1)$$

where $\mathcal{G}(\epsilon, k)$ serves as an estimate of the Jaccard similarity given an error rate and $\mathcal{F}(J, k)$ estimates the converse. Note $\mathbb{E}(J) \geq \mathcal{G}(\epsilon, k)$ (using Jensen's inequality).

MinHash Approximation: The MinHash algorithm is a fast and space-efficient approximation technique to compute an unbiased estimate of Jaccard similarity [4], without explicitly computing the underlying set intersection and union. Let s be a fixed parameter. Assuming universe U is the totally ordered

set of all possible items, and $\Omega : U \rightarrow U$ is a permutation chosen uniformly at random, Broder [4] proved that $P(\min_{x \in \mathcal{A}} \Omega(x) = \min_{x \in \mathcal{B}_i} \Omega(x)) = J(\mathcal{A}, \mathcal{B}_i)$, and that

$$|S(\mathcal{A} \cup \mathcal{B}_i) \cap S(\mathcal{A}) \cap S(\mathcal{B}_i)| / |S(\mathcal{A} \cup \mathcal{B}_i)| \quad (2)$$

is an unbiased estimate of $J(\mathcal{A}, \mathcal{B}_i)$, where $S(\mathcal{A})$ (called the *sketch* of \mathcal{A}) is the set of the smallest s hashed items in \mathcal{A} , i.e., $S(\mathcal{A}) = \min_s \{\Omega(x) : x \in \mathcal{A}\}$. Typically, the denominator $|S(\mathcal{A} \cup \mathcal{B}_i)|$ is referred as the MinHash sketch size and the numerator as the count of shared sketch elements. This estimate is unbiased provided $S(\mathcal{A})$ is a simple random sample of \mathcal{A} . Increasing the sketch size improves the accuracy of the estimate.

Assuming s is fixed and the true Jaccard similarity $j = J(\mathcal{A}, \mathcal{B}_i)$ is known, the count of shared sketch elements between $S(\mathcal{A})$ and $S(\mathcal{B}_i)$ follows a hypergeometric distribution. Since s is much smaller than $|\mathcal{A}|$, it can be approximated by the binomial distribution.

$$p(|S(\mathcal{A} \cup \mathcal{B}_i) \cap S(\mathcal{A}) \cap S(\mathcal{B}_i)| = x | s, j) = \binom{s}{x} j^x (1-j)^{s-x} \quad (3)$$

As an example, Fig. 1 illustrates this distribution for a read with known Jaccard similarity $j = \mathcal{G}(\epsilon = 0.15, k = 16)$ (using Eq. 1) and sketch size s varying from 200 to 500.

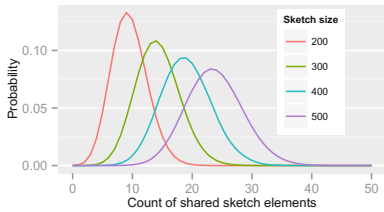


Fig. 1. Probability distributions of count of shared sketch elements for a read with 15% alignment error ($\epsilon = 0.15$) and k -mer size of 16, with varying sketch sizes. Estimated Jaccard similarity computed using Eq. 1 is 0.0475.

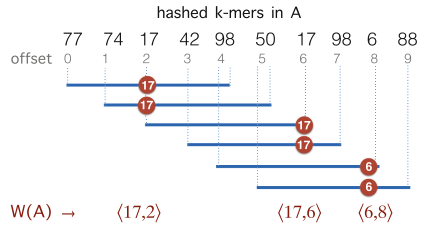


Fig. 2. Illustration of the winnowing method on a sequence of hashed k -mers in A . $W(A)$ represents the minimizers sampled from the sequence with window size $w = 5$.

Winnowing: Winnowing is a local fingerprinting algorithm, proposed to measure similarity between documents by using a subset of hashed words [25]. Unlike MinHash sketching, it bounds the maximum positional gap between any two consecutive selected hashes. It works by sampling the smallest hashed item in every consecutive fixed size sliding window (Fig. 2). Formal description of this algorithm in the context of genomic sequences follows.

Let A_0 denote the set of all k -mer tuples $\langle k_i, i \rangle$ in sequence A , i denoting the k -mer position. Let w be the window-size used for winnowing, and K_j be the

set of w consecutive k -mer tuples starting at position j in A , i.e., $K_j = \{\langle k_i, i \rangle : j \leq i < j + w\}$. Assume Ω is a hash function defined as a random permutation. Then, the set of *minimizers* sampled by the winnowing algorithm in sequence A is $W(A) = \{ \min_{\langle k, i \rangle \in K_j} \langle \Omega(k), i \rangle : 0 \leq j \leq |A_0| - w \}$, where

$$\min(\langle k_1, i_1 \rangle, \langle k_2, i_2 \rangle) = \begin{cases} \langle k_1, i_1 \rangle & k_1 < k_2 \text{ or } (k_1 = k_2 \text{ and } i_1 > i_2); \\ \langle k_2, i_2 \rangle & \text{otherwise;} \end{cases}$$

Schleimer *et al.* [25] prove that the expected set count of minimizers selected from a random sequence A is $2|A_0|/w$. Moreover, $W(A)$ can be computed efficiently in $O(|A|)$ time and $O(w)$ space using a double-ended queue, as sequence A is read in a streaming fashion [26].

3 Problem Formulation

Given a read A and the maximum per base error rate ϵ_{max} , our goal is to identify target positions in reference B where A aligns with $\leq \epsilon_{max}$ per-base error rate. This problem can be exactly solved in $O(|A| \cdot |B|)$ time by designing a suitable quadratic time alignment algorithm. When mapping to a large database of reference sequences, solving this problem exactly is computationally prohibitive. Hence, we define an approximate version of this problem using the Jaccard coefficient as a proxy for the alignment as follows: Let B_i denote the substring of size $|A|$ in B starting at position i ($0 \leq i \leq |B| - |A|$). For a given k , we seek all mapping positions i in B such that

$$J(A, B_i) \geq \mathcal{G}(\epsilon_{max}, k) - \delta \tag{4}$$

Note that if A aligns with B_i with per-base error rate $\leq \epsilon_{max}$, then $\mathbb{E}(J(A, B_i)) \geq \mathcal{G}(\epsilon_{max}, k)$ (using Eq. 1). As this equation applies only to the expected value of $J(A, B_i)$, we lower this threshold by δ to account for variation in the estimate. The parameter δ is defined as the margin of error in Jaccard estimation using a 90% confidence interval.

4 The Proposed Algorithm

Directly computing $J(A, B_i)$ for all positions i is as asymptotically expensive as the alignment algorithm. The rationale for reformulating the problem in terms of Jaccard coefficients is that it enables the design of fast heuristic algorithms. We present an algorithm to estimate $J(A, B_i)$ efficiently using a combination of MinHash and winnowing techniques. In addition, we compute an estimate of the alignment error rate ϵ for each mapping reported. Our method relies on an indexing and search strategy we developed to prune the incorrect mapping positions efficiently.

4.1 Definitions

Let $W(A)$ be the set of minimizers computed for read A using the winnowing method with window-size w . We sketch $W(A)$ instead of sketching A itself. Assuming s is a fixed parameter, we define $S(W(A))$ as the set of the s smallest hashed k -mers that were sampled using winnowing of A , i.e., $S(W(A)) = \min_s \{h : \langle h, pos \rangle \in W(A)\}$. To estimate $J(A, B_i)$, we define *winnowed-minhash* estimate $\mathcal{J}(A, B_i)$ for $J(A, B_i)$ as

$$\mathcal{J}(A, B_i) = \frac{|S(W(A) \cup W(B_i)) \cap S(W(A)) \cap S(W(B_i))|}{|S(W(A) \cup W(B_i))|} \quad (5)$$

In contrast to the MinHash approximation (Eq. 2), our estimator $\mathcal{J}(A, B_i)$ uses winnowing to reduce the sampling frame before picking the minimum hash values. Even though $S(W(A))$ is no longer a simple random sample of the k -mers in A , we empirically show in Sect. 8.1 that the quality of the Jaccard estimation using $\mathcal{J}(A, B_i)$ is as good as the MinHash estimation. We use $W_h(A)$ to denote the set of hashed k -mers in $W(A)$, i.e., $W_h(A) = \{h : \langle h, pos \rangle \in W(A)\}$.

4.2 Indexing the Reference

Retaining the minimizers $W(B_i)$ is sufficient for Jaccard similarity estimation $\mathcal{J}(A, B_i)$ (Eq. 5). Since $W(B_i) \subseteq W(B)$ (Sect. 2), we compute $W(B)$ from the reference sequence B in order to be able to extract $W(B_i)$ efficiently for any i . The set $W(B)$ can be computed from B in a linear scan in $O(|B|)$ time. We store $W(B)$ as an array \mathcal{M} of tuples $\langle h, pos \rangle$. When created, the set is naturally in ascending sorted order of the positions. Further, to enable $O(1)$ look-up of all the occurrences of a particular minimizer’s hashed value h , we also replicate $W(B)$ as a hash table \mathcal{H} with h as the key and an array of its positions $\{pos : \langle h, pos \rangle \in W(B)\}$ as the mapped value. The expected space requirements for \mathcal{M} and \mathcal{H} are $2|B|/w$ (Sect. 2). We postpone our discussion on how to compute an appropriate window-size w to Sect. 5. Besides low memory requirements, a key advantage of this indexing strategy is that a new reference sequence can be incrementally added to the existing data structure in time linear to its length, which is not feasible for suffix array or Burrows-Wheeler transform based indices, typically used in most mapping software.

4.3 Searching the Reference

The goal of the search phase is to identify for each read A , positions i such that $J(A, B_i) \geq \mathcal{G}(\epsilon_{max}, k) - \delta$. We instead compute the winnowed-minhash estimate $\mathcal{J}(A, B_i)$. Let $\tau = \mathcal{G}(\epsilon_{max}, k) - \delta$. To avoid directly evaluating $\mathcal{J}(A, B_i)$ for each B_i , we state and prove the following theorem:

Theorem 1. *Assuming sketch size $s \leq |W_h(A)|$,*

$$\mathcal{J}(A, B_i) \geq \tau \Rightarrow |W_h(A) \cap W_h(B_i)| \geq s \cdot \tau \quad \forall i \ 0 \leq i \leq |B| - |A|.$$

Proof.

$$s \leq |W_h(A)| \implies |S(W(A) \cup W(B_i))| = s \quad (6)$$

From Eq. 5.

$$\begin{aligned} \mathcal{J}(A, B_i) \geq \tau &\implies \frac{|S(W(A) \cup W(B_i)) \cap S(W(A)) \cap S(W(B_i))|}{|S(W(A) \cup W(B_i))|} \geq \tau \\ &\implies \frac{|S(W(A) \cup W(B_i)) \cap S(W(A)) \cap S(W(B_i))|}{s} \geq \tau \quad (\text{using Eq. 6}) \end{aligned}$$

Note that $S(W(A) \cup W(B_i)) \subseteq S(W(A)) \cup S(W(B_i))$. Therefore,

$$\begin{aligned} &\frac{|(S(W(A)) \cup S(W(B_i))) \cap S(W(A)) \cap S(W(B_i))|}{s} \geq \tau \\ &\implies |S(W(A)) \cap S(W(B_i))| \geq s \cdot \tau \end{aligned}$$

But, $S(W(A)) \subseteq W_h(A)$ and $S(W(B_i)) \subseteq W_h(B_i)$

Therefore, $|W_h(A) \cap W_h(B_i)| \geq s \cdot \tau$ □

We use the above condition as a filter and only consider positions in B which satisfy $|W_h(A) \cap W_h(B_i)| \geq s \cdot \tau$. To maximize effectiveness of the filter, we set the sketch size $s = |W_h(A)|$. The search proceeds in two successive stages. The first stage identifies candidate positions i using Theorem 1, and the second stage computes $\mathcal{J}(A, B_i)$ at each candidate position i . The position is retained as output if $\mathcal{J}(A, B_i) \geq \tau$, and discarded otherwise.

Stage 1: Algorithm 1 outlines the first stage of our mapping procedure. It calculates all offset positions i in B such that $|W_h(A) \cap W_h(B_i)| \geq \lceil s \cdot \tau \rceil = m$. The output list T is created in the form of one or more tuple ranges $\langle x, y \rangle$, implying that the criterion holds true for all B_i , $x \leq i \leq y$. We begin by computing the minimizer hashed values $W_h(A)$ by winnowing the read A , and compute the positions of their occurrence in the reference (line 4). Accordingly, $L = \{pos : h \in W_h(A) \wedge \langle h, pos \rangle \in W(B)\}$. Next, we sort the array L to process all the positions in ascending order. If B_i satisfies the filtering criterion, there should be at least m entries in L with values between $[i, i + |A|]$. It also implies that m consecutive entries should exist in L with positional difference between the first and m^{th} entry being $< |A|$. This criterion is efficiently evaluated for all B_i using a linear scan on L (lines 6–9). If satisfied, we push the associated candidate range into T . To avoid reporting B_i more than once, we merge two consecutive overlapping tuple ranges into one.

Stage 2: Evaluation of each tuple $\langle x, y \rangle$ in the Stage 1 output array T requires computing $\mathcal{J}(A, B_i) \forall i, x \leq i \leq y$. Accordingly, we compute the minimum s unique sketch elements within $W_h(A) \cup W_h(B_i)$, and count the ones shared

Algorithm 1. Stage 1 of mapping read

Input: read A , reference index map \mathcal{H} (hash k -mer $\rightarrow pos[]$), s, τ

Output: list T of candidate regions in the reference

```

1  $m = \lceil s \cdot \tau \rceil$ 
2  $T = L = []$ 
3 for  $e \in W_h(A)$  do
4    $L.append(\mathcal{H}(e))$ 
5 sort( $L$ )
6 for  $i \leftarrow 0$  to  $|L| - m$  do
7    $j \leftarrow i + (m - 1)$ 
8   if  $(L[j] - L[i]) < |A|$  then
9      $T.append(\langle L[j] - |A| + 1, L[i] \rangle)$ 

```

Algorithm 2. Stage 2 of mapping read

Input: index \mathcal{M} , Stage 1 output T, s, τ

Output: \mathcal{P}

```

1  $\mathcal{L}_0 = \mathcal{L} = \{\}$ ,  $\mathcal{L}_0.insert(W_h(A))$ 
2 for  $\langle x, y \rangle \in T$  do
3    $i \leftarrow x, j \leftarrow x + |A|$ ,  $\mathcal{L} \leftarrow \mathcal{L}_0$ 
4    $\mathcal{L}.insert(\text{getMinimizers}(i, j))$ 
5   if  $\mathcal{J} = \text{solveJaccard}(\mathcal{L}) \geq \tau$  then
6      $\mathcal{P}.append\langle i, \mathcal{J} \rangle$ 
7   while  $i \leq y$  do
8      $\mathcal{L}.delete(\text{getMinimizers}(i, i + 1))$ 
9      $\mathcal{L}.insert(\text{getMinimizers}(j, j + 1))$ 
10    if  $\mathcal{J} = \text{solveJaccard}(\mathcal{L}) \geq \tau$  then
11       $\mathcal{P}.append\langle i, \mathcal{J} \rangle$ 
12     $i \leftarrow i + 1, j \leftarrow j + 1$ 
13 Function  $\text{getMinimizers}(p, q)$ 
14    $\text{return } \{h : \langle h, pos \rangle \in W(B), p \leq pos < q\}$ 
15 Function  $\text{solveJaccard}(\mathcal{L})$ 
16    $shared\_sketch = \sum_{k=0}^{s-1} \mathcal{L}[k]$ 
17    $\text{return } \mathcal{J} = shared\_sketch/s$ 

```

between A and B_i . We show the step-by-step procedure in Algorithm 2. We use \mathcal{L} to contain the minimizer hashed values $\{h \in W_h(A) \cup W_h(B_i)\}$. To implement \mathcal{L} , we make use of the C++ ordered map data structure that supports logarithmic time insertion, deletion and linear time iteration over unique ordered keys. We keep the hashed value as the map's key, and map it to 1 if it appears in both the reference and the read, and 0 otherwise. For each tuple $\langle x, y \rangle$, we begin by saving the hashed values $W_h(A)$ in read A into map \mathcal{L} (lines 1, 3). Two loops (lines 2, 7) evaluate each tuple $\langle x, y \rangle$ in T , and consider each B_i , $x \leq i \leq y$ for Jaccard estimation $\mathcal{J}(A, B_i)$. The function `getMinimizers` gathers the reference minimizer hashes $W_h(B_i)$ by sequentially iterating over \mathcal{M} in the required position range and populating the minimizers associated with each B_i into the map \mathcal{L} (lines 4, 8-9). Note that a few incorrect corner minimizers $\{h : \langle h, pos \rangle \in W(B), i \leq pos \leq i + |A| \} \setminus W_h(B_i)$ can appear in \mathcal{L} that were winnowed from windows overlapping with B_i . However, these can be discarded by recomputing the minimum of the first and last window of B_i . Finally, function `solveJaccard` computes $|S(W(A) \cup W(B_i)) \cap S(W(A)) \cap S(W(B_i))|$ by

iterating over s minimum unique sketch elements and counting the ones shared between A and B_i . If $\mathcal{J}(A, B_i) \geq \tau$, then the position i and Jaccard estimate $\mathcal{J}(A, B_i)$ are saved into the output \mathcal{P} as pair $(i, \mathcal{J}(A, B_i))$. The corresponding estimate of the alignment error rate ϵ in this case, computed using Eq. 1, would be $\mathcal{F}(\mathcal{J}(A, B_i), k)$.

5 Selecting Window and Sketch Sizes

The sketch size for Jaccard similarity estimation is inversely proportional to the window size w (Sect. 4.3). A larger window size improves the runtime and space requirement during the search but also negatively affects the statistical significance and accuracy of our estimate. To achieve the right balance, we analyze the p-value of a mapping location being reported under the null hypothesis that both query and reference sequences are random. For the subsequent analysis, we will assume the sketch size is s , the count of shared sketch elements is a discrete random variable Z , the k -mer size is k , the alphabet set is Σ , and the read and reference sequence sizes are q and r respectively.

Location i is reported if $\mathcal{J}(A, B_i) \geq \tau$, i.e., at least $\lceil s \cdot \tau \rceil$ sketch elements are shared. Following [19], consider two random sequences of length q with k -mer sets X and Y respectively. The probability of a random k -mer α appearing in X or Y , assuming $q \gg k$, is $P(\alpha \in X) = P(\alpha \in Y) = 1 - (1 - |\Sigma|^{-k})^q$. Therefore, the expected Jaccard similarity $J_{null} = J(X, Y)$ is given by

$$J_{null} = \frac{P(\alpha \in X \cap Y)}{P(\alpha \in X \cup Y)} = \frac{P(\alpha \in X) \cdot P(\alpha \in Y)}{P(\alpha \in X) + P(\alpha \in Y) - P(\alpha \in X) \cdot P(\alpha \in Y)}$$

For sketch size s , the probability that x or more sketch elements are shared is $P(Z \geq x | J_{null}, s) = \sum_{j=x}^s \binom{s}{j} (J_{null})^j (1 - J_{null})^{s-j}$. Using this equation, we compute the probability of a random sequence of length q mapping to at least one substring in a random reference sequence of size $r \gg q$ as $1 - (1 - P(Z \geq x | J_{null}, s))^r$. For a minimum read length l_0 and $x = \lceil s \cdot \tau \rceil$, we wish to ensure that this probability is kept below a user-specified threshold p_{max} . As reported mapping locations i must satisfy $\mathcal{J}(A, B_i) \geq \tau$ and $q \geq l_0$, a mapping with $\mathcal{J}(A, B_i) = \tau, q = l_0$, in general, will have the highest probability of generating a random match. Therefore, we compute the maximum value of w that satisfies the p_{max} constraint for this instance. Sketch size s is set to $|W_h(A)|$, which from Sect. 4.3 is expected to be $q \cdot 2/w$. Since x, s , and w have a circular dependency, we iteratively solve for w , starting from the maximum value l_0 , until the probability of a random mapping is $\leq p_{max}$. Influence of different parameters on window size is shown in Fig. 3. The window size w increases with increasing p_{max} or l_0 , but has an inverse relationship with ϵ_{max} . These plots also highlight that as read length and error rate improve, our algorithm automatically adapts to a larger window size, greatly improving efficiency.

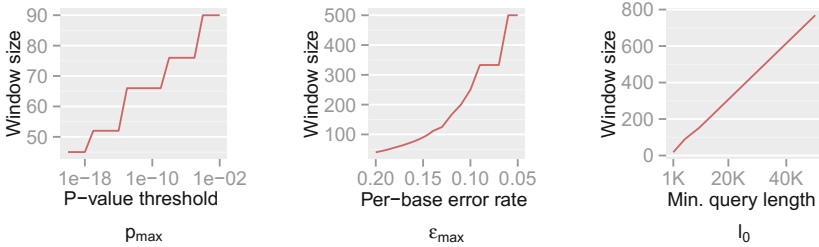


Fig. 3. Illustration of how w varies with p_{max} , ϵ_{max} , and l_0 , respectively. The default values are set as $l_0 = 5000$, $\epsilon_{max} = 0.15$, $p_{max} = 0.001$, $k = 16$, and $r = 10^9$. Steps appear in the first two curves because Z is a discrete variable.

6 Proof of Sensitivity

We analyze the sensitivity exhibited by our algorithm in identifying correct mapping locations as a function of the read alignment error rate. Let i be a correct mapping location for read A . If ϵ_{true} is the true error rate in aligning A with B_i , then $J_{true} \approx \mathcal{G}(\epsilon_{true}, k)$. Our algorithm reports this mapping location if the Jaccard estimate $\mathcal{J}(A, B_i) \geq \tau$, i.e., the count of shared sketch elements $Z \geq s \cdot \tau$. The associated probability is given by $P(Z \geq s \cdot \tau | J_{true}, s) \approx \sum_{j=\lceil s \cdot \tau \rceil}^s \binom{s}{j} (J_{true})^j (1 - J_{true})^{s-j}$. We report the corresponding values in Table 1 while varying ϵ_{max} and ϵ_{true} from 0.04 to 0.20 error rate, for two sketch sizes $s = 200$ and 500, respectively. In an ideal scenario, a mapping should be reported only if $\epsilon_{true} \leq \epsilon_{max}$, i.e., a perfect algorithm would have “1” in each of the entries at or above the diagonal, and “0” in all other positions. From the table, it is evident our algorithm achieves close to ideal sensitivity for alignment error rates up to 20%.

Table 1. Probability of a mapping location being reported by our algorithm for different values of ϵ_{true} and ϵ_{max} . True mapping locations correspond to $\epsilon_{true} \leq \epsilon_{max}$, i.e., entries at or above the diagonal in the tables. Sketch sizes are set to 200 and 500 for the left and right tables, respectively. The k -mer size k is set to 16.

ϵ_{true}	ϵ_{max}				
	0.04	0.08	0.12	0.16	0.20
0.04	0.951	1	1	1	1
0.08	0	0.937	1	1	1
0.12	0	0.016	0.925	1	1
0.16	0	0	0.184	0.907	0.997
0.20	0	0	0.003	0.403	0.922

ϵ_{true}	ϵ_{max}				
	0.04	0.08	0.12	0.16	0.20
0.04	0.939	1	1	1	1
0.08	0	0.949	1	1	1
0.12	0	0	0.937	1	1
0.16	0	0	0.013	0.904	1
0.20	0	0	0	0.104	0.896

7 Other Implementation Details

Optimizing for Variable Read Lengths: In contrast to cyclic short-read sequencing, single-molecule technologies can generate highly variable read lengths (e.g. 10^2 – 10^5 bases). Previously, we discussed how the window size w is determined using the minimum read length l_0 in Sect. 5. From Fig. 3(c), notice that we can further reduce the sampling rate (i.e. use a larger window size) for reads longer than l_0 while still satisfying the p-value constraint. However, to realize this, the sampling scheme for indexing the reference sequence B needs to be consistent with that of query. We propose the idea of *multilevel winnowing* to further optimize the runtime of our algorithm by choosing custom window size for each input read. Suppose $W_w(B)$ denotes the set of winnowed fingerprints in the reference computed using window size w , then $W_{2w}(B) \subseteq W_w(B)$ [25]. We exploit this property to construct a multilevel reference index with multiple window sizes $\{w, 2w, 4w \dots\}$ recursively. This optimization yields us faster mapping time per base pair for reads longer than l_0 as we independently compute the window size for a given read length $l \geq l_0$, and round it to the closest smaller reference window size $\{w, 2w, 4w \dots\}$. The expected time and space complexity to index the reference using multiple levels is unaffected because the expected size of $W_{2^{x+1}w}(B)$ is half of $W_{2^xw}(B)$ and $W_{2^{x+1}w}(B)$ can be determined in linear time from $W_{2^xw}(B)$.

Strand Prediction: To account for the reads sequenced from the reverse strand relative to the reference genome, we compute and store only canonical k -mers, i.e. the lexicographically smaller of the forward and reverse-complemented k -mer. For each k -mer tuple $\langle k, i \rangle$ in $W(A)$ and $W(B)$, we append a strand bit 1 if the forward k -mer is lexicographically smaller and -1 otherwise. While evaluating the read mappings in Stage 2, we compute the mapping strand of the read through a consensus vote among the shared sketches using sum of pairwise products of the strand bits.

8 Experimental Results

8.1 Quality of Jaccard Estimation

We first show that the accuracy of the *winnowed-minhash* estimator \mathcal{J} to estimate the Jaccard similarity is as good as the direct MinHash approximation, which is an unbiased statistical estimator. We construct a random sequence of length 5 kbp with each character having equal probability of being either A, C, G or T. We generate reads while introducing substitution errors at each position with probability 0.15. Note that both substitutions and indels have a similar effect of altering the k -mers containing them, and a uniform distribution of errors alters more k -mers than a clustering of errors. Figure 4 shows the estimation difference against the true Jaccard similarity using MinHash and our estimator for two different sketch sizes $s = 100$ and $s = 200$. Based on these results, we conclude that the bias in our estimation is practically negligible as the mean error

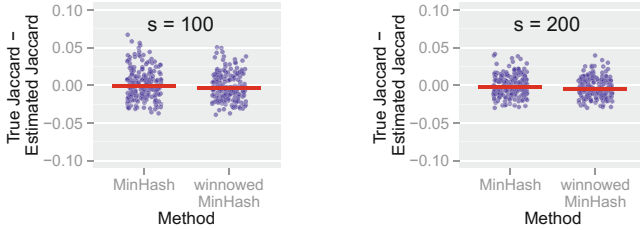


Fig. 4. Jaccard similarity estimation using MinHash and *winnowed-minhash* estimator $\mathcal{J}(A, B_i)$ over simulated reads, with sketch sizes $s = 100$ and $s = 200$. Red bar indicates the average estimation difference over all reads.

by our method in estimating Jaccard similarity is < 0.003 for both sketch sizes. Similar to MinHash approximation, we note that the magnitude of estimation error reduces with increasing sketch size.

8.2 Mapping MinION and PacBio Reads

We refer the C++ implementation of our algorithm as *mashmap* and compare its run-time performance and memory usage against alignment based long-read mappers BWA-MEM (v0.7.15-r114) [13], BLASR (vSMRTportal 2.3.0) [5], and *minimap* (v0.2) [14]. We also perform a comparison of the approximate mapping targets generated by *mashmap* and *minimap*. Like *mashmap*, *minimap* uses winnowing to index the reference, but does not use the MinHash approximation to estimate Jaccard similarity or nucleotide identity. Instead, *minimap* seeks clusters of minimizer matches to identify regions of local similarity. Importantly, *minimap* approximates a local alignment process, which is useful for split-read mapping. However, because *mashmap* is currently designed to find complete read mappings, we only consider this case for the following comparisons.

Datasets and Methodology: We evaluated the algorithms by mapping long read datasets generated using single-molecule sequencers from Pacific Biosciences and Oxford Nanopore, and report single-threaded execution timings on an AMD Opteron 2376 CPU with 64 GB RAM. We use two datasets, labeled N1 and P1 respectively, both containing reads of length ≥ 5 kbp. Dataset N1 is a random sample of 30,000 reads from the MinION (R9/1D) sequencing dataset of the *Escherichia coli* K12 genome [17]. Dataset P1 contains 18,000 reads generated through a single SMRT cell from PacBio’s (P6/C4) sequencing of the human genome (CHM1) [6]. We map N1 to *E. coli* K12 (4.6 Mbp) and P1 to the human reference (3.2 Gbp). For *mashmap*, we use the following parameters: $l_0 = 5000$, $\epsilon_{max} = 0.15$, and $p_{max} = 0.001$. When a read maps to multiple locations, *mashmap* only reports locations where mapping error rate is no more than 1% above the minimum of error rate over all such locations.

Run-Time Performance: Run-times for the index building and mapping stages, and memory used, for the four methods are compared in Table 2.

Table 2. Runtime and memory usage comparison of mashmap against minimap, BWA-MEM and BLASR for N1, P1 datasets. BWA-MEM was executed with long read mapping parameters `-x pacbio/ont2d`.

Method	N1 (MinION-K12)			P1 (Pacbio-CHM1)		
	Index	Map	Memory (MB)	Index	Map	Memory (GB)
mashmap	0.5 s	54 s	17	5 m 52 s	1 m 24 s	3.7
minimap	0.7 s	37 s	232	3 m 7 s	1 m 56 s	6.8
BWA-MEM	2.6 s	5 h 39 m	72	1 h 19 m	6 h 46 m	5.5
BLASR	1.3 s	10 h 17 m	697	40 m 36 s	20 h 40 m	17.6

As both BWA-MEM and BLASR are alignment based methods, we expect their run-times to be significantly higher. Indeed, they take several hours in comparison to seconds (N1) or a few minutes (P1) taken by mashmap and minimap. The principal challenge is whether the latter methods can retain the quality obtainable through alignment based methods. We note that mashmap has the lowest memory footprint for both datasets, and its run-time compares favorably with minimap. The ability to compute the sampling rate at runtime gives mashmap its edge in terms of memory usage.

Quality of Mapping: As there is no standard benchmark using real datasets, we assess sensitivity/recall using BWA-MEM’s starting read mapping positions, and precision by computing Smith-Waterman (SW) alignments of the reported mappings (Table 3). Since both minimap and BWA-MEM also report split-read alignments, we post-filter their results to only keep alignments with $\geq 80\%$ read coverage. Recall is measured against BWA-MEM alignments which satisfy the $\epsilon_{max} = 0.15$ cutoff ($\geq 85\%$ identity). Because both minimap and mashmap estimate mapping positions, the reported mapping is assumed equivalent to BWA-MEM if the predicted starting position of a read is within $\pm 50\%$ of its length. Precision was directly validated using Smith-Waterman (SW) alignment (with scoring matrix: $match = 1$, $mismatch = -1$, $gap_{open} = -2$, $gap_{extend} = -1$). For minimap’s and our results, we allow SW-identity $\geq 75\%$ and query coverage $\geq 80\%$. Results in Table 3 show that both mashmap and minimap have close to ideal sensitivity/recall, demonstrating their ability to uncover the right target locations.

Table 3. Precision and recall statistics of mashmap and minimap using datasets N1 and P1.

Id	Recall statistics			Precision statistics	
	mashmap	minimap	#BWA mappings	mashmap	minimap
N1	100%	99.87%	10,823	94.39%	94.32%
P1	96.8%	98.7%	10,115	84.59%	30.34%

Mashmap also achieves high precision, avoiding false positives on the repetitive human genome. Minimap’s low precision on human is largely driven by false-positive mappings to repetitive sequence, which could potentially be resolved with alternative clustering parameters. Mashmap false positives are dominated by reported mappings with a SW query coverage less than 80% of the read length. It may be possible to avoid such mappings by considering the positional distribution of shared sketch elements during the second stage filter, or by adopting a local alignment reporting strategy like minimap.

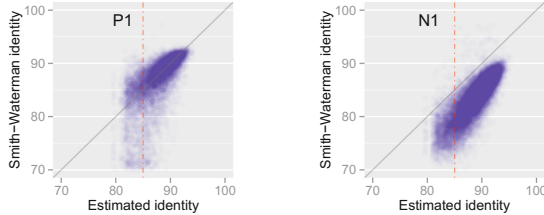


Fig. 5. Correlation between Smith-Waterman identity and the identity estimated by mashmap using datasets P1 (PacBio) and N1 (MinION). Red dotted line corresponds to the error cut-off $\epsilon_{max} = 0.15$.

We compare our identity estimates $(1 - \epsilon) \times 100$ against the SW alignment identities in Fig. 5. For the PacBio reads, we observe that most of the points are aligned close to $y = x$. However, for the nanopore reads, our approach overestimates the identity. This is because PacBio sequencing produces mostly random errors, whereas current nanopore errors are more clustered and systematic [11].

8.3 Mapping to RefSeq

We perform mapping of a publicly available PacBio read set consisting of 127,565 reads (each ≥ 5 kbp) sequenced from a mock microbial community containing 20 strains [20]. To demonstrate the scalability of our algorithm, we map these reads against the complete NCBI RefSeq database (838 Gbp) containing sequences from 60,892 organisms. This experiment was executed using default parameters ($l_0 = 5000$, $\epsilon_{max} = 0.15$, $p_{max} = 0.001$) on an Intel Xeon CPU E7-8837 with 1 TB memory. BWA-MEM and minimap could not index the entire RefSeq database at once with this memory limitation. Mashmap took 29 CPU hours to index the reference and 16 CPU hours for mapping, with a peak memory usage of 660 GB. Note that the same index can be repeatedly used for mapping sequences, conferring our method the ability to process data in real-time. To check the accuracy of our results, we ran BWA-MEM against the 20 known genomes of the mock community. The recall of mashmap against BWA-MEM mappings ranged from 97.7% to 99.1% for all the 20 genomes in the mock community.

9 Conclusions

We have presented a fast approximate algorithm for mapping long reads to large reference genomes. Instead of reporting base-level alignments, mashmap reports all reference intervals with sufficient Jaccard similarity compared to the k -mer spectrum of the read. In contrast to earlier techniques based on MinHash and winnowing, we provide a formal characterization of the mappings the algorithm is intended to uncover, and provide a provably good algorithm for computing them. In addition, we report an estimate of the alignment error rate tailored to each mapping under an assumed error model. Mashmap provides significant benefits in run-time, memory usage, and scalability, while achieving precision and recall similar to alignment-based methods. Future work aims to extend this method to split-read mapping, compressed reference databases, and additional error models. For example, the *winnowed-minhash* operation could be applied to paths within a de Bruijn graph to recover identity estimates and identify the database sequences most similar to a query sequence. Such approximate algorithms promise to help address the ever increasing scale of genomic data.

Acknowledgments. This research was supported in part by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health, and the U.S. National Science Foundation under IIS-1416259.

References

1. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.* **25**(17), 3389–3402 (1997)
2. Ashton, P.M., Nair, S., Dallman, T., Rubino, S., Rabsch, W., Mwaigwisya, S., Wain, J., O’Grady, J.: MinION nanopore sequencing identifies the position and structure of a bacterial antibiotic resistance island. *Nat. Biotechnol.* **33**(3), 296–300 (2015)
3. Berlin, K., Koren, S., Chin, C.S., Drake, J.P., Landolin, J.M., Phillippy, A.M.: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat. Biotechnol.* **33**(6), 623–630 (2015)
4. Broder, A.Z.: On the resemblance and containment of documents. In: *Proceedings of Compression and Complexity of Sequences 1997*, pp. 21–29. IEEE (1997)
5. Chaisson, M.J., Tesler, G.: Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinf.* **13**(1), 238 (2012)
6. Chaisson, M.J., Huddleston, J., Dennis, M.Y., Sudmant, P.H., Malig, M., Hormozdiari, F., Antonacci, F., Surti, U., Sandstrom, R., Boitano, M., et al.: Resolving the complexity of the human genome using single-molecule sequencing. *Nature* **517**(7536), 608–611 (2015)
7. Chin, C.S., Alexander, D.H., Marks, P., Klammer, A.A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E.E., et al.: Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat. Methods* **10**(6), 563–569 (2013)

8. Delcher, A.L., Phillippy, A., Carlton, J., Salzberg, S.L.: Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.* **30**(11), 2478–2483 (2002)
9. Fan, H., Ives, A.R., Surget-Groba, Y., Cannon, C.H.: An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16**(1), 1 (2015)
10. Koren, S., Harhay, G.P., Smith, T.P., Bono, J.L., Harhay, D.M., Mcvey, S.D., Radune, D., Bergman, N.H., Phillippy, A.M.: Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biol.* **14**(9), 1 (2013)
11. Laehnemann, D., Borkhardt, A., McHardy, A.C.: Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Brief. Bioinf.* **17**(1), 154–179 (2016)
12. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with bowtie 2. *Nat. Methods* **9**(4), 357–359 (2012)
13. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arxiv preprint [arXiv:1303.3997](https://arxiv.org/abs/1303.3997) (2013)
14. Li, H.: Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**, btw152 (2016)
15. Li, H., Durbin, R.: Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics* **25**(14), 1754–1760 (2009)
16. Li, H., Homer, N.: A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinf.* **11**(5), 473–483 (2010)
17. Loman, N.J.: Nanopore R9 rapid run data release (2016). <https://goo.gl/UIHVtL>. Accessed 8 Sept 2016
18. Loose, M., Malla, S., Stout, M.: Real time selective sequencing using nanopore technology. *Nat. Methods* **13**(9), 751–754 (2016)
19. Ondov, B.D., Treangen, T.J., Melsted, P., Mallonee, A.B., Bergman, N.H., Koren, S., Phillippy, A.M.: Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.* **17**, 132 (2016)
20. Pacific Biosciences: Human microbiome mock community shotgun sequencing data (2014). <https://goo.gl/kjRcLb>. Accessed 8 Sept 2016
21. Popic, V., Batzoglou, S.: Privacy-preserving read mapping using locality sensitive hashing and secure kmer voting. *bioRxiv*, 046920 (2016)
22. Quick, J., Loman, N.J., Duraffour, S., Simpson, J.T., Severi, E., Cowley, L., Bore, J.A., Koundouno, R., Dudas, G., Mikhail, A., et al.: Real-time, portable genome sequencing for Ebola surveillance. *Nature* **530**(7589), 228–232 (2016)
23. Roberts, M., Hayes, W., Hunt, B.R., Mount, S.M., Yorke, J.A.: Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**(18), 3363–3369 (2004)
24. Ruffalo, M., LaFramboise, T., Koyutürk, M.: Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics* **27**(20), 2790–2796 (2011)
25. Schleimer, S., Wilkerson, D.S., Aiken, A.: Winnowing: local algorithms for document fingerprinting. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 76–85. ACM (2003)
26. Smith, K.C.: Sliding window minimum implementations (2016). <https://goo.gl/8RC54b>. Accessed 8 Sept 2016
27. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. *J. Mol. Biol.* **147**(1), 195–197 (1981)

Determining the Consistency of Resolved Triplets and Fan Triplets

Jesper Jansson^{1,2(✉)}, Andrzej Lingas³, Ramesh Rajaby^{4,5},
and Wing-Kin Sung^{4,6}

¹ Laboratory of Mathematical Bioinformatics, ICR, Kyoto University,
Gokasho, Uji, Kyoto 611-0011, Japan

jj@kuicr.kyoto-u.ac.jp

² Department of Computing, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, China

³ Department of Computer Science, Lund University, 22100 Lund, Sweden
Andrzej.Lingas@cs.lth.se

⁴ School of Computing, National University of Singapore,
13 Computing Drive, Singapore 117417, Singapore

ramesh.rajaby@gmail.com, ksung@comp.nus.edu.sg

⁵ NUS Graduate School for Integrative Sciences and Engineering, National
University of Singapore, 28 Medical Drive, Singapore 117456, Singapore

⁶ Genome Institute of Singapore, 60 Biopolis Street, Genome,
Singapore 138672, Singapore

Abstract. The $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem takes as input two sets R^+ and R^- of resolved triplets and two sets F^+ and F^- of fan triplets, and asks for a distinctly leaf-labeled tree that contains all elements in $R^+ \cup F^+$ and no elements in $R^- \cup F^-$ as embedded subtrees, if such a tree exists. This paper presents a detailed characterization of how the computational complexity of the problem changes under various restrictions. Our main result is an efficient algorithm for dense inputs satisfying $R^- = \emptyset$ whose running time is linear in the size of the input and therefore optimal.

Keywords: Phylogenetic tree · Rooted triplets consistency · Algorithm · Computational complexity

1 Introduction

Phylogenetic trees have been used by biologists for more than 150 years to describe evolutionary history. In the last 50 years, many methods for systematically reconstructing phylogenetic trees from different kinds of data have been proposed [10, 23]. In general, inferring a reliable phylogenetic tree is a time-consuming task for large data sets, but the *supertree approach* (see, e.g., [2, 3]) may in many cases provide a reasonable compromise between accuracy and computational efficiency by way of divide-and-conquer: first, infer a set of trees for small, overlapping subsets of the species using a computationally expensive method such as maximum likelihood [7, 10], and then merge all the small trees

into one big tree with some combinatorial algorithm. In this context, the fundamental problem of determining if a given set of *resolved triplets* (rooted, binary phylogenetic trees with exactly three leaf labels each) can be combined without conflicts, and if so, constructing such a tree can be solved efficiently by Aho *et al.*'s BUILD algorithm from [1]. BUILD has therefore been extended in various ways [8, 12, 13, 16, 18–21, 24], for example, to also allow *fan triplets* (rooted, non-binary phylogenetic trees with three leaf labels each) or *forbidden* resolved triplets in the input, and to handle related optimization problems where the input may contain errors and the objective is to find a tree that satisfies as much of the input as possible (for details, see [6, 9] and the references therein).

Below, we investigate how the computational complexity of the basic decision problem varies according to which types of inputs are allowed and present some new results that expose the boundary between efficiently solvable and intractable versions of the problem.

1.1 Problem Definitions

A *phylogenetic tree* is a rooted, unordered, distinctly leaf-labeled tree in which every internal node has at least two children. (From here on, phylogenetic trees are simply referred to as “trees” and every leaf in a tree is identified with its corresponding leaf label.) For any tree T , the set of all nodes in T is denoted by $V(T)$ and the set of all leaf labels occurring in T is denoted by $\Lambda(T)$. The *degree* of a node $u \in V(T)$ is the number of children of u , and the *degree* of T is the maximum degree of all nodes in $V(T)$. For any $u, v \in V(T)$, $\text{lca}^T(u, v)$ denotes the lowest common ancestor in T of u and v .

A *rooted triplet* is a tree with precisely three leaves. Let t be any rooted triplet and suppose that $\Lambda(t) = \{x, y, z\}$. If t is binary then t is called a *resolved triplet* and we write $t = xy|z$, where $\text{lca}^t(x, y)$ is a proper descendant of $\text{lca}^t(x, z) = \text{lca}^t(y, z)$. On the other hand, if t is not binary then t is called a *fan triplet* and we write $t = x|y|z$. Note that there are four different rooted triplets leaf-labeled by $\{x, y, z\}$, namely $xy|z$, $xz|y$, $yz|x$, and $x|y|z$.

For any tree T and $\{x, y, z\} \subseteq \Lambda(T)$, the resolved triplet $xy|z$ is *consistent with T* if $\text{lca}^T(x, y)$ is a proper descendant of $\text{lca}^T(x, z) = \text{lca}^T(y, z)$. Similarly, the fan triplet $x|y|z$ is *consistent with T* if $\text{lca}^T(x, y) = \text{lca}^T(x, z) = \text{lca}^T(y, z)$. Finally, for any tree T , let $T|_{\{x, y, z\}}$ be the rooted triplet with leaf label set $\{x, y, z\}$ that is consistent with T , and let $t(T)$ be the set of *all* rooted triplets (resolved triplets as well as fan triplets) consistent with T , i.e., define $t(T) = \{T|_{\{x, y, z\}} : \{x, y, z\} \subseteq \Lambda(T)\}$.

The problem studied in this paper is:

The $\mathcal{R}^+ \mathcal{F}^-$ CONSISTENCY problem:

Given two sets R^+ and R^- of resolved triplets and two sets F^+ and F^- of fan triplets over a leaf label set L , output a tree T with $\Lambda(T) = L$ such that $R^+ \cup F^+ \subseteq t(T)$ and $(R^- \cup F^-) \cap t(T) = \emptyset$, if such a tree exists; otherwise, output *null*.

In other words, R^+ and F^+ specify rooted triplets that are required to be embedded in the output tree, while R^- and F^- are forbidden rooted triplets. See Fig. 1 for two examples. Throughout the paper, we use n to denote the cardinality of the input leaf label set L .

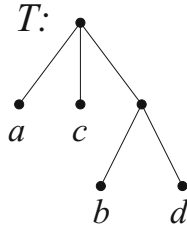


Fig. 1. As an example, consider the following instance of the $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem: $L = \{a, b, c, d\}$, $R^+ = \emptyset$, $R^- = \{cd|a\}$, $F^+ = \{a|b|c\}$, and $F^- = \{b|c|d\}$. The shown tree T satisfies $t(T) = \{a|b|c, bd|a, a|c|d, bd|c\}$, so $R^+ \cup F^+ \subseteq t(T)$ and $(R^- \cup F^-) \cap t(T) = \emptyset$ hold. Thus, T is a valid solution. As another example, if L, R^+, R^- , and F^- are the same as above but F^+ is changed to $F^+ = \{a|b|c, a|b|d\}$ then the answer is *null*.

The special cases of the $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem where one or more of the four input sets R^+, R^-, F^+, F^- are empty will also be denoted by removing the corresponding “+” and “-” symbols from the problem name. For example, the $\mathcal{R}^-\mathcal{F}^+$ CONSISTENCY problem requires that $R^+ = F^- = \emptyset$. To simplify the notation, if $R^+ = R^- = \emptyset$ then we omit the “ \mathcal{R} ”, and analogously for “ \mathcal{F} ”; e.g., \mathcal{R}^- means $R^+ = F^+ = F^- = \emptyset$. Ignoring the trivial case where all of R^+, R^-, F^+, F^- are empty, this yields exactly 14 problem variants in addition to the original problem. Our goal is to establish the computational complexity of all these problem variants as well as some other potentially useful special cases. Because of space constraints, the proofs of Lemmas 5 and 6 have been deferred to the journal version.

1.2 Overview of Old and New Results

Aho *et al.* [1] presented a polynomial-time algorithm named BUILD that solves the \mathcal{R}^+ CONSISTENCY problem, and Ng and Wormald [18] extended BUILD to solve $\mathcal{R}^+\mathcal{F}^+$ CONSISTENCY in polynomial time. Using a similar approach, He *et al.* [13] showed how to solve \mathcal{R}^{+-} CONSISTENCY in polynomial time. As for negative results, Bryant [4, Theorem 2.20] proved that \mathcal{R}^- CONSISTENCY is NP-hard under the additional constraint that the output tree is binary. Three direct consequences of these known results are given in Sect. 2 (Lemmas 1, 2, and 3). In Sect. 3, we shall prove that the \mathcal{F}^{+-} CONSISTENCY problem is NP-hard (Theorem 1). Significantly, Lemmas 1, 2, and 3 together with Theorem 1 then provide a complete characterization of the polynomial-time solvability of

all 15 variants of the $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem defined in Sect. 1.1 since each of the remaining problem variants is either a special case of a polynomial-time solvable problem variant or a generalization of an NP-hard one. See Table 1.

Table 1. Overview of the computational complexity of the 15 different variants of the $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem. “P” means solvable in polynomial time. The results written in bold text are due to [1, 13, 18].

CONSISTENCY	\emptyset	\mathcal{F}^+	\mathcal{F}^-	\mathcal{F}^{+-}
\emptyset	×	P	P	NP-hard (Theorem 1)
\mathcal{R}^+	P	P	P (Lemma 2)	NP-hard
\mathcal{R}^-	P	P	NP-hard (Lemma 3)	NP-hard
\mathcal{R}^{+-}	P	P (Lemma 1)	NP-hard	NP-hard

Motivated by these observations, we then try to identify some way of restricting the $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY problem that leads to more efficiently solvable problem variants. One natural restriction is to require the degree of the output tree to be at most D for some integer $D \geq 2$; unfortunately, Sect. 4 demonstrates that this generally makes the problems *harder*. See Table 2 for a summary. In particular, Theorem 2 proves that even \mathcal{F}^+ CONSISTENCY is NP-hard when restricted to degree- D trees for every fixed $D \geq 4$. Furthermore, by Corollary 2, D -bounded degree \mathcal{R}^- CONSISTENCY becomes NP-hard for every fixed $D \geq 2$. The only efficiently solvable problem variants that we know of are covered by Corollary 1, stating that D -bounded degree $\mathcal{R}^+\mathcal{F}^-$ CONSISTENCY remains polynomial-time solvable for every $D \geq 2$.

Table 2. The complexity of $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY when the output tree is required to have degree at most D . “NP-hard*” (with an asterisk) means NP-hard for every fixed $D \geq 4$ and trivially polynomial-time solvable for $D = 2$ while the complexity for $D = 3$ is still open.

Bounded degree CONSISTENCY	\emptyset	\mathcal{F}^+	\mathcal{F}^-	\mathcal{F}^{+-}
\emptyset	×	NP-hard* (Theorem 2)	P	NP-hard*
\mathcal{R}^+	P	NP-hard*	P (Corollary 1)	NP-hard*
\mathcal{R}^-	NP-hard (Corollary 2)	NP-hard	NP-hard	NP-hard
\mathcal{R}^{+-}	NP-hard	NP-hard	NP-hard	NP-hard

Therefore, we need to find another way to restrict the problem. For this purpose, Sect. 5 considers inputs that are *dense* in the sense that for each $L' \subseteq L$

with $|L'| = 3$, at least one rooted triplet t with $\Lambda(t) = L'$ is specified in R^+ , R^- , F^+ , or F^- . As shown in [9], the maximization version of $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY (whose objective is to output a tree T with $\Lambda(T) = L$ maximizing the value of $|T(R^+ \cup F^+)| + |(R^- \cup F^-) \setminus T(R^- \cup F^-)|$, where $T(X)$ for any set X of rooted triplets denotes the subset of X consistent with T) admits a polynomial-time approximation scheme (PTAS) when restricted to dense inputs, whereas no such PTAS is known for the non-dense case; in fact, the non-dense case of the maximization problem is APX-complete [5, Proposition 2]. This gives us some hope that $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY may be easier for dense inputs. Although $\mathcal{R}^-\mathcal{F}^-$ CONSISTENCY turns out to be NP-hard in the dense case by Lemma 3, $\mathcal{R}^+\mathcal{F}^{+-}$ CONSISTENCY restricted to dense inputs indeed admits a polynomial-time algorithm (Theorem 4), and moreover, its time complexity is $O(n^3)$ which is optimal because the size of a dense input is $\Omega(n^3)$. The situation for dense inputs is summarized in Table 3.

Table 3. The complexity of $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY restricted to dense inputs. The results written in bold text are due to [1, 13, 18].

Dense CONSISTENCY	\emptyset	\mathcal{F}^+	\mathcal{F}^-	\mathcal{F}^{+-}
\emptyset	\times	P	P	P
\mathcal{R}^+	P	P	P	P (Theorem 4)
\mathcal{R}^-	P	P	NP-hard (Lemma 3)	NP-hard
\mathcal{R}^{+-}	P	P (Lemma 1)	NP-hard	NP-hard

2 Preliminaries

This section lists some simple results that follow immediately from previous work.

Lemma 1. *The $\mathcal{R}^{+-}\mathcal{F}^+$ CONSISTENCY problem is solvable in polynomial time.*

Proof. For any instance of $\mathcal{R}^{+-}\mathcal{F}^+$ CONSISTENCY, by removing each fan triplet of the form $x|y|z$ from F^+ and inserting the three resolved triplets $xy|z$, $xz|y$, $yz|x$ into R^- , one obtains an equivalent instance of \mathcal{R}^{+-} CONSISTENCY to which the *MTT* algorithm in [13] can be applied. By [13], the running time becomes $O(|R^+| \cdot n + (|R^-| + |F^+|) \cdot n \log n + n^2 \log n)$. \square

Lemma 2. *The $\mathcal{R}^+\mathcal{F}^-$ CONSISTENCY problem is solvable in polynomial time.*

Proof. For any instance of $\mathcal{R}^+\mathcal{F}^-$ CONSISTENCY, run the BUILD algorithm [1] with input R^+ and let T be its output. If T is not *null* then, as long as T is non-binary, select any internal node u with degree larger than two and any two children c_1 and c_2 of u , remove the edges $\{u, c_1\}$ and $\{u, c_2\}$, create a new child v of u , and insert the edges $\{v, c_1\}$ and $\{v, c_2\}$. Finally, output T . Using a

fast implementation of BUILD from [14] along with an improved data structure for supporting dynamic graph connectivity queries [15] (see [17] for details), the $\mathcal{R}^+\mathcal{F}^-$ CONSISTENCY problem becomes solvable in $\min\{O(|R^+| \cdot \log^2 n + |F^-| + n), O(|R^+| + |F^-| + n^2 \log n)\}$ time. \square

Lemma 3. *The $\mathcal{R}^-\mathcal{F}^-$ CONSISTENCY problem is NP-hard, even if restricted to dense inputs.*

Proof. According to Bryant [4, Theorem 2.20], the \mathcal{R}^- CONSISTENCY problem is NP-hard when the output tree is constrained to be binary. Given any instance of Bryant's version of the problem consisting of a set R of (forbidden) resolved triplets, construct an equivalent instance of the $\mathcal{R}^-\mathcal{F}^-$ CONSISTENCY problem by letting $R^- = R$ and letting F^- be the set of all $\binom{|L|}{3}$ fan triplets over the leaf label set $L = \bigcup_{t \in R} \Lambda(t)$ (note that F^- is dense). Since the reduction is a polynomial-time reduction, the latter problem is also NP-hard. \square

3 \mathcal{F}^{+-} CONSISTENCY is NP-Hard

Here, we prove that the \mathcal{F}^{+-} CONSISTENCY problem is NP-hard by giving a polynomial-time reduction from the NP-hard problem SET SPLITTING (see, e.g., [11]):

SET SPLITTING:

Given a set $S = \{s_1, s_2, \dots, s_n\}$ and a collection $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ of subsets of S where $|C_j| = 3$ for every $C_j \in \mathcal{C}$, does (S, \mathcal{C}) have a set splitting, i.e., can S be partitioned into two disjoint subsets S' and S'' such that for every $C_j \in \mathcal{C}$ it holds that C_j is not a subset of S' and C_j is not a subset of S'' ?

We now describe the reduction. Given an instance (S, \mathcal{C}) of SET SPLITTING, where we assume w.l.o.g. that $\bigcup_{C_j \in \mathcal{C}} C_j = S$, construct an instance of \mathcal{F}^{+-} CONSISTENCY as follows:

- Let $L = S \cup \{x, y, z', z''\} \cup \{\alpha_j, \beta_j, \gamma_j : 1 \leq j \leq m\}$ be the leaf label set.
- For $1 \leq j \leq m$, denote $C_j = \{c_j^1, c_j^2, c_j^3\}$, where $c_j^1, c_j^2, c_j^3 \in S$. Define $F^+ = \{x|y|z', x|y|z'', x|z'|z''\} \cup \{x|y|s_i : s_i \in S\} \cup \{x|c_j^1|\alpha_j, c_j^2|c_j^3|\alpha_j, x|c_j^2|\beta_j, c_j^1|c_j^3|\beta_j, x|c_j^3|\gamma_j, c_j^1|c_j^2|\gamma_j : 1 \leq j \leq m\}$.
- Define $F^- = \{s_i|z'|z'' : s_i \in S\}$.

The next lemma ensures the correctness of the reduction:

Lemma 4. *(S, \mathcal{C}) has a set splitting if and only if there exists a tree T with $\Lambda(T) = L$ such that $F^+ \subseteq t(T)$ and $F^- \cap t(T) = \emptyset$.*

Proof. \Rightarrow) Suppose that (S', S'') is a set splitting of (S, \mathcal{C}) . Create a tree T with $\Lambda(T) = L$ whose root has $4 + 2m$ children in the following way. First, let two leaves labeled by x and y as well as two internal nodes u' and u'' be children of the root of T , and attach $1 + |S'|$ leaves labeled by $\{z'\} \cup S'$ and $1 + |S''|$ leaves labeled by $\{z''\} \cup S''$ as children of u' and u'' , respectively. Next, for each $C_j \in \mathcal{C}$, exactly two of the three elements c_j^1, c_j^2, c_j^3 have the same parent in T because (S', S'') is a set splitting; let u_j be this common parent. By definition, $u_j \in \{u', u''\}$. The three leaves $\alpha_j, \beta_j, \gamma_j$ are inserted into T according to which one of these cases holds:

- c_j^1 and c_j^2 have the same parent u_j : Attach a leaf labeled by γ_j as a child of u_j and two leaves labeled by α_j, β_j as children of the root of T .
- c_j^1 and c_j^3 have the same parent u_j : Attach a leaf labeled by β_j as a child of u_j and two leaves labeled by α_j, γ_j as children of the root of T .
- c_j^2 and c_j^3 have the same parent u_j : Attach a leaf labeled by α_j as a child of u_j and two leaves labeled by β_j, γ_j as children of the root of T .

It is straightforward to verify that $F^+ \subseteq t(T)$ and $F^- \cap t(T) = \emptyset$.

\Leftarrow) Suppose that T is a tree with $\Lambda(T) = L$ such that $F^+ \subseteq t(T)$ and $F^- \cap t(T) = \emptyset$. Let $r = \text{lca}^T(x, y)$. The node r must be the root of T because (1) $x|y|q \in t(T)$ for all $q \in \{z', z''\} \cup S$ and (2) for each $\delta_j \in \{\alpha_j, \beta_j, \gamma_j\}$, $1 \leq j \leq m$, there exists an $s_i \in S$ such that $x|s_i|\delta_j \in t(T)$. Let T' (resp. T'') be the subtree of T rooted at a child of r which contains z' (resp. z''); then, $T' \neq T''$ since $x|z'|z'' \in t(T)$ and x cannot belong to T' due to $x|y|z' \in t(T)$. Furthermore, each $s_i \in S$ belongs to either T' or T'' since $s_i|z'|z'' \notin t(T)$.

Next, we show by contradiction that for every $C_j \in \mathcal{C}$, exactly one or two of the three elements c_j^1, c_j^2, c_j^3 belong to T' (and hence, that exactly one or two of the three elements belong to T''). Suppose that all three elements belong to T' . The condition $c_j^1|c_j^2|\gamma_j, c_j^1|c_j^3|\beta_j, c_j^2|c_j^3|\alpha_j \in t(T)$ implies that $\alpha_j, \beta_j, \gamma_j$ also belong to T' . But then $x|c_j^1|\alpha_j, x|c_j^2|\beta_j, x|c_j^3|\gamma_j$ cannot be consistent with T , which is impossible. In the same way, all three elements cannot belong to T'' .

In summary, selecting $S' = \Lambda(T') \cap S$ and $S'' = \Lambda(T'') \cap S$ yields a set splitting of (S, \mathcal{C}) . \square

Since the reduction can be carried out in polynomial time, Lemma 4 gives:

Theorem 1. *The \mathcal{F}^{+-} CONSISTENCY problem is NP-hard.*

4 D -Bounded Degree $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY

We now consider the computational complexity of D -bounded degree $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY, i.e., where the degree of the output tree is constrained to be at most D for some integer $D \geq 2$. First, by noting that the method in the proof of Lemma 2 always outputs a binary tree, we have:

Corollary 1. *For every fixed $D \geq 2$, the D -bounded degree $\mathcal{R}^+\mathcal{F}^-$ CONSISTENCY problem is solvable in polynomial time.*

In contrast, many other variants become NP-hard, as shown in the rest of this section.

4.1 D -Bounded Degree \mathcal{F}^+ CONSISTENCY is NP-Hard

This subsection proves that for every fixed integer $D \geq 4$, the D -bounded degree \mathcal{F}^+ CONSISTENCY problem is NP-hard. The proof relies on a simple polynomial-time reduction from the K -COLORING problem, which is NP-hard for every fixed $K \geq 3$ (see [11]):

K -COLORING:

Given an undirected, connected graph $G = (V, E)$ and a positive integer K , does G have a K -coloring, i.e., can V be partitioned into K (possibly empty) disjoint subsets V_1, V_2, \dots, V_K such that for every $\{u, v\} \in E$ it holds that $i \neq j$ where $u \in V_i$ and $v \in V_j$?

The reduction is as follows. Given an instance of $(D - 1)$ -COLORING, create an instance of D -bounded degree \mathcal{F}^+ CONSISTENCY by setting $L = V \cup \{x\}$ and $F^+ = \{x|u|v : \{u, v\} \in E\}$.

Lemma 5. *G has a $(D - 1)$ -coloring if and only if there exists a tree T with degree at most D and $\Lambda(T) = L$ such that $F^+ \subseteq t(T)$.*

Theorem 2. *For every fixed $D \geq 4$, the D -bounded degree \mathcal{F}^+ CONSISTENCY problem is NP-hard.*

4.2 D -Bounded Degree \mathcal{R}^- CONSISTENCY is NP-Hard

Bryant [4, Theorem 2.20] proved that the D -bounded degree \mathcal{R}^- CONSISTENCY problem is NP-hard for $D = 2$ by reducing from the following NP-hard problem (see, e.g., [11]):

3SAT:

Given a set U of Boolean variables and a collection $C = \{C_1, C_2, \dots, C_m\}$ of disjunctive clauses over U , each containing exactly 3 literals, is there a truth assignment for U that makes every clause in C true?

The main idea in Bryant's reduction is to represent every literal by a leaf label and define the forbidden resolved triplets so that in any valid tree, assigning true to all literals contained in one particular subtree rooted at a child of the root (and assigning false to the rest) results in a valid truth assignment. In this subsection, we adapt Bryant's proof to obtain an analogous result for the case $D = 3$ by introducing an additional leaf label x and defining a slightly more involved set of forbidden resolved triplets. More precisely, given an instance of 3SAT, we construct an instance of 3-bounded degree \mathcal{R}^- CONSISTENCY with $L = U \cup \bar{U} \cup C \cup C' \cup \{x, t, f\}$ and $R^- = R_1 \cup R_2 \cup R_3 \cup R_4$, where $\bar{U} = \{\bar{u} : u \in U\}$, $C' = \{C'_j : C_j \in C\}$, and:

- $R_1 = \{tf|x, tx|f, fx|t\}$,
- $R_2 = \{u\bar{u}|x, ux|\bar{u}, \bar{u}x|u, u\bar{u}|t, u\bar{u}|f : u \in U\}$,

- $R_3 = \{C_j C'_j | x, C_j x | C'_j, C'_j x | C_j, C_j C'_j | t, C_j C'_j | f : C_j \in C\}$, and
- $R_4 = \{u_j v_j | C_j, w_j C_j | t : C_j \in C\}$, where we write $C_j = (u_j \vee v_j \vee w_j)$ with $u_j, v_j, w_j \in U \cup \bar{U}$.

Note that R_4 is defined asymmetrically.

Lemma 6. *There is a truth assignment for U making every clause in C true if and only if there exists a tree T with degree at most 3 and $\Lambda(T) = L$ such that $R^- \cap t(T) = \emptyset$.*

Theorem 3. *For $D = 3$, the D -bounded degree \mathcal{R}^- CONSISTENCY problem is NP-hard.*

Corollary 2. *For every fixed $D \geq 2$, the D -bounded degree \mathcal{R}^- CONSISTENCY problem is NP-hard.*

Proof. For $D \in \{2, 3\}$, see above. For $D \geq 4$, the NP-hardness follows from Theorem 2 and the polynomial-time reduction which, for each fan triplet of the form $x|y|z$ in F^+ in any given instance of the D -bounded degree \mathcal{F}^+ CONSISTENCY problem, includes three resolved triplets $xy|z, xz|y, yz|x$ in R^- . \square

5 An Optimal Algorithm for Dense $\mathcal{R}^+ \mathcal{F}^{+-}$ CONSISTENCY

Recall from Sect. 1.2 that an input to $\mathcal{R}^{+-} \mathcal{F}^{+-}$ CONSISTENCY is called *dense* if, for every $L' \subseteq L$ with $|L'| = 3$, at least one rooted triplet t with $\Lambda(t) = L'$ is in R^+, R^-, F^+ , or F^- . In this section, we present the main result of the paper, namely an algorithm called **DenseBuild** that solves the special case $\mathcal{R}^+ \mathcal{F}^{+-}$ CONSISTENCY (i.e., where $R^- = \emptyset$) restricted to dense inputs, and show that its running time is $O(n^3)$, which is optimal. Two tools used by **DenseBuild** are the *fan graph* and the *clique graph*, defined and studied in Sect. 5.1. Algorithm **DenseBuild** is presented in Sect. 5.2.

According to Sect. 1.2, $\mathcal{R}^+ \mathcal{F}^{+-}$ CONSISTENCY is NP-hard. Intuitively, the problem becomes easier for dense inputs because if T is a tree consistent with the input then the set $Z = \{x|y|z : x, y, z \in L \text{ and } x, y, z \text{ belong to three different subtrees attached to the root of } T\}$ forms a subset of F^+ , in which case F^+ contains enough information to partition L into the leaf label sets of the subtrees rooted at the children of the root of T (see Lemma 7). Moreover, such a partition can be computed in polynomial time using Lemmas 8–10. In contrast, when the input is not dense or when one considers dense $\mathcal{R}^{+-} \mathcal{F}^{+-}$ CONSISTENCY, not all of Z may appear in the input F^+ .

5.1 The Fan Graph and the Clique Graph

Consider any $L' \subseteq L$. Define $R^+|L' = \{t \in R^+ : \Lambda(t) \subseteq L'\}$, $F^+|L' = \{t \in F^+ : \Lambda(t) \subseteq L'\}$, and $F^-|L' = \{t \in F^- : \Lambda(t) \subseteq L'\}$. The *fan graph* $\mathcal{G}_{L'}$ is the undirected graph (L', E') , where for any $x, y \in L'$, it holds that $\{x, y\} \in E'$ if and only if $x|y|z \in F^+|L'$ for some $z \in L'$.

If T is a tree that is consistent with the input then the degree of the root of T can be determined from \mathcal{G}_L as follows:

Lemma 7. *Suppose $|L| \geq 3$ and that there exists a tree T that is consistent with the input. Let p be the degree of the root of T , let C_1, C_2, \dots, C_m be the connected components of \mathcal{G}_L , and let $\Lambda(C_i)$ for each $i \in \{1, 2, \dots, m\}$ be the set of vertices in C_i . The following holds:*

1. *If $m \geq 2$ then $p = 2$. Furthermore, if S' is any binary tree with m leaves and for each $i \in \{1, 2, \dots, m\}$, S_i is a tree with $\Lambda(S_i) = \Lambda(C_i)$ such that $(F^+|\Lambda(C_i)) \subseteq t(S_i)$ and $(F^-|\Lambda(C_i)) \cap t(S_i) = \emptyset$, then the tree S obtained by replacing the m leaves in S' by the trees in $\{S_i : 1 \leq i \leq m\}$ satisfies $F^+ \subseteq t(S)$ and $F^- \cap t(S) = \emptyset$.*
2. *If $m = 1$ then $p \geq 3$. Furthermore, the value of p and the partition of L into subsets L_1, L_2, \dots, L_p are unique, where each L_i is the leaf label set of a subtree rooted at a child of the root of T .*

Proof.

1. First we show that $p = 2$ by contradiction. Suppose $p \geq 3$ and let x, y , and z be any three leaves from three different subtrees rooted at the children of the root of T . Since the input is dense, at least one rooted triplet t with $\Lambda(t) = \{x, y, z\}$ is specified in R^+ , F^+ , or F^- ; by the choice of x, y, z , it has to be $x|y|z$. But then the edges $\{x, y\}$, $\{x, z\}$, and $\{y, z\}$ are in \mathcal{G}_L so x, y , and z belong to the same connected component C_i . By repeating the argument, every leaf in L belongs to C_i , which contradicts $m \geq 2$.

Next, consider any two connected components C_i and C_j in \mathcal{G}_L . By the definition of \mathcal{G}_L , there is no fan triplet in F^+ with leaves belonging to both C_i and C_j . Hence, F^+ equals $\bigcup_{i=1}^m (F^+|\Lambda(C_i))$. By the definition of S , $\bigcup_{i=1}^m (F^+|\Lambda(C_i)) \subseteq t(S)$. Finally, since S is binary, $F^- \cap t(S) = F^- \cap (\bigcup_{i=1}^m t(S_i)) = \bigcup_{i=1}^m ((F^-|\Lambda(C_i)) \cap t(S_i)) = \emptyset$.

2. To prove that $p \geq 3$, suppose on the contrary that $p = 2$. Let A and B be the two sets of leaves in the subtrees rooted at the two children of the root of T . Since \mathcal{G}_L is connected, there exists some $a \in A$ and $b \in B$ such that $\{a, b\}$ is an edge of \mathcal{G}_L . By the definition of \mathcal{G}_L , there exists some $c \in L$ where $a|b|c \in F^+$. However, this is impossible since $p = 2$. This gives $p \geq 3$.

Next, we prove the uniqueness of the partition of L by contradiction. Suppose that T_1 and T_2 are two trees with $F^+ \subseteq t(T_1)$, $F^+ \subseteq t(T_2)$, and $F^- \cap t(T_1) = F^- \cap t(T_2) = \emptyset$ and that the partitions of L induced by the children of the root of T_i are different for $i = 1$ and $i = 2$. For $i \in \{1, 2\}$, denote the root of T_i by r_i . We claim that there exist $x, y, z \in L$ such that for some $i \in \{1, 2\}$: (1) x, y appear in the same subtree rooted at a child of r_i and z in another such subtree; and (2) x, y, z appear in three different subtrees rooted at the children of r_{3-i} . To prove the claim, for some $i \in \{1, 2\}$, take any two leaves x and y in the same subtree D_i rooted at a child of r_i but in different subtrees D_{3-i}, D'_{3-i} rooted at a child of r_{3-i} . Without loss of generality, assume $i = 1$. If there exists a leaf z in another subtree D'_1 rooted at a child of r_1 and z belongs to a subtree D'_2 rooted at a child of r_2 different from D_2 and D'_2 then we are done. Otherwise, all leaves not in D_2 or D'_2 also appear in D_1 and we let a be any such leaf; moreover, all leaves not in D_1

appear in either D_2 or D'_2 and we let b be any such leaf, and then define w as follows: (i) $w = x$ if b and y are in the same subtree rooted at a child of r_2 , and (ii) $w = y$ if b and x are in the same subtree. The three leaves a , b , and w then satisfy the claim. Since the claim is true, $x|y|z \notin t(T_i)$ while $x|y|z \in t(T_{3-i})$. This means that if $x|y|z \in F^+$ then $F^+ \subseteq t(T_i)$ is false, if $x|y|z \in F^-$ then $F^- \cap t(T_{3-i})$ is false, and if one of $xy|z$, $xz|y$, and $yz|x$ is in R^+ then $R^+ \subseteq t(T_{3-i})$ is false, giving a contradiction in every case. \square

The three lemmas below will be used by `DenseBuild` in Sect. 5.2 to construct the partition in Lemma 7.2. In the rest of this subsection, assume that \mathcal{G}_L contains a single connected component and that there exists a tree T that is consistent with the input. For every $a, b \in L$, define $f(a, b) = |\{z : a|b|z \in F^+\}|$.

Lemma 8. *If $a, b \in L$ are any two leaves that maximize the value of $f(a, b)$, i.e., $f(a, b) = \max_{x, y \in L} f(x, y)$, then a and b belong to the smallest and second smallest subtrees T_a and T_b rooted at children of the root of T (with ties broken arbitrarily). Also, $f(a, b) = |\Lambda(T)| - |\Lambda(T_a)| - |\Lambda(T_b)|$.*

Proof. Consider any $x, y \in L$ and define $s = lca^T(x, y)$. For any $x|y|z \in F^+$, we have $lca^T(x, y) = lca^T(x, z) = lca^T(y, z) = s$, which means that $z \in \Lambda(T[s]) \setminus (\Lambda(T[s_x]) \cup \Lambda(T[s_y]))$, where $T[u]$ for any node u in T denotes the subtree of T rooted at u and s_x (resp., s_y) is the child of s that is an ancestor of x (resp., y). Thus, $f(x, y) = |\Lambda(T[s])| - |\Lambda(T[s_x])| - |\Lambda(T[s_y])|$.

Next, according to Lemma 7.2, since \mathcal{G}_L consists of one connected component, T has at least three subtrees attached to the root. To maximize the value of $f(x, y)$, we therefore choose s to be the root of T and $T[s_x]$ and $T[s_y]$ to be the smallest and second smallest subtrees attached to s . The lemma follows. \square

Lemma 9. *Let $a, b \in L$ be two leaves that maximize the value of $f(a, b)$. Define $L' = \{a, b\} \cup \{x \in L : a|b|x \notin F^+\}$ and take any $z \notin L'$. Then the leaf label sets of the two smallest subtrees attached to the root of T are $A = \{a\} \cup \{x \in L' : a|x|z \notin F^+\}$ and $B = L' \setminus A = \{b\} \cup \{x \in L' : b|x|z \notin F^+\}$.*

Proof. By Lemma 8, a and b appear in the two smallest subtrees T_a and T_b attached to the root of T . For every leaf label x in T_a or T_b , $a|b|x \notin F^+$. Thus, $L' = \Lambda(T_a) \cup \Lambda(T_b)$. Since $z \notin L'$, z is not in the subtrees containing a and b . On the other hand, for every leaf x in T_a with $x \neq a$, we have $x \in L'$ and therefore $a|x|z \notin F^+$. Hence, $\Lambda(T_a) = \{a\} \cup \{x \in L' : a|x|z \notin F^+\}$. In the same way, $\Lambda(T_b) = \{b\} \cup \{x \in L' : b|x|z \notin F^+\}$. \square

Finally, suppose that a , b , and L' are defined as in Lemma 9. Let c be either one of a and b . The *clique graph* \mathcal{Q}_L is the undirected graph (L'', E'') , where $L'' = L \setminus L'$ and $\{x, y\} \in E''$ if and only if $c|x|y \notin F^+$. The clique graph has the following useful properties:

Lemma 10. *Let C be any connected component in \mathcal{Q}_L . Then C forms a complete graph. Also, the set of vertices in C equals the set of leaves in some subtree attached to the root of T .*

Proof. Let T_c be the subtree rooted at a child of the root of T that contains c . Consider any $x \in L \setminus L'$ and note that x cannot appear in T_c . For any $y \in L \setminus L'$, if $c|x|y \in F^+$ then x and y have to be in two different subtrees attached to the root of T . Conversely, for any $x, y \in L \setminus L'$ in the same subtree attached to the root of T , we have $c|x|y \notin F^+$. Therefore, the set of leaves in each subtree attached to the root of T induce a complete subgraph in \mathcal{Q}_L . \square

5.2 Algorithm DenseBuild

We now develop an efficient algorithm for $\mathcal{R}^+ \mathcal{F}^{+-}$ CONSISTENCY restricted to dense inputs. The algorithm is named **DenseBuild** and its pseudocode is summarized in Fig. 2. (Refer to Sect. 5.1 for the notation defined there.) The basic strategy is to use the information contained in R^+ , F^+ , and F^- to partition the leaf label set L into subsets corresponding to the leaf label sets of the subtrees rooted at the children of the root of the solution, and then construct each such subtree recursively. On a high level, this is similar to the BUILD algorithm of Aho *et al.* [1] which also uses top-down recursion, but **DenseBuild** has to do the leaf partitioning in a different way to take the fan triplets into account. Also, **DenseBuild** needs to distinguish between when the root has degree 2 and degree strictly larger than 2 (cf., Lemma 7).

As a preprocessing step, **DenseBuild** constructs the fan graph \mathcal{G}_L and assigns a weight $w(x, y)$ to each edge $\{x, y\}$ in \mathcal{G}_L equal to $|\{x|y|z \in F^+ : z \in L\}|$. In the preprocessing step, the algorithm also computes and stores the value $f(a, b)$ for every $a, b \in L$. The next lemma shows that when the algorithm calls itself recursively, it does not have to recompute any $f(a, b)$ -values. For any $L' \subseteq L$ and $a, b \in L'$, define $f_{L'}(a, b) = |\{z : a|b|z \in F^+|L'\}|$.

Lemma 11. *Suppose that T is a tree with $A(T) = L$ and that T is consistent with the input. Let $L' \subseteq L$ be the set of leaves in a subtree rooted at any child of the root of T . Then $f_{L'}(a, b) = f_L(a, b) = f(a, b)$ for every $a, b \in L'$.*

Proof. Fix $a, b \in L'$. For any fan triplet of the form $a|b|z \in F^+$, z also has to belong to L' , and therefore $a|b|z \in F^+|L'$. Conversely, $a|b|z \in F^+|L'$ implies $a|b|z \in F^+$ by definition. Hence, $\{z : a|b|z \in F^+\} = \{z : a|b|z \in F^+|L'\}$. \square

After the preprocessing step is complete, **DenseBuild** proceeds as follows. It computes the connected components C_1, C_2, \dots, C_m of \mathcal{G}_L in step 1. According to Lemma 7, there are two main cases: if $m \geq 2$ then the root of any tree consistent with the input must have degree two, but if $m = 1$ then the root must have degree at least three.

In the former case (steps 2.1–2.3), the algorithm recursively constructs a tree T_i for the leaves in C_i for each $i \in \{1, 2, \dots, m\}$, thus handling the input rooted triplets over leaves within each connected component. To handle the rest, i.e., those whose leaves belong to more than one connected component in \mathcal{G}_L , the algorithm constructs an instance of non-dense \mathcal{R}^+ CONSISTENCY whose leaf label set represents the set of connected components in \mathcal{G}_L and whose set of

Algorithm DenseBuild

Input: Three sets R^+ , F^+ , F^- of rooted triplets over a leaf label set L forming a dense instance of $\mathcal{R}^+\mathcal{F}^{+-}$ CONSISTENCY.

The algorithm assumes the following preprocessing: \mathcal{G}_L has been constructed and edge-weighted, and $f(a, b)$ for all $a, b \in L$ have been pre-computed.

When making recursive calls, the algorithm passes $L' \subseteq L$ and $\mathcal{G}_{L'}$ as parameters.

Output: A tree T with $\Lambda(T) = L$ such that $R^+ \cup F^+ \subseteq t(T)$ and $F^- \cap t(T) = \emptyset$, if such a tree exists; otherwise, *null*.

```

1  Let  $C_1, C_2, \dots, C_m$  be the connected components of  $\mathcal{G}_L$ ;
2  if  $(m > 1)$  then
2.1 For  $i \in \{1, 2, \dots, m\}$ , extract  $\mathcal{G}_{L_i}$  from  $\mathcal{G}_L$  and compute  $T_i =$ 
    DenseBuild( $L_i, \mathcal{G}_{L_i}$ ), where  $L_i$  is the set of leaf labels in  $C_i$ ;
2.2 Let  $R' = \{C_i C_j | C_k : \exists xy|z \in R^+ \text{ with } x \in C_i, y \in C_j, z \in C_k\}$  and let  $T'$ 
    be the output of the BUILD algorithm on input  $R'$ ;
2.3 if  $T' = null$  or  $T_i = null$  for any  $i \in \{1, 2, \dots, m\}$  then return null;
    else let  $T$  be the tree obtained by arbitrarily refining  $T'$  to a binary tree
    and replacing each leaf  $C_i$  in  $T'$  by the tree  $T_i$ , and return T;
    else
3    /*  $(m = 1)$  */
3.1 Find  $a, b \in L$  that maximize  $f(a, b)$ ;
3.2 Let  $L' = \{a, b\} \cup \{x : a|b|x \notin F^+\}$ ,  $z \notin L'$ ,  $L_1 = \{a\} \cup \{x \in L' : a|x|z \notin$ 
     $F^+\}$ , and  $L_2 = L' \setminus L_1$ ;
3.3 Build the clique graph  $\mathcal{Q}_L$  and let  $L_3, \dots, L_p$  be the leaf labels in the dif-
    ferent connected components in  $\mathcal{Q}_L$ ;
3.4 if  $(\{x|y|z : x \in L_i, y \in L_j, z \in L_k, \text{ where } i, j, k \text{ are different}\} \subseteq F^+)$  and
     $\{xy|z \in R^+ : x \in L_i, y \in L_j, z \in L_k, \text{ where } i, j, k \text{ are different}\} = \emptyset$  then
3.4.1 Decrement  $w(x, y)$ ,  $w(x, z)$ , and  $w(y, z)$  by one for every  $x|y|z \in F^+$  such
    that  $x \in L_i, y \in L_j, z \in L_k$ , and  $i, j, k$  are different;
3.4.2 For  $i \in \{1, 2, \dots, p\}$ , extract  $\mathcal{G}_{L_i}$  from  $\mathcal{G}_L$  and compute  $T_i =$ 
    DenseBuild( $L_i, \mathcal{G}_{L_i}$ );
3.4.3 if  $T_i = null$  for any  $i \in \{1, 2, \dots, p\}$  then return null;
    else create a tree  $T$  by attaching the root of  $T_i$  for every  $i \in \{1, 2, \dots, p\}$ 
    to a common root node and return T;
    else
3.4.4 return null;
    endif
    endif
End DenseBuild

```

Fig. 2. Algorithm DenseBuild.

resolved triplets is $\{C_i C_j | C_k : \exists xy|z \in R^+ \text{ with } x \in C_i, y \in C_j, z \in C_k\}$. It then applies the BUILD algorithm from [1] to obtain a tree T' (if one exists) consistent with all resolved triplets in R^+ involving leaves from more than one connected component. (If no such T' exists or if some T_i -tree is *null*, DenseBuild will return *null* and give up.) Then, DenseBuild arbitrarily refines T' into a binary tree as in the proof of Lemma 2 above. Finally, the output tree T is obtained by replacing each C_i -leaf in T' by the corresponding T_i -tree. By Lemma 7.1, T is consistent with all fan triplets in F^+ and no fan triplets in F^- .

In the latter case (steps 3.1–3.4.4), Lemma 7.2 ensures that the partition of L into leaf label sets of the subtrees rooted at the children of the root is uniquely defined. This partition is recovered in steps 3.1–3.3 in accordance with Lemmas 8–10. Next, step 3.4 verifies that the resulting partition L_1, L_2, \dots, L_p is valid by checking if $x|y|z \in F^+$ and $xy|z \notin R^+$ hold for every $x \in L_i, y \in L_j, z \in L_k$ where i, j, k are different. If the partition is valid then, for each $i \in \{1, 2, \dots, p\}$, the algorithm first constructs \mathcal{G}_{L_i} (to avoid building \mathcal{G}_{L_i} from scratch, the weight $w(x, y)$ of each edge $\{x, y\}$ in \mathcal{G}_{L_i} is updated by subtracting 1 for every fan triplet $x|y|z \in F^+$ that contributed to $w(x, y)$ in \mathcal{G}_L but no longer exists on subsequent recursion levels; any edge whose weight reaches 0 is removed). Then, it recursively builds a tree T_i with $A(T_i) = L_i$. The output tree T is formed by attaching the roots of all the T_i -trees to a common root node.

Theorem 4. *Algorithm DenseBuild solves the dense version of the $\mathcal{R}^+ \mathcal{F}^{+-}$ CONSISTENCY problem in $O(n^3)$ time.*

Proof. The preprocessing step constructs \mathcal{G}_L , assigns weights to the edges in \mathcal{G}_L , and computes all values of $f(a, b)$ where $a, b \in L$, which takes $T_A(n) = O(n^3)$ time in total. We now bound the time needed to execute DenseBuild(L, \mathcal{G}_L) assuming that the preprocessing has been taken care of.

Let $T_B(n)$ be the total time used by the calls to BUILD in step 2.2 on all recursion levels, and let $T_C(n)$ be the total time for all other computations. To analyze $T_B(n)$, let n_1, n_2, \dots, n_k be the cardinalities of the leaf label sets of the constructed sets R' of resolved triplets in the successive calls to BUILD in step 2.2. By applying Henzinger *et al.* fast implementation of BUILD (Algorithm B' in [14]), we get $T_B(n) = \sum_{i=1}^k O(n_i^3 + n_i^2 \log n_i) = O(\sum_{i=1}^k n_i^3)$. Also, $n_1 + n_2 + \dots + n_k = O(n)$ because every leaf in each such constructed instance of \mathcal{R}^+ CONSISTENCY corresponds to either an internal node or a leaf in the tree output by DenseBuild, which has $O(n)$ nodes. Thus, $T_B(n) = O(n^3)$. Next, we derive an upper bound on $T_C(n)$. For any partition of L into L_1, L_2, \dots, L_m , let $c(L_1, L_2, \dots, L_m)$ denote the number of possible fan triplets of the form $x|y|z$ such that $x \in L_i, y \in L_j, z \in L_k$ and i, j, k are different. Observe that $c(L_1, L_2, \dots, L_m) = O(\binom{|L|}{3} - \sum_{i=1}^m \binom{|L_i|}{3})$. Then $T_C(n)$ consists of the time needed to find the m connected components in \mathcal{G}_L , which is $O(|L|^2)$, plus the time to:

- If $m \geq 2$:
 - (a) build \mathcal{G}_{L_i} for all $i \in \{1, 2, \dots, m\}$ ($O(c(L_1, L_2, \dots, L_m))$ time);
 - (b) construct R' (also $O(c(L_1, L_2, \dots, L_m))$ time); and
 - (c) handle the recursive calls ($\sum_{i=1}^m T_C(|L_i|)$ time).
- If $m = 1$:
 - (a) find the partition of L into L_1, L_2, \dots, L_p in steps 3.1–3.3 ($O(|L|^2)$ time);
 - (b) verify that the partition is valid in step 3.4 ($O(c(L_1, L_2, \dots, L_p))$ time); and
 - (c) handle the recursive calls ($\sum_{i=1}^p T_C(|L_i|)$ time).

Define $q = \max\{m, p\}$. In total, $T_C(n) = O(|L|^2) + O(c(L_1, L_2, \dots, L_q)) + \sum_{i=1}^q T_C(|L_i|)$, which gives $T_C(n) = O(n^3)$ by induction.

Finally, $T_A(n) + T_B(n) + T_C(n) = O(n^3)$. □

6 Concluding Remarks

The newly derived results (see Tables 1, 2, 3 for a summary) highlight the following open problems:

- What is the computational complexity of the D -bounded degree \mathcal{F}^+ CONSISTENCY problem when $D = 3$? I.e., is the following problem solvable in polynomial time: Given a set F^+ of fan triplets, does there exist a degree-3 tree consistent with all of F^+ ?
- For the special case of $D = 3$, do the following problems have the same computational complexity or not: D -bounded degree \mathcal{F}^+ CONSISTENCY, D -bounded degree \mathcal{F}^{+-} CONSISTENCY, D -bounded degree $\mathcal{R}^+\mathcal{F}^+$ CONSISTENCY, and D -bounded degree $\mathcal{R}^+\mathcal{F}^{+-}$ CONSISTENCY?
- How does the complexity of $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY and its problem variants change when other parameters such as the *height* of the output tree are restricted or if one requires the output tree to be *ordered* in such a way that its left-to-right sequence of leaves must equal a prespecified sequence? Note that the analogue of \mathcal{R}^+ CONSISTENCY in the *unrooted* setting where the input is a set of “quartets” (unrooted, distinctly leaf-labeled trees with four leaves where every internal node has three neighbors) is already NP-hard [22].
- Can fixed-parameter tractable algorithms be developed for any of the NP-hard variants of $\mathcal{R}^{+-}\mathcal{F}^{+-}$ CONSISTENCY?

One may also consider a minimization version of the D -bounded degree \mathcal{F}^+ CONSISTENCY problem, in which the input is a set F^+ of fan triplets and the objective is to construct a tree with as small degree as possible that is consistent with all fan triplets in F^+ . However, this is a difficult problem since Lemma 5 and the polynomial-time inapproximability result for the minimization version of K -COLORING by Zuckerman [25, Theorem 1.2] imply that the problem cannot be approximated within a ratio of $n^{1-\epsilon}$ for any constant $\epsilon > 0$ in polynomial time, unless $P = NP$.

Acknowledgments. The authors would like to thank Sylvain Guillemot and Avraham Melkman for some discussions related to the topic of this paper. J.J. was partially funded by The Hakubi Project at Kyoto University and KAKENHI grant number 26330014.

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**(3), 405–421 (1981)
2. Bininda-Emonds, O.R.P.: The evolution of supertrees. *TRENDS Ecol. Evol.* **19**(6), 315–322 (2004)
3. Bininda-Emonds, O.R.P., Cardillo, M., Jones, K.E., MacPhee, R.D.E., Beck, R.M.D., Grenyer, R., Price, S.A., Vos, R.A., Gittleman, J.L., Purvis, A.: The delayed rise of present-day mammals. *Nature* **446**(7135), 507–512 (2007)
4. Bryant, D.: Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis. Ph.D. thesis, University of Canterbury, Christchurch, New Zealand (1997)
5. Byrka, J., Gawrychowski, P., Huber, K.T., Kelk, S.: Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks. *J. Discrete Algorithms* **8**(1), 65–75 (2010)
6. Byrka, J., Guillemot, S., Jansson, J.: New results on optimizing rooted triplets consistency. *Discrete Appl. Math.* **158**(11), 1136–1147 (2010)
7. Chor, B., Hendy, M., Penny, D.: Analytic solutions for three taxon ML trees with variable rates across sites. *Discrete Appl. Math.* **155**(6–7), 750–758 (2007)
8. Constantinescu, M., Sankoff, D.: An efficient algorithm for supertrees. *J. Classif.* **12**(1), 101–112 (1995)
9. Jansson, J., Lingas, A., Lundell, E.-M.: The approximability of maximum rooted triplets consistency with fan triplets and forbidden triplets. In: Cicalese, F., Porat, E., Vaccaro, U. (eds.) *CPM 2015. LNCS*, vol. 9133, pp. 272–283. Springer, Cham (2015). doi:[10.1007/978-3-319-19929-0_23](https://doi.org/10.1007/978-3-319-19929-0_23)
10. Felsenstein, J.: *Inferring Phylogenies*. Sinauer Associates, Inc., Sunderland (2004)
11. Garey, M., Johnson, D.: *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York (1979)
12. Gąsieniec, L., Jansson, J., Lingas, A., Östlin, A.: On the complexity of constructing evolutionary trees. *J. Comb. Optim.* **3**(2–3), 183–197 (1999)
13. He, Y.J., Huynh, T.N.D., Jansson, J., Sung, W.-K.: Inferring phylogenetic relationships avoiding forbidden rooted triplets. *J. Bioinform. Comput. Biol.* **4**(1), 59–74 (2006)
14. Henzinger, M.R., King, V., Warnow, T.: Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. *Algorithmica* **24**(1), 1–13 (1999)
15. Holm, J., de Lichtenberg, K., Thorup, M.: Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM* **48**(4), 723–760 (2001)
16. Jansson, J., Lemence, R.S., Lingas, A.: The complexity of inferring a minimally resolved phylogenetic supertree. *SIAM J. Comput.* **41**(1), 272–291 (2012)
17. Jansson, J., Ng, J.H.-K., Sadakane, K., Sung, W.-K.: Rooted maximum agreement supertrees. *Algorithmica* **43**(4), 293–307 (2005)

18. Ng, M.P., Wormald, N.C.: Reconstruction of rooted trees from subtrees. *Discrete Appl. Math.* **69**(1–2), 19–31 (1996)
19. Semple, C.: Reconstructing minimal rooted trees. *Discrete Appl. Math.* **127**(3), 489–503 (2003)
20. Semple, C., Daniel, P., Hordijk, W., Page, R.D.M., Steel, M.: Supertree algorithms for ancestral divergence dates and nested taxa. *Bioinformatics* **20**(15), 2355–2360 (2004)
21. Snir, S., Rao, S.: Using max cut to enhance rooted trees consistency. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **3**(4), 323–333 (2006)
22. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classif.* **9**(1), 91–116 (1992)
23. Sung, W.: *Algorithms in Bioinformatics: A Practical Introduction*. Chapman & Hall/CRC, Boca Raton (2010)
24. Willson, S.J.: Constructing rooted supertrees using distances. *Bull. Math. Biol.* **66**(6), 1755–1783 (2004)
25. Zuckerman, D.: Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. *Theory Comput.* **3**(1), 103–128 (2007)

Progressive Calibration and Averaging for Tandem Mass Spectrometry Statistical Confidence Estimation: Why Settle for a Single Decoy?

Uri Keich¹(✉) and William Stafford Noble²

¹ School of Mathematics and Statistics F07, University of Sydney,
Sydney, Australia

`uri@maths.usyd.edu.au`

² Department of Genome Sciences, Department of Computer Science and
Engineering, University of Washington, Seattle, USA

`william-noble@uw.edu`

Abstract. Estimating the false discovery rate (FDR) among a list of tandem mass spectrum identifications is mostly done through target-decoy competition (TDC). Here we offer two new methods that can use an arbitrarily small number of additional randomly drawn decoy databases to improve TDC. Specifically, “Partial Calibration” utilizes a new meta-scoring scheme that allows us to gradually benefit from the increase in the number of identifications calibration yields and “Averaged TDC” (a-TDC) reduces the liberal bias of TDC for small FDR values and its variability throughout. Combining a-TDC with “Progressive Calibration” (PC), which attempts to find the “right” number of decoys required for calibration we see substantial impact in real datasets: when analyzing the *Plasmodium falciparum* data it typically yields almost the entire 17% increase in discoveries that “full calibration” yields (at FDR level 0.05) using 60 times fewer decoys. Our methods are further validated using a novel realistic simulation scheme and importantly, they apply more generally to the problem of controlling the FDR among discoveries from searching an incomplete database.

Keywords: Tandem mass spectrometry · Spectrum identification · False discovery rate · Calibration

1 Introduction

In tandem mass spectrometry analysis, the problem of inferring which peptide was responsible for generating an observed fragmentation spectrum is crucial to

Electronic supplementary material The online version of this chapter (doi:[10.1007/978-3-319-56970-3_7](https://doi.org/10.1007/978-3-319-56970-3_7)) contains supplementary material, which is available to authorized users.

any subsequent analysis about the presence or quantity of peptides and proteins in the complex mixture being analyzed. Unfortunately, this spectrum identification problem is difficult to solve because, for any given spectrum, many expected fragment ions will not be observed, and the spectrum is also likely to contain a variety of additional, unexplained peaks.

The most common approach to the spectrum identification problem is peptide database search. Pioneered by SEQUEST [7], the search engine extracts from the peptide database all “candidate peptides” defined by having their mass lie within a pre-specified tolerance of the measured mass of the intact peptide (the “precursor mass”). The quality of the match between each of these candidate peptides and the observed fragmentation spectrum is then evaluated using a score function. Finally, the best-scoring peptide-spectrum match (PSM) for the given spectrum is reported, along with its score.

Sometimes the reported PSM is correct—the peptide assigned to the spectrum was present in the mass spectrometer when the spectrum was generated—and sometimes the PSM is incorrect. Ideally, we would report only the correct PSMs, but obviously we are not privy to this information: all we have is the score of the PSM, indicating its quality. Therefore, we report a thresholded list of top-scoring PSMs, together with the critical estimate of the fraction of incorrect PSMs in our reported list. This work focuses on methods for carrying out this discovery and error estimation procedure.

The problem of controlling the proportion of false discoveries has been studied extensively in the context of multiple hypotheses testing (MHT), starting with the seminal work of Benjamini and Hochberg [3]. Specifically, they introduced a simple procedure that allows us to decide which null hypotheses we reject (thus declaring them as “discoveries”) so that the FDR, which they defined as the *expected value* of the proportion of false discoveries (FDP), is bounded by a pre-determined level α .

The mass spectrometry community, however, relies mostly on other methods to control the FDR. The main reason is that the MHT context is predicated on associating a p-value with each tested null hypothesis, indicating how unlikely that result is assuming the hypothesis is truly a null one. Until recently, no such p-values were computed in the PSM context. Moreover, while considerable effort has of late been invested in computing such p-values [1, 9, 12, 15–17], we recently showed that there are further subtle but fundamental differences between the MHT context and the PSM one, implying that we typically cannot use FDR controlling procedures that were designed for the MHT context [13].

Instead, the most widely used FDR controlling procedure in this context is a decoy-based protocol called target-decoy competition (TDC), proposed by Elias and Gygi [5]. The *target* in TDC refers to the real peptide database of interest, and the *decoy* is a database of randomly shuffled or reversed peptides. The method consists of searching a given set of spectra against the concatenated target-decoy database and retaining the single best-scoring PSM for each spectrum. As a result of this selection, any optimal target PSM that scores less than the corresponding optimal decoy PSM is eliminated from consideration.

Subsequently, at a given score threshold, the number of accepted decoy PSMs provides an estimate of the number of false discoveries, or accepted incorrect target PSMs [4, 6, 10].

The quality of an FDR controlling procedure (more precisely, of a discovery and FDR controlling procedure) can be evaluated along at least three orthogonal dimensions. First, we can gauge the procedure’s power: how many (correct) target discoveries, or spectra identifications, does it report at a given FDR threshold? Second, we can analyze the accuracy of the procedure: how close is the actual FDR to the estimated one? In determining the accuracy we ask whether the method is biased or not, where liberally biased methods (those that underestimate the FDR) are particularly undesirable because they instill in the user more confidence than is due. Third, in addition to controlling bias, we also prefer methods that exhibit less variability, since exceedingly high FDP could have substantial impact on any downstream analysis.

The primary contributions of this paper are two novel procedures for improving decoy-based FDR control procedures in the context of the mass spectrum identification problem. The first procedure—partial calibration—yields improved statistical power relative to TDC; the second procedure—averaged TDC (a-TDC)—yields reduced variance. Both procedures maintain the asymptotic unbiased control of the FDR of TDC, and a-TDC mitigates much of the liberal bias of TDC observed at small FDR values [11].

Partial Calibration. The partial calibration procedure is motivated by our recently described calibration method [12]. We showed that calibrating the scores (placing the scores of all PSMs on the same scale regardless of the spectrum involved) can substantially increase the power of TDC.¹ For example, we found that when calibrating the popular XCorr score [7], using the *same set of spectra* the number of discoveries at 1% FDR increased in the range of 12–31% [12]. However, since our calibration method relied on searching 10,000 randomly generated decoy databases (obtained by repeatedly shuffling each peptide of the target database), our procedure was computationally extremely demanding.

In this work, we show that the advantages calibration offers can be gradually realized starting with a relatively modest requirement of one additional decoy set and increasing according to the user’s computational resources. In a nutshell, partial calibration uses the calibrating decoys to convert the raw scores into empirical p-values. However, whereas our original approach employed the commonly used method of replacing the raw score with the empirical p-value, here we keep both and use a two tiered scoring scheme. The primary score is the empirical p-value, with ties resolved by the secondary score, which is the raw score. This new primary-secondary scheme allows us to use as few as a single calibrating decoy in a meaningful way. Previously, in such a case roughly half

¹ More rigorously we say that the scoring function of an optimal PSM is “calibrated” if the distribution of the score of an optimal PSM in a randomly drawn decoy database is invariant of the spectrum itself.

the (null) scores would have one p-value (0) and the other half would have a different p-value (1), so very little could have been done with this data.

By allowing us to benefit from any number of calibrating decoys, partial calibration raises the question of how many calibrating decoy sets are “enough.” One option is to let our computational resources determine the number of decoys we can afford to generate for the given data. However, the downside of this strategy is that for some datasets and FDR thresholds we would spend too much effort, whereas in other cases we would still achieve sub-par results. An ideal approach would allow us to intelligently trade off between statistical power and computational expense.

Here we propose an ad hoc method, called “progressive calibration” (PC), that employs a doubling strategy to dynamically determine the number of calibrating decoys our partial calibration procedure should use. The method works by factoring in the user’s computational limits, the particular dataset at hand, and the range of FDR values the user is interested in. The latter is a particularly important factor because, empirically, the law of diminishing returns, in terms of number of discoveries per number of calibrating decoys, kicks in much sooner for higher FDR levels.

Averaged TDC. The second primary procedure, a-TDC, is motivated by simulations that show that, for sets of 1000 spectra, the actual FDP among the PSMs selected by using TDC with an FDR threshold of 0.05 can readily be $\pm 50\%$ of that level, and this problem gets much worse for smaller-sized sets of spectra and tighter FDR levels [11]. Although calibration can somewhat reduce TDC’s variability [12], even if we achieve perfect calibration we still cannot get around the inherent decoy-dependent variability of TDC.

a-TDC gets around this problem by applying TDC to the target database paired with a small number (n_p) of randomly drawn “competing” decoy databases and “averaging” the results. Clearly, averaging will reduce the TDC variance, but the challenge is to make sense of this averaging, especially because the list of TDC discoveries varies with each competing decoy database.

One might be tempted to define this list of “average target discoveries” as all the target PSMs that outscore the majority of their decoy competitions. That is, a target PSM is an a-TDC discovery if it is a (TDC) discovery in more than $n_p/2$ of the n_p concatenated target-decoy searches². While intuitively appealing, when paired with the equally appealing averaging of the (TDC) estimated FDR, this approach quickly becomes too liberal: the FDR is underestimated.

We therefore devised a more nuanced approach which sequentially constructs its target discovery list starting from the highest target PSM score. Our method then goes through the decreasing target PSM scores, ensuring that the number of discoveries at the current score level does not deviate from the average number of (TDC) target discoveries, at the same score level, across our n_p independent TDC procedures. In order to meet this guarantee, a-TDC occasionally needs to filter out or reject a target PSM as it goes down the list. At that point, the PSM

² For reference, Supp. Table 1 provides a summary of our notations.

that is selected for rejection is the one with the smallest score among all hitherto selected target discoveries that lost the most decoy competitions (Sect. 2.4).

At this point there are several plausible ways to define the corresponding a-TDC estimate of the FDR among its list of discoveries. We settled on the ratio between the average number of decoy discoveries across the n_p independent TDC procedures, and the actual number of a-TDC discoveries. Importantly, this definition makes a-TDC with a single decoy ($n_p = 1$) identical to TDC.

Verification. We apply our novel procedures—partial calibration (and its adaptive variant, PC) and a-TDC—to real as well as simulated data. Most simulations of the spectrum identification problem, carried out by us as well as others, have used calibrated scores. However, because much of our work here is dedicated to the effects of partial calibration, it was crucial to develop a simulation procedure using uncalibrated scores.

The simulated data supports our claim that our procedures control the FDR on-par or better than TDC does, and both the real and simulated data show that a-TDC reduces the variability of TDC and that partial calibration can yield a sizable increase in the number of target discoveries. In addition, we observe that, for a typical FDR range of interest, PC allows us to enjoy most of the gains offered by our original, brute-force calibration procedure, while employing significantly fewer than 10,000 decoys.

2 Methods

2.1 TDC, FDR Estimation, and Target Discoveries

An FDR controlling procedure returns a list of discoveries together with an estimate of the FDR among the reported discoveries. In particular, TDC defines its list of $T(\rho)$ discoveries at score level ρ as all target PSMs with score $\geq \rho$ that outscore their corresponding decoy competition (i.e., they remain discoveries in the search of the concatenated database). Denoting by $D(\rho)$ the number of decoy discoveries at score level ρ in the concatenated search, TDC estimates the FDR in its target discovery list as $\widehat{\text{FDR}}(\rho) := D(\rho)/T(\rho)$.

Often, the user is more interested in specifying a desired FDR level τ . In this context the score threshold that corresponds to an (estimated) FDR level of τ is $\rho(\tau) := \min \left\{ \rho : \widehat{\text{FDR}}(\rho) \leq \tau \right\}$. We refer to $T(\tau)$, the number of target discoveries at (estimated) FDR level τ , as short for $T(\rho(\tau))$, the number of discoveries at score level $\rho(\tau)$, and similarly for the list of actual target discoveries at FDR level τ . For computational efficiency we limited our attention to a predetermined set of FDR values (Supp. Sect. 1.1). Note that the latter relation between τ and $\rho(\tau)$ is defined for *any* FDR estimation method and is not specific to TDC.

2.2 Calibrating and Competing Decoys

Let Σ denote the set of spectra generated in the experiment. We associate with each spectrum $\sigma \in \Sigma$ its optimal matching peptide in the target database, which

we loosely refer to as the “target PSM,” or just the “target score,” $w(\sigma)$, when we refer to the score of that PSM.

Similarly, we assume that each spectrum σ is searched against two sets of randomly drawn decoy databases that, in practice, are generated by independently shuffling each peptide in the target database. The sets are *statistically identical* but we refer to one, $\{\mathcal{D}_i^b\}_{i=1}^{n_b}$, as the “calibrating” set of decoy databases and the other, $\{\mathcal{D}_j^p\}_{j=1}^{n_p}$, as the “competing” set of decoys. The distinction between the two sets of decoys is based on the different roles they play. The calibrating decoys are used to calibrate the scores whereas the competing decoys are used for estimating the FDR using target decoy competition.

The score of the optimal match to σ in \mathcal{D}_i^b is denoted by $z_i^b(\sigma)$, and similarly $z_j^p(\sigma)$ is the score of the optimal match to σ in \mathcal{D}_j^p . For each fixed σ the distribution of $z_i^b(\sigma)$ (with respect to a randomly drawn decoy) is identical to that of $z_j^p(\sigma)$. Importantly, since we do not assume that the score is necessarily calibrated, the said distribution can vary with the spectrum σ .

2.3 Partial Calibration

By “partial calibration” we refer to a procedure that allows us to convert a raw score into a new score that is “more calibrated.” We prefer to be somewhat vague on what exactly the latter means but, intuitively, it means that the distribution of the decoy scores $z(\sigma)$ is “less varied” with respect to the spectrum σ .

Our specific procedure here first uses the calibrating scores associated with the spectrum σ , $\{z_i^b(\sigma)\}_{i=1}^{n_b}$, to assign to each observed competing decoy score $s = z_j^p(\sigma)$ or target score $s = w(\sigma)$ a new, primary score, $q_\sigma(s)$. This primary score is equivalent to the p-value of s with respect to the empirical cumulative distribution function (ECDF) constructed from the calibrating scores:

$$q_\sigma(s) = q\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}\right) := |\{i : z_i^b(\sigma) < s\}| + \frac{1}{2} |\{i : z_i^b(\sigma) = s\}|.$$

The secondary score assigned to s is the score s itself.

Using our primary-secondary score we define a new linear order, \succ , on the set of all observed target and competing decoy scores as follows. Let s_i be a score of an optimal PSM involving the spectrum σ_i . Instead of using the raw scores s_1 and s_2 to determine the order, we now say $s_1 \succ s_2$ if $q_{\sigma_1}(s_1) > q_{\sigma_2}(s_2)$, or if $q_{\sigma_1}(s_1) = q_{\sigma_2}(s_2)$ and $s_1 \geq s_2$.

Technically, we implement the new order, \succ , by first converting all observed target scores, $\mathcal{W} = \{w(\sigma) : \sigma \in \Sigma\}$, and *competing* decoy scores, $\mathcal{Z}^p = \{z_j^p(\sigma) : \sigma \in \Sigma, j = 1, \dots, n_p\}$, into ranks, where the rank of 1 corresponds to the smallest observed score and the rank of $|\Sigma|(n_p + 1)$ corresponds to the largest observed score. We then map each observed raw score s associated with the spectrum σ to the partially calibrated score

$$\psi_\sigma(s) = \psi\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}, \mathcal{W} \cup \mathcal{Z}^p\right) := q_\sigma\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}\right) + \frac{1}{1+2|\Sigma|(n_p+1)} r(s; \mathcal{W} \cup \mathcal{Z}^p),$$

where $r(s; \mathcal{W} \cup \mathcal{Z}^p)$, is the rank of the raw score s in the list of $|\Sigma|(n_p + 1)$ observed target and competing decoys scores. It is easy to see that, given the set $\mathcal{W} \cup \mathcal{Z}^p$, for any observed pair of scores s_1 and s_2 from that set, $s_1 \succcurlyeq s_2$ if and only if $\psi_\sigma(s_1) \geq \psi_\sigma(s_2)$.

If the size of the calibrating set n_b is very large then, assuming $s_1 \neq s_2$, it is very unlikely that $q_{\sigma_1}(s_1) = q_{\sigma_2}(s_2)$ and the new ordering will coincide with the one determined by the spectrum specific ECDFs that was used in our previously described calibration procedure [12]. At the other extreme end, when there are no calibrating decoys we revert to the ordering determined by the raw score. All other cases in some sense interpolate between these two extremes, with more weight placed on the ECDF the more refined it is.

Two points are worth noting. First, if the raw score is already calibrated then our partial calibration procedure will leave it calibrated. More precisely, recall that we defined the optimal PSM score function as calibrated if the distribution of $z(\sigma)$, the score of the optimal match to σ in a randomly drawn database, \mathcal{D} , is invariant of the spectrum σ . Assuming that the calibrating decoys are drawn at the same time as \mathcal{D} , our new score ψ will also be calibrated. Second, the definition of $\psi_\sigma(s)$ allows us to efficiently utilize an increasing number of calibrating decoys – a fact that we will return to when discussing progressive calibration.

2.4 Averaged TDC

The a-TDC procedure begins with repeatedly applying TDC to the target database \mathcal{D}^t paired with each of the n_p independently drawn (competing) decoy databases \mathcal{D}_i^p , for $i = 1, \dots, n_p$. Let $T_i(\rho)$ and $D_i(\rho)$ denote the number of target, respectively, decoy discoveries at level ρ , that are reported by TDC in the i th application. Recall that $D_i(\rho)$ is used to estimate $F_i(\rho)$, the corresponding number of *false* target discoveries, and note that the reported list of target discoveries typically changes with each decoy database.

Let ρ_i denote the decreasing target PSM scores, and let $\overline{T(\rho_i)} := \sum_{j=1}^{n_p} T_j(\rho_i) / n_p$ and $\overline{D(\rho_i)} := \sum_{j=1}^{n_p} D_j(\rho_i) / n_p$ be the average of numbers of target and decoy discoveries, respectively, at level ρ_i across our n_p TDC procedures. Our a-TDC procedure sequentially constructs its discovery list (and simultaneously its filtered target PSMs list), ensuring that its number of discoveries at level ρ_i , $T(\rho_i)$, does not deviate from $\overline{T(\rho_i)}$ (apart from the inevitable difference due to rounding).

When a-TDC determines that it needs to filter out a target PSM, it rejects the PSM with the lowest partially calibrated score $\psi\left(s; \{z_j^p(\sigma)\}_{j=1}^{n_p}, \mathcal{W}\right)$ among all hitherto selected target PSMs with raw score $s \geq \rho_i$. Note that the latter partially calibrated score is with respect to the *competing* decoys rather than the usual calibrating decoys, and that the rank component of the score is taken only with respect to the target scores. In other words, the rejected PSM is the one with the smallest raw score among all remaining target discoveries scoring $\geq \rho_i$ that lost the most decoy competitions. Finally, a-TDC estimates the FDR in its level ρ_i discovery list as $\widehat{\text{FDR}}(\rho_i) := \overline{D(\rho_i)} / T(\rho_i)$ (Supp. Algorithm 1).

2.5 Progressive Calibration with Mean Cutoff Criterion

Progressive calibration (PC) starts with zero calibrating decoys (raw scores) and goes through several cycles of essentially doubling the number of calibrating decoys. At the i th cycle we randomly draw 2^{i-1} additional calibrating decoy databases and search each of our spectra against these databases. Thus, after the i th doubling, for each spectrum we have a set of $2^i - 1$ calibrating decoy scores, which contains the corresponding calibrating decoy set of the previous doubling cycle. The process terminates if the cutoff criterion below was engaged, or the maximal number of calibrating decoys was reached (2047 for our simulations and 10,000 for the real data).

Adjusting the partially calibrated score to take into account the newly drawn set of decoys PSM scores, in each of PC's doubling cycles, is straightforward due to the identity

$$q\left(s; \{z_i^b(\sigma)\}_{i=1}^m\right) + q\left(s; \{z_i^b(\sigma)\}_{i=m+1}^{n_b}\right) = q\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}\right), \quad (1)$$

which in turn implies

$$\psi\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}, \mathcal{W} \cup \mathcal{Z}^p\right) = q\left(s; \{z_i^b(\sigma)\}_{i=1}^m\right) + q\left(s; \{z_i^b(\sigma)\}_{i=m+1}^{n_b}\right) + \frac{1}{1+2^i \Sigma(n_p+1)} r\left(s; \mathcal{W} \cup \mathcal{Z}^p\right).$$

Hence, we only need to compute the ranks, $r\left(s; \mathcal{W} \cup \mathcal{Z}^p\right)$ for all s observed target scores, \mathcal{W} , and *competing* decoy scores, \mathcal{Z}^p , once and then update $q\left(s; \{z_i^b(\sigma)\}_{i=1}^{n_b}\right)$ using (1).

We will see below (Fig. 1) that for some combinations of data and FDR levels, partial calibration can achieve near optimal results with very few calibrating decoys. In other cases, and particularly for very small FDR levels, achieving near optimal results requires many more calibrating decoys. Taking this into account, PC's stopping criterion focuses only on the mean increase in the number of discoveries for FDR levels in a range that is specified by the user (≥ 0.05 in our experiments). The exact details are in Supp. Sect. 1.2.

Note that the above cutoff criterion applies regardless of whether the FDR estimation is done using TDC or a-TDC. In addition, we only engage the cutoff criterion from the third doubling cycle onward (so we use at least seven calibrating decoys).

2.6 Simulations Using Uncalibrated Scores

One can readily simulate raw decoy scores by searching real spectra against randomly shuffled versions of real peptide databases, but it is less clear how to simulate target PSM scores while still knowing which ones are "correct" and which are "false." Here we accomplish this by first sampling the optimal PSM scores using a variant of our previously described calibrated sampling scheme [14], where false PSMs are drawn from a null distribution and correct PSM are drawn from an alternative, beta distribution, and each spectrum has a fixed number of candidate peptides it can match. We then convert, these calibrated

PSM scores to raw scores using a spectrum specific transformation modeled after a real data set, as explained next.

We begin with associating with each spectrum from the real set (here we used the yeast dataset, Supp. Sect. 1.4) an ECDF constructed from a sample of 10K optimal decoy PSM scores, which are obtained by searching the spectrum against 10K randomly shuffled versions of the target database. While we could have converted the calibrated scores to raw scores using the quantiles of this spectrum-specific ECDF, this would have limited the granularity of our score when analyzing the often encountered high scoring correct PSMs.

Therefore, to preserve the necessary granularity in our scoring function we instead relied on the observation that the distribution of the null optimal PSM scores of a specific spectrum can often be well approximated by a Gumbel EVD [17]. Specifically, we fitted a location-shifted and scaled Gumbel distribution to each of our spectrum-specific ECDFs generated by the yeast real data. We then randomly associated the fitted Gumbel distributions to our simulated spectra and used the quantiles of those fitted distributions to convert our initially sampled calibrated scores (Supp. Sect. 1.3). Using this approach our simulated raw scores inherit the uncalibrated nature of the real yeast data.

2.7 Real Data Analysis

We analyze three real data sets, derived from yeast, *C. elegans* (worm), and *Plasmodium falciparum* (“malaria”). For each of these three sets, we conducted 2000 independent experiments, each of which consisted of randomly drawing 10 distinct competing decoys from a pool of 1K such decoys. We then applied partial calibration to both the target and the competing decoy PSM scores, using an increasing number of calibrating decoys, which were randomly drawn from a pool of 10K such decoys. We next applied a-TDC (using the 10 drawn competing decoys) and TDC (using just the first of those competing decoys) to the increasingly calibrated data, noting the number of discoveries and virtually applying PC to the data. Further details are provided in Supp. Sect. 1.4.

3 Results

3.1 Partial Calibration Yields More Statistical Power

The effectiveness of our partial calibration scheme is demonstrated by our new raw score simulation method. For example, looking at the mean number of TDC target discoveries across 10K runs, each of which simulated a set of spectra of size 10K with 50% native spectra (these are spectra for which the correct peptide is in the target database), we find that this mean consistently increases with the number of calibrating decoys (Fig. 1B). Moreover, for FDR levels which are not very small, much of that increase can be realized with a relatively small number of calibrating decoys, e.g., at an estimated FDR level of 0.05 using no calibrating decoys (raw score) the mean number of TDC discoveries is 3317, but using 31

calibrating decoys it rises to 3726 (12% increase) and with 63 decoys it is at 3942, which is 98% of the maximal 4038 average discoveries (22% increase over the raw score) obtained when using 2047 calibrating decoys (Fig. 1B). Of course, the increase in the mean number of discoveries varies with the parameters of the problem, but our simulations show a consistent increase with the number of calibrating decoys (Supp. Fig. 1).

The accuracy of TDC, in terms of the actual FDR (as estimated by the empirical mean of the FDP across 10K samples) over the nominal threshold, seems largely unchanged by the increase in the number of calibrating decoys (Fig. 1A and Supp. Fig. 2, middle curves). At the same time, as expected, the variability of the estimate decreases slightly with the increased calibration (same figures, upper and lower set of curves).

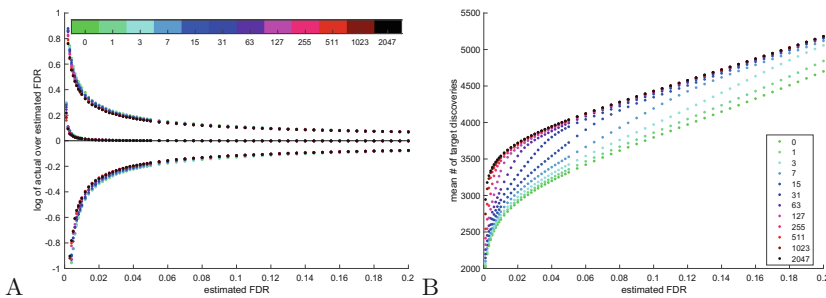


Fig. 1. Partial calibration (TDC). **A** The set of middle curves, which correspond to the log of the ratio between the empirical FDR and the nominal FDR level, essentially coincide for all considered numbers of calibrating decoys (the modest liberal bias of TDC for low FDR values is discussed in Sect. 3.2). The set of lower and upper curves correspond to the log of the ratio of the 0.05 and 0.95 quantiles of the FDP to the nominal FDR level. **B** The mean number of (TDC) target discoveries consistently increases with the number of calibrating decoys, although the law of diminishing returns is quite evident. **A–B** All means and quantiles are taken with respect to 10K simulation runs using our raw score, each with 10K spectra, 50% native spectra. The number of calibrating decoys was varied from 0 to 2047 (see Methods for details).

3.2 Averaged TDC

With Calibrated Scores. Figure 2A demonstrates that a-TDC reduces the variability in FDR estimation, even when the score is perfectly calibrated. This reduction is more pronounced with smaller sets of spectra and smaller FDR levels (Supp. Fig. 3). As a consequence, the variabilities in the reported number of discoveries as well as of *false* discoveries are also reduced (Fig. 2B, and Supp. Figs. 5 and 7). These variance reductions imply that the actual list of target discoveries should also exhibit reduced variability compared with single-decoy TDC, although we did not try to quantify this effect here.

Interestingly, a-TDC also typically mitigates much of the previously noted liberal bias of TDC [11], as can be seen in the set of middle curves of Fig. 2A and Supp. Fig. 3, which compare the empirical FDR (average of the FDP with respect to 10K independently drawn sets) with the selected FDR threshold (nominal level) using TDC as well as a-TDC with 3, 10 and 100 decoys.

With Raw Scores. As expected, when using a raw, uncalibrated score, a-TDC reduces TDC’s variability even slightly more effectively (Fig. 2C, and Supp. Figs. 4, 6, and 8). Unexpectedly however, a-TDC also becomes slightly conservative as the number of competing decoys increases (Fig. 2C, and Supp. Fig. 4). In spite of this trend, with the exception of very small estimated FDR levels, where TDC is clearly liberally biased, a-TDC is typically making at least as many *true* discoveries as does TDC. Moreover, there are cases in which the number of true discoveries increases with the number of competing decoys that a-TDC utilizes, and in particular, in those cases it is typically making more true discoveries than TDC does (Fig. 2D and Supp. Fig. 9).

The a-TDC procedure yields more true discoveries than TDC when using an uncalibrated score because a-TDC benefits from the same effect that partial calibration does: by having a better way to order the PSMs. While, strictly speaking, a-TDC is not reordering the PSMs as partial calibration does, a-TDC selects the target PSMs for filtering based on the partially calibrated score with respect to the competing decoys; hence, a-TDC engages in implicit calibration.

a-TDC Benefits from Partial Calibration. We next investigated the benefits of combining our two procedures, a-TDC and partial calibration. We find that, similar to TDC, a-TDC can gain a significant boost in statistical power, as can be seen by the increase in the number of target discoveries in Supp. Fig. 10. As expected from our analyses of a-TDC’s performance, a-TDC’s power advantage over TDC diminishes with the increase in the number of calibrating decoys (Fig. 2E, Supp. Fig. 11). Indeed, when the score is perfectly calibrated a-TDC should not have more power than TDC does; regardless, for all degrees of calibration, a-TDC does exhibit reduced variability (e.g., Supp. Fig. 12). At the same time, a-TDC become less conservative with the increase in the number of calibrating decoys (Fig. 2F, Supp. Fig. 13).

3.3 Progressive Calibration Dynamically Decides How Many Decoys Are Sufficient

Progressive calibration can be quite effective in significantly reducing the number of calibrating decoys that we use while achieving near-optimal power. In Fig. 3A–B) we repeatedly simulated identifying 10K spectra (50% native) and used PC coupled with TDC to control the FDR. The experiment was repeated 10K times, and the average number of calibrating decoys determined by our PC procedure was only 117. Still, in terms of power little was lost: comparing the ratio of the number of (TDC) target discoveries our PC procedure made to the number of

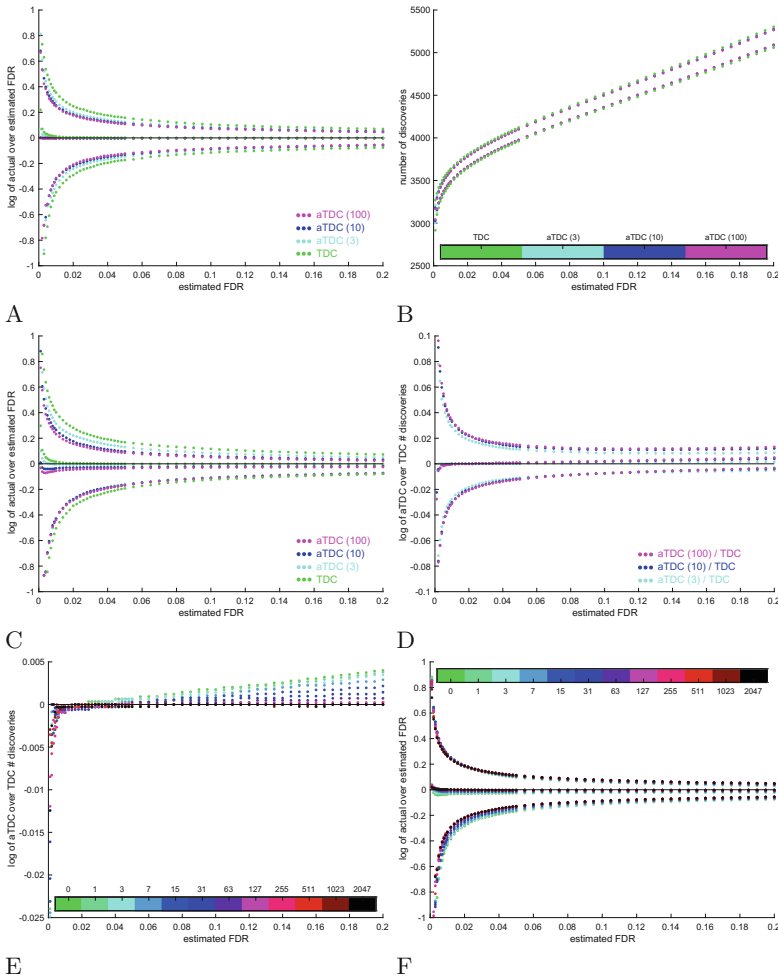


Fig. 2. Averaged TDC (a-TDC). **A–D** Comparing the FDR controlling procedure of a-TDC with 1 (TDC), 3, 10, and 100 competing decoys. **E–F** a-TDC with 10 competing decoys. **A** Plotted are the log of the ratios of the mean (empirical FDR, middle curves) as well as the 0.05 and 0.95 quantiles (upper and lower curves) of the FDP in the target discovery lists of each of the four procedures, to the nominal FDR level. Scores are calibrated. **B** The 0.05 and 0.95 quantiles of the number of target discoveries. Scores are calibrated. **C** Same as panel A, except the simulations were done using the raw (uncalibrated) score. **D** Shown are the logarithm of the median (middle curves), 0.05 and 0.95 quantiles of the number of *true* target discoveries reported by a-TDC (with 3, 10, and 100 decoys) over the corresponding number reported by TDC at the same FDR threshold. Scores are uncalibrated. **E** The log of the median of the ratio of the number of *true* a-TDC to TDC discoveries show that the power advantage of a-TDC over TDC diminishes with the increase in the number of calibrating decoys. **F** Coincidentally, a-TDC becomes less conservative: the middle set of curves show that the log of the empirical FDR (mean of FDP) over the nominal level increases toward 0 for small FDR levels. (The 0.05 and 0.95 quantiles are also provided.) **A–F** All quantiles are taken with respect to 10K simulations, each with 10K spectra, 50% native spectra.

(TDC) target discoveries attained by the maximally considered 2047 calibrating decoys we find a median of 99.3% and 0.95 quantile of 98% for all nominal FDR levels ≥ 0.05 . In other words, while using about 20 times fewer calibrating decoys, PC delivered 98% of the target discoveries at any estimated FDR level ≥ 0.05 in 95% of our 10K experiments. For results using additional spectrum set sizes and proportions of native spectra see Supp. Fig. 14.

Our cutoff criterion is not infallible. Indeed, our simulations show that when the set of spectra or the number of correct PSMs is small, then progressive calibration might fail to achieve the near-optimal results TDC can achieve with “full” calibration. For example, for $n = 500$ and a native spectrum proportion of 10% (Supp. Fig. 15), the median increase (across 10K experiments) in the number of TDC target discoveries made when using 2047 calibrating decoys compared with using PC at FDR level 0.05 is 14%. The corresponding 0.95 quantile is 60% additional discoveries; that is, in 5% of the experiments the increase in discoveries at FDR 0.05 is 60% or higher. One should, however, keep in mind that the mean number of additional discoveries the more intensive calibration effort yields here at FDR 0.05 is about 5. Similarly, with 500 spectra and 50% native spectra, we see in 5% of the experiments an increase higher than 16% when using TDC with 2047 decoys (the median increase is only 1.5%).

It is not surprising that combining a-TDC with PC still offers reduced variability in FDR estimation compared to combining TDC with PC (Supp. Fig. 18). However, with its reduced variability a-TDC can also help us here in better identifying those cases where increased calibration can yield a non-negligible number of additional discoveries. For example, in the same experiment described above with 500 spectra, of which 50% are native, we find that in 95% of the runs the increase in a-TDC discoveries using 2047 decoys over using PC at a nominal FDR level 0.05 is no more than 6.8% (Supp. Fig. 17), compared with 16% when using TDC. This experiment demonstrates that a-TDC is less likely to prematurely terminate the doubling cycle, and it translates here to observing in 5% of the experiments a 14% or more increase in the number of *correct* a-TDC target discoveries compared with TDC at the same FDR level of 0.05 (Supp. Fig. 3C, Supp. Fig. 19). Of course, this increase in correct discoveries does not come for free: when using TDC to control the FDR, PC used an average of 123 calibrating decoys, whereas with a-TDC that number was 134.

3.4 Analysis of Real Data

Thus far, we have described analyses that were carried out with simulated data sets. We also carried out similar analyses using the three real data sets described in Supp. Sect. 1.4. Of course, when analyzing real data we do not know which of the discoveries are false, so instead we compare the reported number of discoveries. As outlined below, the results qualitatively agree with our simulations findings.

Consistent with the analysis of our simulations using raw scores, we see that the number of both TDC and a-TDC target discoveries increases with the number of calibrating decoys (Supp. Fig. 20). Specifically, using the malaria data

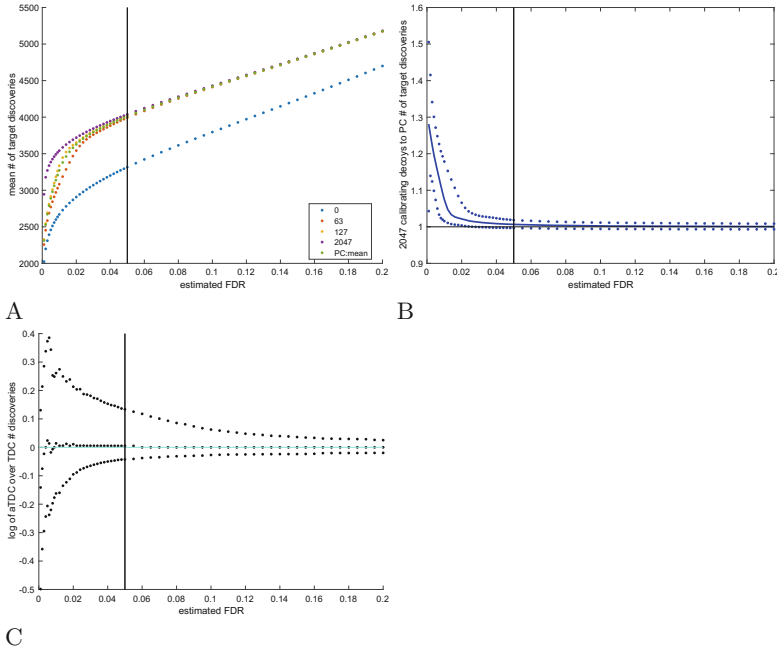


Fig. 3. Progressive calibration (PC). **A** The mean number of TDC discoveries using: 0 (raw score), 63, 127, 2047 calibrating decoys as well as the number determined by PC (63 and 127 are the number of decoys in the two cycles that bound the mean number of decoys used by PC in this experiment: 117). **B** The 0.05, 0.5, and 0.95 quantiles of the ratio of the number of TDC discoveries when using the maximal number of 2047 calibrating decoys to the number of discoveries found by PC. **C** The same quantiles of the ratio of a-TDC (10 competing decoys) to TDC *correct* discoveries (both with PC). **A–C** The vertical bars are located at 0.05, the minimal FDR level of interest for PC in this setup. All means and quantiles are taken with respect to 10K simulations using raw scores, each with 10K spectra in A–B, and 500 spectra in C, 50% native spectra in all.

and an estimated FDR level of 0.05, the mean number of TDC target discoveries gradually increases from 2434 when using no calibrating decoys to 2845 when using 10K calibrating decoys (17%, Fig. 4A). We observed a similar increase in the average number of discoveries at the same FDR level when using a-TDC: 2433 to 2844 when increasing from 0 to 10K calibrating decoys. Also consistent with our simulations, we see that regardless of how well calibrated are our scores, a-TDC reduces the variability in the number of discoveries (Fig. 4B, Supp. Fig. 21).

We also observe in the real data that PC, especially combined with a-TDC, yields near optimal power with a significantly smaller number of calibrating decoys (Supp. Fig. 22). Specifically, the average number of calibrating decoys used by PC with a-TDC (10 competing decoys) are: yeast 262 (278 with TDC),

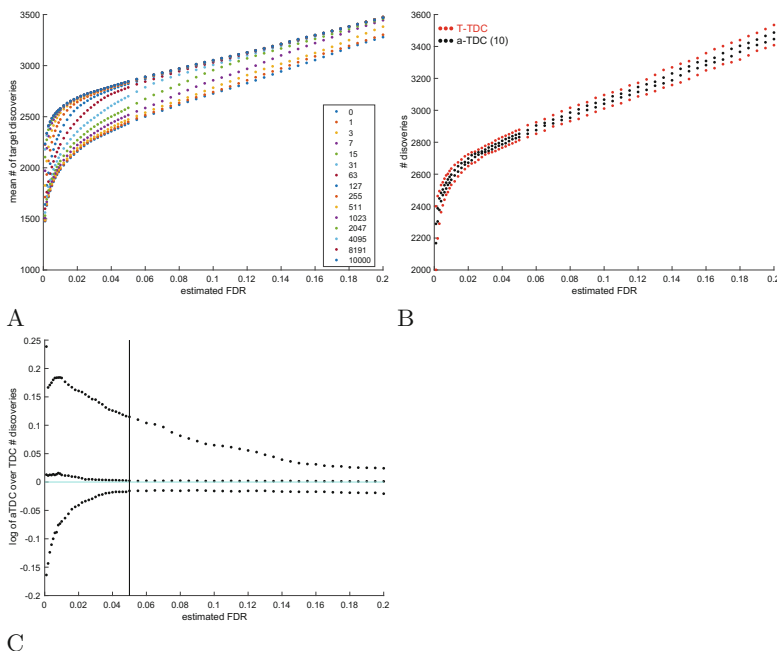


Fig. 4. Malaria data. **A** Partial calibration: the mean number of target discoveries in the malaria dataset increases with the number of calibrating decoys. **B** a-TDC is less variable than TDC: the 0.05 and 0.95 quantiles of the number of a-TDC/TDC discoveries are compared (scores are calibrated using all 10K calibrating decoys). **C** Using PC a-TDC can have more power than TDC: plotted are the log of the ratios of the 0.05, 0.5 and 0.95 quantiles of the number of a-TDC (10 competing decoys) discoveries over the corresponding numbers for TDC discoveries. **A–C** All quantiles are taken with respect to 2000 randomly drawn sets of competing decoys, as described in Supp. Sect. 1.4.

worm 235 (165), and malaria 165 (141). The corresponding power, expressed in terms of the median of the percentage of the number of discoveries made when using all 10K calibrating decoys, is: yeast 99.2% (99.3% for TDC), worm 98.8% (96.9%), malaria 99.3% (98.9%). This shows that typically PC yields almost maximal power (with respect to calibration) using a much smaller number of calibrating decoys.

Again we find that combining PC with a-TDC can reduce the number of premature stops in PC's doubling process. For example, in 100 of our 2K runs (5%) using the malaria data set, the number of TDC discoveries at FDR level 0.05 was at least 13% higher when using all 10K calibrating decoys than when TDC used PC. Applying a-TDC, the corresponding increase was only 2.8% (Supp. Fig. 22) and this translated to a 12.2% increase in the number of a-TDC discoveries in 5% of our 2K runs (Supp. Fig. 4C).

Finally, even using a-TDC, PC can sometime terminate its decoy doubling procedure earlier than we would like. For example, applying a-TDC to the worm data set using all 10K calibrating decoys, we found that in 100 of our 2K runs (5%) there are at least 7.2% more discoveries than when using the number of decoys determined by PC (Supp. Fig. 22).

4 Discussion

We offer two novel methods that rely on additional decoy databases to improve on TDC, the most commonly used FDR controlling procedure for tandem mass spectrum identification. The partial calibration procedure increases the power of TDC by using a primary-secondary score that implicitly interpolates between our original calibration procedure based on the spectrum-specific ECDF and the raw score. This primary-secondary score’s flexibility allows us to gradually enjoy the increase in power that our original calibration procedure offers while investing significantly fewer computational resources: the procedure works even with a single calibrating decoy.

As we noted before [12], we are not the first to point out the value of calibration [10]. However, our approach is different because it does not assume a specific parametric family [16, 17] or require the introduction of a new score function [9, 15]. These previous approaches are less general, and in some cases they might partially fail [12]. In contrast, our approach is generally applicable, albeit at a computational cost.

Our new a-TDC procedure helps reduce the decoy-dependent variability of TDC, both in terms of the composition of the reported list of discoveries, as well as in the associated FDR estimation. The impact of a-TDC is particularly noticeable for smaller datasets, and those are also the ones where the additional computational load of a-TDC is less prohibitive.

Interestingly, Barber and Candés recently proved that a slightly modified version of TDC, where one replaces TDC’s estimated FDR of $\widehat{\text{FDR}}(\rho) := D(\rho)/T(\rho)$ with $\widehat{\text{FDR}}(\rho) := [D(\rho) + 1]/T(\rho)$, does not suffer from the liberal bias that TDC exhibits for small FDR levels [2]. Our experiments above show that a-TDC is also able to mitigate much of the liberal bias of TDC and suggest that a-TDC does not result in the loss of power that is associated with the Barber and Candés correction.

An alternative to a-TDC to reduce variability would be to use multiple decoys in a concatenated search. In such an approach, instead of using, say, 10 decoy databases, each the same size as the target database, one can use a single decoy database that is 10 times larger than the target database. A simple adjustment to the estimated FDR makes this approach feasible; however, it has the obvious downside that the larger the decoy set is, the more target discoveries are lost. In comparison, the number of target discoveries a-TDC filters out is the average number that is filtered out by each of the individual TDC procedures (each using equal-sized sets of decoys and targets).

Note that there is some overlap in the goals of partial calibration and a-TDC: a-TDC can increase the number of discoveries in some cases, and calibration can also reduce variability. However, a-TDC will further reduce the variability even if the score is perfectly calibrated. In light of this observation, it would be particularly interesting to look into the balancing act of allocating extra decoys to a-TDC vs. partial calibration. An altogether different direction for future research on a-TDC is the theoretical asymptotic analysis of its performance as the number of competing decoys increases (and its potential connection with [8]).

We further introduce progressive calibration (PC), a method that attempts to find from the data what is the “right” amount of partial calibration we need to invest in. Based on a simple test of the increase in the number of target discoveries in each of its decoy-doubling cycles, PC can typically yield near-optimal power with significant computational savings. The current stopping criterion employed by PC is ad hoc and could benefit from a deeper analysis in the future including, for example, considering a criterion based on the change in the discovery lists themselves rather than just the number of discoveries.

We analyzed all our methods using a novel simulation procedure that allows us to sample datasets that are realistically modeled after real ones. In particular, our samples capture the uncalibrated nature of commonly used scores like XCorr. Our findings in simulated data are echoed in the analysis of three real data sets, showing that our methods can positively impact real biological analysis.

We note that here we looked at improving TDC using additional, randomly shuffled decoys. It would be interesting to compare the resulting enhanced performance with adjusting the mix-max competing FDR controlling method [11] to allow it to utilize multiple decoys as well.

As noted, TDC is the standard procedure for controlling the FDR, although it is typically carried out using reversed rather than shuffled databases. We see no inherent difference between shuffling the peptides and reversing them, and moreover, while not exactly considering the shuffling procedure, Elias and Gygi noted that [5], “Despite their differences, the four decoy databases considered here—protein reversal, peptide pseudo-reversal, random and Markov chain—yielded similar estimations of total correct identifications, and produced similar numbers of correct identifications.” More generally, the theoretical question of the applicability of TDC, which was raised in [8], has no particularly satisfying answer at this point. We currently view this as a modeling question: you cannot prove your model is suitable; rather, at best you can argue that it is. Regardless, the methods presented here improve on TDC whenever it is applicable.

Finally, we stress that these methods apply more generally than the spectrum identification problem. Indeed, as we recently argued, using TDC in this context is a special case of the problem of controlling the FDR among discoveries from searching an incomplete database [13]. In particular, our methods are relevant to controlling the FDR in peptide and protein identification, as well as in problems that arise in metagenomics sequence homology search and forensics.

References

1. Alves, G., Ogurtsov, A.Y., Yu, Y.K.: RAId_aPS: MS/MS analysis with multiple scoring functions and spectrum-specific statistics. *PLoS ONE* **5**(11), e15438 (2010)
2. Barber, R.F., Candes, E.J.: Controlling the false discovery rate via knockoffs. *Ann. Stat.* **43**(5), 2055–2085 (2015). <http://dx.doi.org/10.1214/15-AOS1337>
3. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. Roy. Stat. Soc. B* **57**, 289–300 (1995)
4. Cerqueira, F.R., Graber, A., Schwikowski, B., Baumgartner, C.: Mude: a new approach for optimizing sensitivity in the target-decoy search strategy for large-scale peptide/protein identification. *J. Proteome Res.* **9**(5), 2265–2277 (2010). pMID: 20199108. <http://dx.doi.org/10.1021/pr901023v>
5. Elias, J.E., Gygi, S.P.: Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat. Methods* **4**(3), 207–214 (2007)
6. Elias, J.E., Gygi, S.P.: Target-decoy search strategy for mass spectrometry-based proteomics. *Methods Mol. Biol.* **604**, 55–71 (2010)
7. Eng, J.K., McCormack, A.L., Yates III, J.R.: An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J. Am. Soc. Mass Spectrom.* **5**, 976–989 (1994)
8. Gupta, N., Bandeira, N., Keich, U., Pevzner, P.: Target-decoy approach and false discovery rate: when things may go wrong. *J. Am. Soc. Mass Spectrom.* **22**(7), 1111–1120 (2011)
9. Howbert, J.J., Noble, W.S.: Computing exact p-values for a cross-correlation shotgun proteomics score function. *Mol. Cell. Proteomics* **13**(9), 2467–2479 (2014)
10. Jeong, K., Kim, S., Bandeira, N.: False discovery rates in spectral identification. *BMC Bioinform.* **13**(Suppl. 16), S2 (2012)
11. Keich, U., Noble, W.S.: Improved false discovery rate estimation procedure for shotgun proteomics. *J. Proteome Res.* **14**(8), 3148–3161 (2015)
12. Keich, U., Noble, W.S.: On the importance of well calibrated scores for identifying shotgun proteomics spectra. *J. Proteome Res.* **14**(2), 1147–1160 (2015)
13. Keich, U., Noble, W.S.: Controlling the FDR in imperfect matches to an incomplete database (2016, submitted)
14. Kertesz-Farkas, A., Keich, U., Noble, W.S.: Tandem mass spectrum identification via cascaded search. *J. Proteome Res.* **14**(8), 3027–3038 (2015)
15. Kim, S., Gupta, N., Pevzner, P.A.: Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *J. Proteome Res.* **7**, 3354–3363 (2008)
16. Klammer, A.A., Park, C.Y., Noble, W.S.: Statistical calibration of the SEQUEST XCorr function. *J. Proteome Res.* **8**(4), 2106–2113 (2009)
17. Spirin, V., Shpunt, A., Seebacher, J., Gentzel, M., Shevchenko, A., Gygi, S., Sunyaev, S.: Assigning spectrum-specific p-values to protein identifications by mass spectrometry. *Bioinformatics* **27**(8), 1128–1134 (2011)

Resolving Multicopy Duplications *de novo* Using Polyploid Phasing

Mark J. Chaisson¹, Sudipto Mukherjee²,
Sreeram Kannan², and Evan E. Eichler^{1,3}(✉)

¹ Department of Genome Sciences, University of Washington,
Seattle, WA 98195, USA

`mchaisso@uw.edu`, `eee@gs.washington.edu`

² Department of Electrical Engineering, University of Washington,
Seattle, WA 98195, USA

`{sudipm,ksreeram}@uw.edu`

³ Howard Hughes Medical Institute, University of Washington,
Seattle, WA 98195, USA

Abstract. While the rise of single-molecule sequencing systems has enabled an unprecedented rise in the ability to assemble complex regions of the genome, long segmental duplications in the genome still remain a challenging frontier in assembly. Segmental duplications are at the same time both gene rich and prone to large structural rearrangements, making the resolution of their sequences important in medical and evolutionary studies. Duplicated sequences that are collapsed in mammalian *de novo* assemblies are rarely identical; after a sequence is duplicated, it begins to acquire *paralog-specific variants*. In this paper, we study the problem of resolving the variations in multicopy, long segmental duplications by developing and utilizing algorithms for polyploid phasing. We develop two algorithms: the first one is targeted at maximizing the likelihood of observing the reads given the underlying haplotypes using *discrete matrix completion*. The second algorithm is based on *correlation clustering* and exploits an assumption, which is often satisfied in these duplications, that each paralog has a sizable number of paralog-specific variants. We develop a detailed simulation methodology and demonstrate the superior performance of the proposed algorithms on an array of simulated datasets. We measure the likelihood score as well as reconstruction accuracy, i.e., what fraction of the reads are clustered correctly. In both the performance metrics, we find that our algorithms dominate existing algorithms on more than 93% of the datasets. While the discrete matrix completion performs better on likelihood score, the correlation-clustering algorithm performs better on reconstruction accuracy due to the stronger regularization inherent in the algorithm. We also show that our correlation-clustering algorithm can reconstruct on average 7.0 haplotypes in 10-copy duplication datasets whereas existing algorithms reconstruct less than one copy on average.

M.J. Chaisson and S. Mukherjee—Joint first authorship.

S. Kannan and E.E. Eichler—Joint last authorship.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 117–133, 2017.

DOI: 10.1007/978-3-319-56970-3_8

1 Introduction

Advances in single-molecule sequencing (SMS) by Pacific Biosciences (Menlo Park, CA), and Oxford Nanopore (Cambridge, UK) have recently enabled the assembly of draft *de novo* mammalian genomes [21, 35] nearing the quality of the original release of the human genome. The goal of *de novo* fragment assembly is to estimate the sequence of a genome given overlaps of relatively short sequencing reads, which is a well-studied problem. While there are multiple formulations of the fragment assembly problem [27, 31], the common challenge is that repeats in the genome longer than the length of sequenced DNA fragments make a unique reconstruction of the genome impossible [30]. Reads produced by SMS are advantageous for *de novo* assembly because the read length is at least two orders of magnitude greater than other high-throughput sequencing methods, so that genome order may be uniquely resolved when repeats are small.

SMS reads are characterized by a raw read accuracy between 75% and 90% with read lengths that follow a log-normal distribution. Initial development in *de novo* assembly of SMS reads focused on efficient methods to detect overlaps between long but noisy reads [6, 28]. Consistent with information theory [26], regions of genomes without sufficiently long repeats are contiguously assembled [24] with SMS reads. A type of repeat not well represented in human and other *de novo* SMS assemblies are *segmental duplications*: sequences 1 to 400 kilobases in length that are duplicated with at least 90% identity [18]. Segmental duplications are at the same time both gene rich and prone to large structural rearrangements, making the resolution of their sequences important in medical and evolutionary studies [37]. Comparing an SMS-based assembly of a Yoruban individual [38] to the human reference (GRCh38) reveals that only 64.2% of known segmentally duplicated bases in the human genome are present in the assembly. Due to the low raw-read accuracy of SMS, reads from different duplication paralogs are frequently merged together into the same sequence in an assembly. As a result, human assemblies of SMS reads contain large contigs with correctly resolved unique sequence and shorter contigs containing the collapse of multiple copies of a duplication into one sequence.

Segmental duplications that are collapsed in real *de novo* assemblies are rarely identical; after a sequence is duplicated, over generations it begins to acquire *paralog-specific variants* (PSVs): single-nucleotide variants that distinguish different duplication paralogs. To put this in an evolutionary context, sequences that have duplicated shortly after the human-chimpanzee divergence (6 million years ago) have acquired up to roughly one PSV per thousand bases [17]. Although the ultimate goal of *de novo* assembly is to completely resolve the sequence of a genome, an intermediate goal is to resolve the individual sequences that are collapsed in the assembly. We propose resolving sequences by estimating the number of duplications collapsed into an individual sequence in an assembly and determining the PSVs belonging to each duplication.

Given S segmental duplication paralogs of the same length containing V variants, one may represent all paralogs as an $S \times V$ matrix P with entries in $\{0, 1\}$, where each entry $P(i, j)$ is 0 if the repeat paralog i is in the consensus

state at site j , or 1 if it is a PSV. The set of N reads from all repeat paralogs may be aligned to the consensus sequence and represented as an $N \times V$ read-fragment matrix X with entries in $\{0, 1, -\}$ corresponding to consensus, variant, or absent (since reads only give information about certain positions). The goal is to reconstruct the paralog matrix P given only the read matrix X , where there are also sequencing errors creating erroneous entries in X . Let us assume that the error probability is ϵ at any position, i.e., with probability $1 - \epsilon$, the location is read correctly, and with probability ϵ , the location is read incorrectly (0 is read as a 1 and vice versa).

For $S = 2$, this problem is identical to haplotype phasing of a diploid genome [4, 25, 29]. Defining a read conflict as two overlapping reads that are non-gap and disagreeing at a site, haplotype phasing with error-free reads may be determined by grouping all conflict-free reads. To handle sequencing errors, a common formulation for haplotype phasing is minimum error correction (MEC), where a minimal number of base changes are applied to reads so that they may be partitioned into two conflict-free sets. For $S = 2$, there has also been an exact information theoretic characterization of when it is possible to phase the genome correctly [13, 36], along with efficient algorithms. This is based on connections to a problem called “community detection” [20] where the goal is to cluster users into communities based on positive or negative interactions between individuals.

When $S > 2$, this corresponds to the much less studied problem of *polyploid phasing*, which was discussed in pioneering work by Aguiar and Istrail [1]. Beginning with HapCompass [1], there has been some work on polyploid phasing using algorithms based on branch-and-extend [5], belief propagation [32] and semi-definite programming [14]. In a recent theoretical work [7], the hardness of optimizing the MEC for $S > 2$ has also been proven, indicating that algorithms for this problem need to be necessarily approximate or tailored to some assumptions. A major drawback of existing works is that they consider only $S = 3, 4$ and none have been developed, optimized, or tested for the high ploidy that is encountered in segmental duplications, where S can be potentially larger than 10, and to the low error-rate in Illumina sequencers. Thus, algorithms that are robust to the high error rates and can handle the high poly-ploidy are imperative in solving the segmental duplication problem, and in this paper, we will design such algorithms.

In particular, we propose two algorithms for solving the problem. The first approach is based on a discrete matrix-completion paradigm where the goal is to maximize the likelihood of the observed data given the underlying haplotypes. The second approach is based on a correlation-clustering framework with an inherent assumption that each haplotype has a PSV (which holds in many types of segmental duplications). By performing detailed simulations, we demonstrate the superior performance of the proposed algorithms over existing algorithms, especially in the high ploidy regime.

2 Haplotype Phasing via Discrete Matrix Completion

2.1 A Probabilistic Model

In order to represent the matrices in real-valued arithmetic, we adopt the following mapping $f: \{0, 1, -\} \rightarrow \{-1, 1, 0\}$, i.e., we represent the consensus allele as -1 , variant as 1 and undisclosed locations as 0 . To model the read matrix X , we first consider an idealized matrix M , which does not contain any noise nor does it contain any undisclosed position. If read n is sampled from the s -th paralog, then the n -th row of this matrix M is given by the s -th row of the paralog matrix, i.e., $M_n = f(P_s)$. The disclosed locations of the matrix are represented by a set Ω , which is the set of tuples (n, v) where read n contains information about variant v . Given M and Ω , the matrix X is not a deterministic function since there are independent read errors, which convert a 1 into a -1 with probability ϵ and vice versa. The probability of observing X given M and Ω is therefore given as follows,

$$\begin{aligned} \log \mathbb{P}(X \mid M, \Omega) &= \sum_{(n,v) \in \Omega} \log \mathbb{P}(X_{n,v} \mid M, \Omega) \\ &= \sum_{(n,v) \in \Omega} \log \left((1 - \epsilon) \mathbb{1}_{X_{n,v} = M_{n,v}} \right) + \log \left(\epsilon \mathbb{1}_{X_{n,v} \neq M_{n,v}} \right) \\ &= d_H(X, M) * \log(\epsilon) + (|\Omega| - d_H(X, M)) * \log(1 - \epsilon) \\ &= -d_H(X, M) * \log\left(\frac{1 - \epsilon}{\epsilon}\right) + (|\Omega|) * \log(1 - \epsilon), \end{aligned}$$

where $d_H(X, M)$ is the Hamming distance between the two matrices X and M in the locations Ω , i.e., where $X \neq 0$. Different haplotype assembly algorithms have sought to minimize varied objective criteria in order to obtain the correct clustering of reads belonging to the respective haplotypes [34]. Some of the noteworthy objectives are minimum edge removal (MER), minimum single-nucleotide polymorphism removal (MSR) and MEC. The quantity $d_H(X, M)$ is called the error criterion, and in our approach, maximizing the likelihood is equivalent to minimizing this error criterion referred to as MEC.

We observe that the ideal matrix M has repeated rows, since all rows sampled from the same paralog are identical. This implies that the matrix M has low rank. Indeed the matrix M can be factorized as the product of two matrices $M = A \cdot B$, where $A \in \mathbb{R}^{N \times S}$ with $A_{ij} \in \{0, 1\} \forall i, j$ and $B \in \mathbb{R}^{S \times V}$ with $B_{ij} \in \{-1, 1\} \forall i, j$. Each row of A is an elementary vector of length S denoting which paralog the read is from and matrix B is identical to $f(P)$ (represented in $\{-1, 1\}$).

The observed matrix X is a noisy partial observation of the low-rank matrix M , and the goal is to reconstruct the matrices A and B given X . If each read spanned the entire segmental duplication, the problem would be trivial, since similar reads can be grouped together and taking a consensus inside clusters reveals the segmental duplications. The difficulty is posed by the fact that read lengths are much smaller and do not span all variant positions.

Each read only provides partial phasing information. The resulting X matrix is thus sparse, and our goal can be formally stated as follows:

$$\operatorname{argmin}_{A,B} d_H(X, A \cdot B). \quad (1)$$

Real-valued versions of this problem have received much attention and are called the matrix completion problem. While this problem has a rich history, there is a significant difference in our setting, since the matrices A and B have structure (i.e., A has only elementary row vectors and B has binary entries) and the matrix X is ternary. We therefore have to develop new algorithms that exploit the discrete structure of the problem.

The problem of finding missing entries in a matrix arises in diverse research domains. One of the most illustrative examples is the Netflix challenge where users rate a small fraction of movies at random and the task is to predict user preferences for an unrated movie; a key assumption in this domain is that the true matrix of preferences is low-rank. While a low-rank matrix-completion problem is known to be NP-Hard, there are methods that can give provably correct reconstruction under probabilistic rather than worst-case assumptions [10, 33]. Popular techniques for this problem include convex relaxation of the rank to nuclear norm [33], singular value thresholding [9] and alternating minimization [22], all of which have theoretical guarantees as well. The key difference between these works and our problem is that they consider real-valued matrix completion, whereas, in this paper, we adapt and extend the algorithms to the discrete setting inherent to the phasing problem.

In a recent paper [8], Cai *et al.* formulate haplotype phasing as a low-rank matrix-completion problem and use structure constrained alternating minimization for obtaining the haplotypes. In the paper, they demonstrate improved performance over HapCompass for diploid and simulated polyploid data (with $S = 3, 4$). We show in this paper that while that method has good performance with small S , the performance starts deteriorating with higher S . The main reason for the deteriorating performance is the inability of the algorithm to exploit the discrete structure of the problem (for example, the algorithm does not use the fact that the B matrix is binary, instead treating it as a real-valued matrix). We alleviate this problem in the present paper by proposing an algorithm that explicitly exploits this fact.

2.2 Iterative Two-Stage Matrix Completion

Our problem stated in (1) is a hard combinatorial problem. One can design alternating minimization-based techniques for this problem, where A and B are optimized alternatively while keeping the other variable fixed. While such methods monotonically increase likelihood, they are not guaranteed to find the global optimum of the problem and display high sensitivity to initial conditions. The key idea in our approach is to first neglect the discrete nature of our problem and view it as a real-valued matrix-completion problem. We then “round” the results obtained from this real-valued matrix completion to obtain a feasible

Algorithm 1. Iterative Matrix Completion

Input : Noisy incomplete Matrix X , Rank Estimate S

- 1: Initialize $A_{init} \in \mathbb{R}^{N \times S}$ and $B_{init} \in \mathbb{R}^{S \times V}$ with sign-corrected SVD.
- 2: $e \leftarrow$ Error rate
- 3: $k \leftarrow S$
- 4: **while** $k \geq 2$ and MEC score decreases **do**
- 5: $B_{est} \leftarrow$ RealMatCom(A_{init}, B_{init}, X, k)
- 6: $A_{est}, B_{est} \leftarrow$ DiscreteMatCom(B_{est}, X, e)
- 7: Choose the best segment based on individual scores
- 8: $A_{init} \leftarrow A_{est}$
- 9: $B_{init} \leftarrow B_{est}$
- 10: $k \leftarrow k - 1$
- 11: **end while**

Output : Estimated Haplotypes B_{est}

Algorithm 2. Real-Valued Matrix Completion

- 1: **procedure** REALMATCOM(A_{init}, B_{init}, X, k)
 - 2: $A \leftarrow A_{init}$
 - 3: $B \leftarrow B_{init}$
 - 4: **while** stopping criterion not satisfied **do**
 - 5: Minimize A using projected gradient descent
 - 6: Minimize $B_{1:k}$ using projected gradient descent
 - 7: **end while**
 - 8: **return** sign(B)
 - 9: **end procedure**
-

Algorithm 3. Discrete-Valued Matrix Completion

- 1: **procedure** DISCRETEMATCOM(B_{est}, X, e)
 - 2: **while** MEC score decreases **do**
 - 3: **for** each row i of X **do**
 - 4: **for** each segment s of B_{est} **do**
 - 5: $d(i, s) \leftarrow$ Hamming distance of X_i and $B_{est,s}$ for known entries
 - 6: $W_i \leftarrow$ Window size of revealed entries of X_i
 - 7: $A_{est, is} \leftarrow (1 - e)^{W_i - d(i, s)} \cdot e^{d(i, s)}$
 - 8: **end for**
 - 9: Update overall MEC score and score for each individual segment
 - 10: Normalize $A_{est, i}$ to be a probability distribution
 - 11: **end for**
 - 12: Initialize $B_{est, new} \in \mathbb{R}^{S \times V}$ with zeros
 - 13: **for** each row i of X **do**
 - 14: $B_{est, new} \leftarrow B_{est, new} + \mathcal{P}_\Omega(A_{est, i}^T \cdot X_i)$
 - 15: **end for**
 - 16: $B_{est} \leftarrow$ sign($B_{est, new}$)
 - 17: **end while**
 - 18: **return** A_{est}, B_{est}
 - 19: **end procedure**
-

solution for the discrete problem. This rounded solution then becomes the initial value of a discrete matrix-completion routine designed based on the alternative minimization technique. While this method already has superior performance compared to existing approaches, we found that in the regime when the ploidy is high, the algorithm is able to extract some dominant haplotypes correctly while being incorrect on the other haplotypes. In order to overcome this barrier, in iteration i , we only fix the best $i - 1$ haplotypes based on the current MEC, then optimize for the rest. A schematic representation of this algorithm is depicted in Fig. 1, and the detailed pseudocode is in Algorithms 1, 2 and 3.

A standard approach in combinatorial optimization is to relax the integer constraints in the problem in order to get a real-valued optimization problem, and then to round the obtained results to get a feasible solution. We follow a similar approach here by relaxing our discrete problem to a continuous optimization problem, and along with it, we relax the objective too. Instead of optimizing according to the Hamming distance objective with the discrete constraints on A, B (see (1)), we instead minimize the Frobenius norm of the difference while at the same time assuming that A and B are real valued.

The noisy low-rank matrix completion can be formally stated as an optimization problem.

$$\min_{A, B} \frac{1}{2} \|\mathcal{P}_\Omega(A \cdot B - X)\|_F^2$$

The objective function is a squared sum of errors over all the known entries of X . $\mathcal{P}_\Omega(\cdot)$ is the projection operator and Ω is the set of known indices of X . So, $\mathcal{P}_\Omega(Z_{ij}) = Z_{ij}$ if $(i, j) \in \Omega$ and 0 otherwise. While we relax the integer constraints of the problem, we assume the following linear constraints to hold.

$$0 \leq A_{ij} \leq 1 \quad \forall i \in [N], j \in [S] \quad (2)$$

$$-1 \leq B_{ij} \leq 1 \quad \forall i \in [S], j \in [V] \quad (3)$$

Since the optimization is over unknown matrices A and B in a product form, the problem is non-convex. However, alternating minimization algorithms are known to have guaranteed reconstruction performance in certain regimes [22] and therefore we resort to using such algorithms. Thus, we first solve the optimization over A , keeping B fixed, which makes the problem convex in A and vice versa.

2.3 Projected Gradient Descent

The alternating minimization for our problem therefore can be stated as follows:

$$\begin{aligned} \min_A \quad & \frac{1}{2} \|\mathcal{P}_\Omega(A \cdot B - X)\|_F^2 \\ \text{s.t.} \quad & 0 \leq A_{ij} \leq 1 \quad \forall i, j \end{aligned}$$

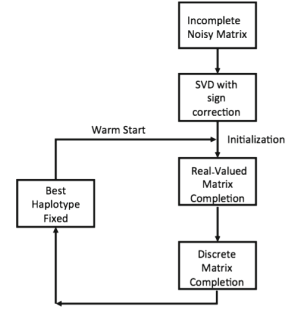


Fig. 1. The initialization and iteration workflow for discrete matrix completion.

and similarly

$$\begin{aligned} \min_B \frac{1}{2} \|\mathcal{P}_\Omega(A \cdot B - X)\|_F^2 \\ \text{s.t.} \quad -1 \leq B_{ij} \leq 1 \quad \forall i, j \end{aligned}$$

To incorporate the constraints on the variables, we use a projected gradient descent to minimize each of the convex formulations.

2.4 Initialization

Since the overall problem is non-convex, it is required to choose a suitable initialization for better performance. For this purpose, we use the singular value decomposition (SVD). This is a factorization of a $m \times n$ rectangular matrix of rank r in the form $\mathcal{U}\Sigma\mathcal{V}^T$, where \mathcal{U} is a $m \times r$ unitary matrix, Σ is a $r \times r$ diagonal matrix with non-negative diagonal entries, and \mathcal{V} is a $n \times r$ unitary matrix. The columns of \mathcal{U} and \mathcal{V} are called the left and right singular vectors, respectively. Prior theoretical results [22] suggest taking the S singular vectors of $\mathcal{P}_\Omega(X)$ as the initial guess for A and B . While this is a reasonable initialization, the signs of the singular vectors obtained from SVD decomposition may not be consistent with our problem since we require the entries of A to be strictly non-negative. We note that the signs of the singular vectors can be swapped without affecting the SVD. Therefore, in our algorithm, in order to ensure this sign consistency, we reverse the signs of certain rows of B to ensure that all columns of A have a positive sum.

$$\mathcal{P}_\Omega(X) = \mathcal{U} \cdot \Sigma \cdot \mathcal{V}^T \Gamma = \text{sign}(\mathbb{1}^T \mathcal{U}) A_{\text{init}} = \mathcal{U} * \text{diag}(\Gamma) B_{\text{init}} = (\mathcal{V} * \text{diag}(\Gamma))^T$$

For details of the projected gradient descent, we refer the reader to Appendix A.

2.5 Discrete Matrix Completion

We round the output of the real-valued matrix completion to satisfy the discrete constraints of A and B and utilize this to run a discrete alternating minimization algorithm to solve (1). The optimization of A given a fixed B is easy to solve: the basic idea is to assign each read to the segment that minimizes the Hamming distance with the read. To optimize B given a fixed A , we find the consensus of all the reads that are informative about a given position. In our algorithm, instead of having A to be a hard decision of which segment a given read belongs to, each row i of A encodes the probability that read i belongs to segment j . Therefore, while optimizing over B , we utilize the weighted consensus rather than the plain consensus of the read assignments. This procedure of refinement comes under the purview of a broader class of algorithms called the expectation maximization (EM) algorithm [16] as well as Variational Bayes [40]. The matrix A can be viewed as hidden variables encoding the membership of read fragments to duplication copies and B as the parameters for the exact underlying segments. We refer the reader to Algorithm 3 for a detailed description of the algorithm.

2.6 Choosing the Best Segment and Effective Rank Reduction

As pointed out earlier, the algorithm as stated above works well with small polyploid instances; however, in the presence of higher ploidy, the algorithm returns only the top few haplotypes correctly. For example, consider the cascading topology of repeats in Fig. 2, it is easier to resolve segment 7 but the other segments are more easily confused. Therefore, we propose an iterative algorithm, where in each iteration, the best haplotype is fixed and then the algorithm is run to optimize over possibilities of the other haplotypes. Thus, in order to do matrix completion with S haplotypes, the algorithm is iterated over $S - 1$ times. Such algorithms have a precedent even in real-valued matrix completion. For example, stagewise alternating minimization is shown to have better theoretical guarantees in [22]. In our implementation, at iteration i , the best $i - 1$ haplotypes are chosen as the ones with minimum Hamming distance from their assigned reads.

3 Haplotype Phasing with Correlation Clustering

One limitation of the MEC objective function and therefore of the discrete matrix-completion algorithm is that the ploidy must be known *a priori* or estimated. Since the MEC objective itself decreases monotonically with ploidy, it is not possible to estimate the ploidy using the MEC objective. This can be potentially remedied using regularized alternatives that account for model complexity like Akaike information criterion, Bayesian information criterion, and minimum length description. We propose an alternative algorithm here that can jointly estimate the ploidy while estimating the haplotypes themselves. This algorithm is based on a key assumption, distinct from the assumptions of the discrete matrix-completion problem: that each of the haplotypes have uniquely identifying variants. While this assumption is stronger, it can lead to stronger regularization of the problem by restricting the search space and therefore leads to better estimates, especially when the ploidy is high.

The basic idea of the algorithm is the following: each locus is represented as a vertex and reads that straddle multiple vertices create edges between the vertices that have either positive or negative weight based on whether reads share the variant or not. The goal is then to cluster the nodes into groups that share the same variant, with each cluster representing a haplotype and each locus (node) in the cluster representing a haplotype-specific variant.

To formally define our algorithm, we begin with an alternative formulation for polyploid phasing through *correlation clustering* [3], with the premise that a metric defines how similar or dissimilar two objects are, and clusters maximize the amount of similarity within each cluster and dissimilarity between clusters. Importantly, in correlation clustering the number of clusters is discovered as a result of clustering and not as a parameter.

We use an augmented form of the single-nucleotide polymorphism conflict graph \mathcal{G}_S introduced in [25], denoted $\mathcal{G}_{PSV} = (V, E)$, $E = \{E^+, E^-\}$. The construction of \mathcal{G}_{PSV} requires the fragment matrix M , and some data-dependent parameters: the expected range of coverage per haplotype c_{min} and c_{max} , and a distance d that is the maximum distance reads are expected to overlap variants.

A vertex exists for each of the columns (sites) in the fragment matrix M , connected by an edge $(u, v) \in E^+$ if u and v are overlapped by between c_{min} and c_{max} reads that are variant (e.g., 1) at both sites, or an edge $(u, v) \in E^-$ if the sites corresponding to u and v are within d bases and $(u, v) \notin E^+$. A weight $W(u, v)$ is assigned to each edge.

Correlation clustering on \mathcal{G}_{PSV} corresponds to finding clusters $C = c_1, \dots, c_n$ that minimize the sum of weighted negative edges within each cluster and weighed positive edges between clusters: $\text{Score}_{CC} = \sum_{c_i} (\sum_{(u,v) \in c_i, (u,v) \in E^-} w(u, v) + \sum_{(u \in c_i, v \notin c_i), (u,v) \in E^+} w(u, v))$, where $w(u, v)$ may reflect certainty of clustering, or more simply $w(u, v) = 1$ to count edges. Each cluster defines a set of sites that belong to a haplotype. This was shown to be APX-hard [12, 15, 19], and approximations based on linear programming (LP) were described in [12, 15]. We developed an implementation of the LP approach that was successful at clustering smaller datasets; however, the number of constraints grows with $|E|^2$, and $|E|$ grows by p^2v^2 , for ploidy p and number of PSVs v , which requires excessive resources for larger datasets.

To evaluate correlation clustering on larger datasets, we developed a simple randomized heuristic to search similar to the method of [2] for clusters that provide acceptable values for Score_{CC} that follows the steps:

1. Define clusters likely to represent repeat paralogs through a random search.
2. Merge clusters with sufficient overlap and assign nodes to unique clusters.
3. Optimize clusters by swapping vertices from adjacent clusters.

We define the *neighbor similarity* $\text{Sim}(u, v)$ of two vertices to be the number of neighbors shared between u and v connected by edges in E^+ , and $\text{Score}(V, E, c)$ to be the Score_{CC} of a single cluster c assuming all vertices $V \setminus c$ are in a separate cluster. First, clusters are formed by iteratively adding vertices neighboring a cluster as long as the neighbor similarity is sufficient and addition of the vertex decreases Score_{CC} , described in Algorithm 4.

Algorithm 4. Find cluster

```

procedure FINDCLUSTER( $V, v_i, E, s$ )
   $c \leftarrow v_i$ 
  repeat
    for all  $v \in c$  do
      for all  $n \in \text{Neighbors}(v) \notin c$  do
        if  $\text{Sim}(v, n) \geq s$  and  $\text{Score}(V, E, c \cup n) < \text{Score}(V, E, c)$  then
           $c \leftarrow c \cup n$ 
        end if
      end for
    end for
  until  $c$  has not grown
  return  $c$ 
end procedure

```

Given parameters for neighbor similarity s , a maximal number of search iterations `max_search_it` and swap iterations `max_swap_it`, and fraction cluster overlap f^{ovp} , the method `FindCluster` is used to find a set of clusters C by first initializing $C = \emptyset$, iteratively selecting a vertex $v_i \notin C$, and adding the result of `FindCluster`(V, v_i, E, s) to C until C contains all vertices in V or `max_it` iterations are reached. The resulting clusters in C are not disjoint, and so any cluster c_i with a fraction of vertices overlapping with a cluster $c_j > f^{\text{ovp}}$ is first merged into c_j , then remaining vertices belonging to more than one cluster are assigned to the largest cluster for which they are a member. Finally, the clusters are further optimized by selecting edges $(u, v) \in E^+$ where $u \in c_i$ and $v \in c_j$ and swapping u and v if this improves `ScoreCC` for up to `max_swap_it` iterations.

4 Results

We benchmarked our methods on a dataset of simulated collapsed segmental duplications. It is difficult to simulate the complex mosaic architecture of segmental duplications [23], and so we elected to use a simplified model of 100 kbp non-mosaic duplications. While this lacks the complexity of mosaic duplication architecture, the length is greater than the average duplication unit (~ 30 kbp), ensuring evaluation on challenging problems. Starting with an ancestral sequence, sequences are duplicated according to a specified tree topology T and mutation rate r , where each child node is a copy of a parent node mutated at a rate of r random single-nucleotide variant mutations per base. In real data, duplications arise with many complex histories [17, 39]. To capture the complexity of evolution, we used two classes of trees: 12 simulations from well-defined topologies such as flat, bifurcating, and cascading resulting in four to eight duplicated sequences, and 50 simulations from random tree topologies that have 10 duplicated sequences. Examples of the duplication topologies are shown in Fig. 2. The mutation rate was varied across 0.01, 0.005, 0.001, and 0.0005 mutations per base to simulate various ages of duplications. For each set of duplications we simulated $50\times$ read coverage using the `Alchemy SMS` read simulator [11], a model-based simulator that emulates a sequencing run by Pacific Biosciences, and mapped reads back to the ancestral sequence. PSV sites are detected as sites that contain between 25 and 60 non-ancestral bases.

For each of the simulated topologies and mutation rates, we evaluated the discrete matrix completion (DMP), correlation clustering (CC), and structure constrained gradient descent (SCGD). The SCGD method has been shown to outperform other previously developed methods in polyploid phasing [8].

We report MEC values by computing the sum of Hamming distance between each read and the consensus sequence for each haplotype. For CC, we assign reads to each haplotype according to the minimal Hamming distance from the read to each haplotype. For duplications simulated under models of high mutation rates (0.01 and 0.005), the DMP method is able to obtain a lower MEC score than

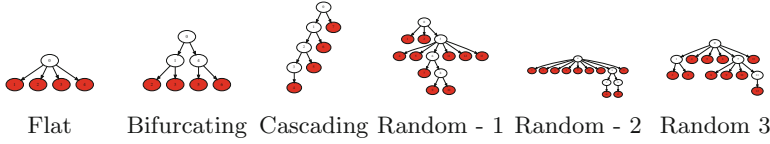


Fig. 2. Examples of topologies of duplication simulations. In total there are 12 structured trees and 50 random topologies. The divergence between any two simulated duplications is given by the mutation rate $r \times$ the shortest path between the duplications in the tree.

the other methods. Out of the 128 datasets for which every method is able to run within the time constraint set on our server, we compare the performance. We observe that CC obtains the best MEC score in 11% of the datasets, DMP 85%, and SCGD 3.9%.

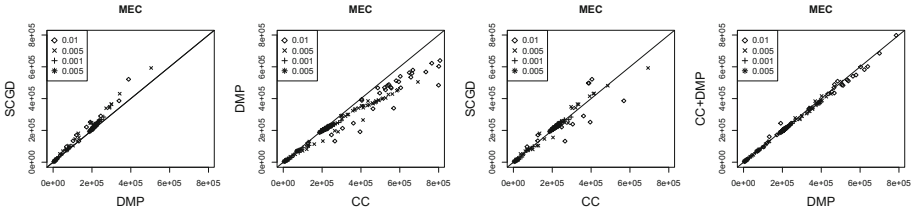


Fig. 3. MEC scores for the DMP, CC, SCGD, and CC+DMP methods. The DMP method is shown to produce haplotypes with lower MEC scores than the other methods, particularly for the high mutation rate simulations. Lower MEC score is better.

For each haplotype we count the number of reads in the haplotype that are shared with the reads simulated in each duplication and define a matching statistic as the sum of number of reads in the maximally matched duplication divided by the total number of reads. This statistic ranges between 1 for perfect reconstruction of haplotypes down to a $1/p$ when all of them are collapsed into a single reconstructed haplotype. The results are shown in Fig. 4. CC had the greatest matching score in 67.7% of the datasets, DMP 26.1% of the datasets, and SCGD on 6.1% of the datasets. Interestingly, while the CC method has a higher MEC statistic, it has a greater number of correctly partitioned reads when compared with the ground truth. We reason that this is because the CC method exploits the assumption that the positions are variant specific explicitly resulting in stronger regularization, so that even though the likelihood score is somewhat lower for CC method than other methods, it is able to fit the data more accurately. The other methods DMP and SCGD are unable to exploit this assumption and therefore overfit more severely to the data. The DMP method is sensitive to the initialization conditions for B_{est} , and so we used a solution derived by CC as initial conditions for DMP. We measured improvements on

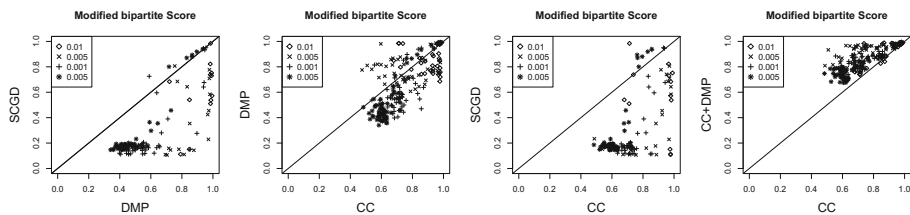


Fig. 4. Matching statistics for the DMP, CC, SCGD, and CC+DMP methods. A perfect reconstruction of haplotypes shows a score of 1, while a random assignment will score $1/\text{ploidy}$. Higher matching score is better.

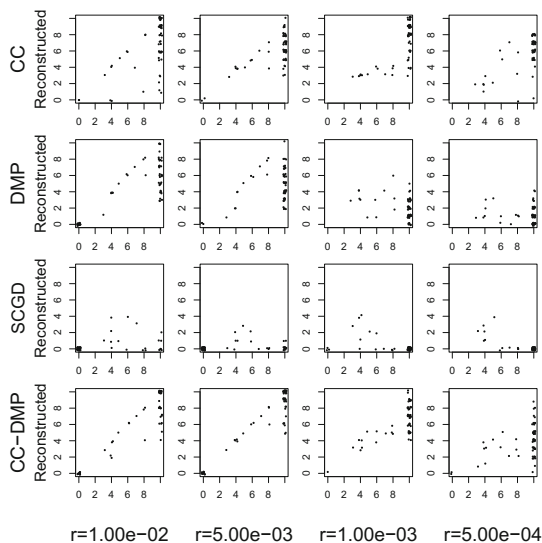


Fig. 5. Correctly assembled haplotypes for the DMP, CC, SCGD, and CC+DMP methods. Each point is the number of correctly phased genotypes per simulation, with points jittered for display.

this combination (CC+DMP) relative to DMP on MEC (Fig. 3, right), and CC for matching score. While the MEC score was largely unchanged, 220 of the 224 simulations where both CC and CC+DMP had a solution had a greater matching score in CC+DMP (Fig. 4).

We also measure a more stringent quality of reconstruction accuracy: we ask for what fraction of the true haplotypes have a reconstructed cluster to which 90% of the correct reads are assigned. Formally, for each simulated duplication we determined which haplotype had the most reads overlapping with the reads simulated from that duplication and counted how many such haplotypes had at least 90% of the reads from that haplotype reciprocally assigned to that duplication. This gives an indication of the number of copies of a segmental duplication that would be correctly assembled given the phased haplotypes.

For the 48 simulations with duplication copy number between 3 and 8, the CC method resolves 70% of duplication copies, while the DMP and SCGD methods resolve 66% and 26%, respectively (Fig. 5). For duplications of ploidy 10, the CC method resolves on average 7.0 copies of each duplication, whereas the DMP and SCGD methods resolve on average 3.3 and 0.03 copies, respectively. The CC+DMP combined method resolved 80% of duplications for simulations of copy number between 3 and 8. However, for copy number 10 duplications this provided marginal improvements over CC alone, providing solutions for 24 fewer simulations than CC, resolving on average 7.1 duplications per simulation.

5 Conclusions

The resolution of segmental duplications remains problematic in *de novo* assemblies. Deviating from the typical formulations of *de novo* assembly, we present a new formulation and two novel algorithms for resolving high-copy collapsed duplications that rely on polyploid phasing. We demonstrated that while it is possible to optimize for MEC, methods that focus on resolving clusters with unique PSVs actually resolve more duplications despite having a higher MEC value, perhaps due to less over-fitting of results to variants present in ancestral copies of a duplication. In future work we hope to improve the rank estimation for the discrete matrix-completion method, possibly leveraging the clusters discovered by correlation clustering, and characterizing the conditions under which correlation clustering converges to the correct clusters. Finally, we plan on applying these methods to resolving duplications in published human assemblies [35,38].

Acknowledgements. This work was supported, in part, by U.S. National Institutes of Health (NIH) grants 5R01HG002385-15 (E.E.E. and M.J.C.) and 5R01HG008164-02 (S.K. and S.M.). E.E.E. is an investigator of the Howard Hughes Medical Institute.

A Appendix

After each gradient step, the resultant matrix is projected onto the box. The updates for A and B are as follows:

$$\begin{aligned} \tilde{A}^{(t+1)} &\leftarrow A^{(t)} - \alpha_A \nabla_A f(A) \\ \text{Then } A_{ij}^{(t+1)} &= \begin{cases} 0, & \text{if } \tilde{A}_{ij}^{(t+1)} < 0 \\ \tilde{A}_{ij}^{(t+1)}, & \text{if } 0 \leq \tilde{A}_{ij}^{(t+1)} \leq 1 \\ 1, & \text{if } \tilde{A}_{ij}^{(t+1)} > 1 \end{cases} \\ \tilde{B}^{(t+1)} &\leftarrow B^{(t)} - \alpha_B \nabla_B f(B) \\ \text{Then } B_{ij}^{(t+1)} &= \begin{cases} -1, & \text{if } \tilde{B}_{ij}^{(t+1)} < -1 \\ \tilde{B}_{ij}^{(t+1)}, & \text{if } -1 \leq \tilde{B}_{ij}^{(t+1)} \leq 1 \\ 1, & \text{if } \tilde{B}_{ij}^{(t+1)} > 1 \end{cases} \end{aligned}$$

where $f(\cdot)$ is the objective function. The projected gradient descent allows us to incorporate additional constraints on the problem as well. If we further enforce that the sum of each row of A equals 1, then we would have the projection as $A_{ij}^{(t+1)} = \max\{0, \tilde{A}_{ij}^{(t+1)} - \nu_i\}$ where ν_i can be computed for each row i using the equality

$$\sum_{j=1}^S \max\{0, \tilde{A}_{ij}^{(t+1)} - \nu_i\} = 1$$

We allow a maximum of 50 iteration steps for minimizing each of A and B , and 100 iteration steps for alternating minimization. We exit the iterations if the change in norm is insignificant ($1e - 02$) or if the objective value change is below a tolerance ($1e - 04$). The learning rate values have to be computed in order to ensure that gradient steps do not diverge. Our choices of learning rates have been

$$\alpha_A = C \frac{\|\nabla f(A^{(t)})\|_F^2}{\|\mathcal{P}_\Omega(\nabla f(A^{(t)}) \cdot B^{(t)})\|_F^2}$$

and

$$\alpha_B = C \frac{\|\nabla f(B^{(t)})\|_F^2}{\|\mathcal{P}_\Omega(A^{(t)} \cdot \nabla f(B^{(t)}))\|_F^2}$$

where $C \in (0, 1)$.

References

1. Aguiar, D., Istrail, S.: Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics* **29**(13), i352–i360 (2013)
2. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. *J. ACM (JACM)* **55**(5), 23 (2008)
3. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. *Mach. Learn.* **56**(1–3), 89–113 (2004)
4. Bansal, V., Bafna, V.: Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics* **24**(16), i153–i159 (2008)
5. Berger, E., Yorukoglu, D., Peng, J., Berger, B.: Haptree: a novel Bayesian framework for single individual polyplotyping using NGS data. *PLoS Comput. Biol.* **10**(3), e1003502 (2014)
6. Berlin, K., Koren, S., Chin, C.-S., Drake, J.P., Landolin, J.M., Phillippy, A.M.: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat. Biotechnol.* **33**(6), 623–630 (2015)
7. Bonizzoni, P., Dondi, R., Klau, G.W., Pirola, Y., Pisanti, N., Zaccaria, S.: On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *J. Comput. Biol.* **23**, 718–736 (2016)
8. Cai, C., Sanghavi, S., Vikalo, H.: Structured low-rank matrix factorization for haplotype assembly. *J. Sel. Top. Sig. Process.* **10**(4), 647–657 (2016)
9. Cai, J.-F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **20**(4), 1956–1982 (2010)
10. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. *Commun. ACM* **55**(6), 111–119 (2012)

11. Chaisson, M.J.: <https://github.com/mchaisso/blasr>
12. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 524–533. IEEE (2003)
13. Chen, Y., Kamath, G., Suh, C., Tse, D.: Community recovery in graphs with locality (2016). arXiv preprint [arXiv:1602.03828](https://arxiv.org/abs/1602.03828)
14. Das, S., Vikalo, H.: SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genom.* **16**(1), 4 (2015)
15. Demaine, E.D., Immorlica, N.: Correlation clustering with partial information. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) APPROX/RANDOM-2003. LNCS, vol. 2764, pp. 1–13. Springer, Heidelberg (2003). doi:[10.1007/978-3-540-45198-3_1](https://doi.org/10.1007/978-3-540-45198-3_1)
16. Dempster, A.P.: Laird, N, M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. Ser. B (Methodol.)* **39**, 1–38 (1977)
17. Dennis, M.Y., Nuttle, X., Sudmant, P.H., Antonacci, F., Graves, T.A., Nefedov, M., Rosenfeld, J.A., Sajjadian, S., Malig, M., Kotkiewicz, H., et al.: Evolution of human-specific neural SRGAP2 genes by incomplete segmental duplication. *Cell* **149**(4), 912–922 (2012)
18. Eichler, E.E.: Recent duplication, domain accretion and the dynamic mutation of the human genome. *Trends Genet.* **17**(11), 661–669 (2001)
19. Emanuel, D., Fiat, A.: Correlation clustering – minimizing disagreements on arbitrary weighted graphs. In: Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 208–220. Springer, Heidelberg (2003). doi:[10.1007/978-3-540-39658-1_210](https://doi.org/10.1007/978-3-540-39658-1_210)
20. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3), 75–174 (2010)
21. Gordon, D., Huddleston, J., Chaisson, M.J.P., Hill, C.M., Kronenberg, Z.N., Munson, K.M., Malig, M., Raja, A., Fiddes, I., Hillier, L.W., et al.: Long-read sequence assembly of the gorilla genome. *Science* **352**(6281), aae0344 (2016)
22. Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. In: Proceedings of 45h Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 665–674, ACM, New York (2013)
23. Jiang, Z., Tang, H., Ventura, M., Cardone, M.F., Marques-Bonet, T., She, X., Pevzner, P.A., Eichler, E.E.: Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nat. Genet.* **39**(11), 1361–1368 (2007)
24. Koren, S., Walenz, B.P., Berlin, K., Miller, J.R., Phillippy, A.M.: Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *bioRxiv*, p. 071282 (2016)
25. Lancia, G., Bafna, V., Istrail, S., Lippert, R., Schwartz, R.: SNPs problems, complexity, and algorithms. In: Heide, F.M. (ed.) ESA 2001. LNCS, vol. 2161, pp. 182–193. Springer, Heidelberg (2001). doi:[10.1007/3-540-44676-1_15](https://doi.org/10.1007/3-540-44676-1_15)
26. Motahari, A., Ramchandran, K., Tse, D., Ma, N.: Optimal DNA shotgun sequencing: noisy reads are as good as noiseless reads (2013). arXiv preprint [arXiv:1304.2798](https://arxiv.org/abs/1304.2798)
27. Myers, E.W.: Toward simplifying and accurately formulating fragment assembly. *J. Comput. Biol.* **2**(2), 275–290 (1995)
28. Myers, G.: Efficient local alignment discovery amongst noisy long reads. In: Brown, D., Morgenstern, B. (eds.) WABI 2014. LNCS, vol. 8701, pp. 52–67. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-44753-6_5](https://doi.org/10.1007/978-3-662-44753-6_5)

29. Patterson, M., Marschall, T., Pisanti, N., Iersel, L., Stougie, L., Klau, G.W., Schönhuth, A.: WhatsHap: haplotype assembly for future-generation sequencing reads. In: Sharan, R. (ed.) RECOMB 2014. LNCS, vol. 8394, pp. 237–249. Springer, Cham (2014). doi:[10.1007/978-3-319-05269-4_19](https://doi.org/10.1007/978-3-319-05269-4_19)
30. Pevzner, P.A.: Dna physical mapping and alternating Eulerian cycles in colored graphs. *Algorithmica* **13**(1–2), 77–105 (1995)
31. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. *Proc. Nat. Acad. Sci.* **98**(17), 9748–9753 (2001)
32. Puljiz, Z., Vikalo, H.: Decoding genetic variations: communications-inspired haplotype assembly. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**(3), 518–530 (2016)
33. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.* **52**(3), 471–501 (2010)
34. Schwartz, R., et al.: Theory and algorithms for the haplotype assembly problem. *Commun. Inf. Syst.* **10**(1), 23–38 (2010)
35. Seo, J.-S., Rhie, A., Lee, S., Sohn, M.-H., Kim, C.-U., Hastie, A., Cao, H., Yun, J.-Y., Kim, J., et al.: De novo assembly and phasing of a Korean human genome. *Nature* **538**, 243 (2016)
36. Si, H., Vikalo, H., Vishwanath, S.: Haplotype assembly: an information theoretic view. In: 2014 IEEE Information Theory Workshop (ITW), pp. 182–186. IEEE (2014)
37. Stankiewicz, P., Lupski, J.R.: Genome architecture, rearrangements and genomic disorders. *Trends Genet.* **18**(2), 74–82 (2002)
38. Steinberg, K.M., Graves-Lindsay, T., Schneider, V.A., Chaisson, M.J.P., Tomlinson, C., Huddleston, J.L., Minx, P., Kremitzki, M., Albrecht, D., Magrini, V., et al.: High-quality assembly of an individual of Yoruban descent. *bioRxiv*, p. 067447 (2016)
39. Usher, C.L., Handsaker, R.E., Esko, T., Tuke, M.A., Weedon, M.N., Hastie, A.R., Cao, H., Moon, J.E., Kashin, S., Fuchsberger, C., et al.: Structural forms of the human amylase locus and their relationships to SNPs, haplotypes and obesity. *Nat. Genet.* **47**(8), 921–925 (2015)
40. Welling, M., Kurihara, K.: Bayesian k-means as a maximization-expectation algorithm (2007)

A Bayesian Active Learning Experimental Design for Inferring Signaling Networks

Robert Osazuwa Ness^{1,2(✉)}, Karen Sachs³, Parag Mallick³, and Olga Vitek²

¹ Department of Statistics, Purdue University, West Lafayette 47907, USA
nessr@purdue.edu

² College of Science, College of Computer and Information Science,
Northeastern University, Boston 02115, USA

³ School of Medicine, Stanford University, Palo Alto 94305, USA

Abstract. Machine learning methods for learning network structure, applied to quantitative proteomics experiments, reverse-engineer intracellular signal transduction networks. They provide insight into the rewiring of signaling within the context of a disease or a phenotype. To learn the causal patterns of influence between proteins in the network, the methods require experiments that include targeted interventions that fix the activity of specific proteins. However, the interventions are costly and add experimental complexity.

We describe a active learning strategy for selecting optimal interventions. Our approach takes as inputs pathway databases and historic datasets, expresses them in form of prior probability distributions on network structures, and selects interventions that maximize their expected contribution to structure learning. Evaluations on simulated and real data show that the strategy reduces the detection error of validated edges as compared to an unguided choice of interventions, and avoids redundant interventions, thereby increasing the effectiveness of the experiment.

Keywords: Machine learning · Active learning · Causal inference · Bayesian network · Probabilistic graphical models · Biological networks

1 Introduction

Signaling networks describe chains of protein interactions that determine how cells process signals from their environment. The deregulation of signaling networks occurs under many conditions, e.g. in diseases such as cancer [30], gene knockouts, or introduction of a drug. The patterns of such deregulation can be inferred from quantitative proteomic experiments conducted under the conditions of interest, using causal inference and Bayesian networks [12, 38].

In these investigations, signaling is induced with a stimulus perturbation, and a measurement technology acquires information on the activity of signaling proteins [42]. Bulk experiments quantify aggregate signaling activity across a sample. In contrast, single cell technologies provide cell-level resolution of signaling activity. For example, in flow cytometry cells are chemically fixed, and

intracellular signaling proteins are tagged with fluorescently-labeled antibodies. The cytometer then records the antibodies’ fluorescence in individual cells, each reflecting the relative abundance of signaling proteins in different states of enzymatic activity [32]. Similarly, in mass cytometry (CyTOF) experiments, intracellular signaling proteins are tagged with heavy-metal isotopes, and the mass spectrometer records the mass-to-charge ratio of the charged isotope tags [1].

Causal Bayesian networks represent signaling proteins as nodes, and regulatory relationships as directed edges. It interprets a network as a topological map of the underlying signaling network. By comparing the structure inferred under a condition to *canonical pathways* in sources such as KEGG and Reactome, we can learn the patterns of network deregulation. Data repositories, such as CytoBank [4], provide *historic data*, which can be incorporated in the analysis and interpretation of experimental results [12, 45]. This prior information is especially important for higher throughput experiments because it helps eliminate spurious correlations and false discoveries of relationships between proteins [29].

To distinguish causal relationships from statistical associations, causal network inference requires *targeted interventions* on some proteins [9, 29], e.g., using small-molecule inhibitors that block a protein’s enzymatic activity. An insufficient set of interventions results in only a partially causally oriented network [31]. At the same time, increasing the number of interventions increases the complexity of the experiment and the cost. This cost is wasted when targeted interventions redundantly orient the same edges.

In this paper, we propose a strategy for optimal design of bulk or single-cell proteomic experiments aiming at causal inference. The design prioritizes targeted interventions, and provides a criterion to stop adding interventions. It combines prior knowledge in the form of canonical pathways imported from sources such as KEGG [22] with historic data. The strategy outputs a sequence of interventions that we call a “batch”, i.e. a minimal subset of candidate interventions that contributes maximal causal information given the available data. We then describe an active learning framework, that iterates between selecting interventions and acquiring data to obtain a fully inferred causal network. To the best of our knowledge, this is the first active learning approach to experimental design for inference of signaling networks. Accurate signaling network structure inference depends on the “right” data; the proposed design approach provides experimentalists with a roadmap for generating that data.

2 Background

2.1 Directed Graphs as Causal Models of Signaling

A causal Bayesian network denotes a set of p signaling proteins with p nodes $V = \{v_1, \dots, v_p\}$. The nodes are variables representing levels of signaling activity of the proteins. For example, v_1 can take discrete signaling states “active” or “inactive”, or continuous values quantifying the abundance of a protein form. The model expresses causal relations between nodes with a directed acyclic graph structure (DAG) G . The edge direction in the DAG represents the causal effect of

a change in the signaling state of a parent node on the state of the child. The DAG is best interpreted as a snapshot of a dynamic system [20]. This interpretation is strongest when the signaling response has reached some quasi-steady-state.

Each node in the DAG has a conditional probability distribution given its parents. It is a probabilistic representation of the regulatory influences of the parents on the child [31]. A key advantage of the probabilistic interpretation is that it encodes *conditional independence*, i.e. the probability that the state of a protein is independent of the state of all its upstream proteins, if we know (i.e. condition on) the states of its direct parents. This allows us to ignore the correlation between a protein and proteins more than one step upstream.

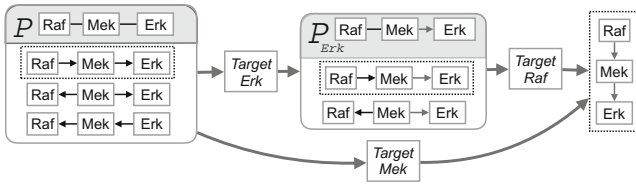


Fig. 1. Illustration of DAG equivalence classes. The DAG $\text{Raf} \rightarrow \text{Mek} \rightarrow \text{Erk}$ is the “ground truth” canonical MAPK signaling pathway, which we seek to learn by causal inference. The left box shows the equivalence class P , represented by the PDAG $\text{Raf} - \text{Mek} - \text{Erk}$. The PDAG contains three DAGs, all statistically indistinguishable in absence of interventions. The cardinality of P is 3. The middle box shows the PDAG P_{Erk} , obtained after an intervention on Erk . P_{Erk} is a subclass of P that has eliminated $\text{Raf} \leftarrow \text{Mek} \leftarrow \text{Erk}$. The cardinality of P_{Erk} is 2. The right box shows the single ground truth DAG obtained after an additional intervention on Raf . An alternative single intervention on Mek simultaneously compels the direction of both edges, and is more effective at discovering the ground truth.

From the statistical perspective, the goal of causal inference is to infer the DAG structure representing the signaling network, using associations between proteins as input. However, statistical associations are not sufficient to orient the edges in the DAG [29]. We illustrate this with the simple 3-protein canonical MAPK signaling pathway $\text{Raf} \rightarrow \text{Mek} \rightarrow \text{Erk}$. Imagine that the structure of the pathway is unknown, and needs to be inferred. A causal inference algorithm would (1) detect pairwise statistical correlations between abundances of each pair of the three proteins, pointing to the three candidate edges, (2) test Raf and Erk for conditional independence, given the state of Mek , and (3) in presence of conditional independence, eliminate the edge between Raf and Erk . After that, additional interventions are required to orient the edges between Raf and Mek , and between Mek and Erk . The left box in Fig. 1 illustrates that, in absence of interventions, the ground truth is statistically indistinguishable from the other two causally incorrect DAGs.

A set of statistically indistinguishable DAGs form a *Markov equivalence class* P , comprised of DAGs with same edges but varying orientations, which have

equal statistical likelihood for the dataset [24, 31]. A Markov equivalence class is represented with a partially directed acyclic graph (PDAG). The directed edges in a PDAG have the same orientation in all the DAGs in \mathcal{P} . The undirected edges in the PDAG have varying directions in the DAGs, and therefore represent edges with uncertain causality. Figure 1 illustrates PDAGs and DAGs in the MAPK pathway example. The cardinality of a PDAG is defined as the number of its DAGs.

Targeted interventions proceed by fixing the state of a node, such that it does not vary with that of its parents [9, 24, 31]. Fixing the node introduces an additional constraint, and eliminates members of the equivalence class that fail to satisfy this constraint. For example, in Fig. 1 a small inhibitor fixes Erk’s enzymatic activity state to “off”, such that the activity of Erk becomes independent of the state of Mek. The intervention fails to regulate the activity of Mek, and therefore eliminates the DAG with an edge $\text{Erk} \rightarrow \text{Mek}$.

The reduced equivalence class is formally defined as a *transition-sequence Markov equivalence class*, i.e. the equivalence class after a sequence of “transitions” (interventions) [43]. Each additional intervention orients more edges in a PDAG, and a sufficiently large set of interventions compels all the edges. In Fig. 1, interventions on Erk and Raf eliminated all but the ground truth from the equivalence class.

As the example illustrates, a batch of interventions targeting Erk and Raf would reveal the ground truth DAG. However, so would a batch containing a single intervention on Mek. The goal of this work is to identify batches of interventions, which reveal the most causality while minimizing the number of interventions they contain, and by extension the experimental complexity and cost.

2.2 Bayesian Inference of Causal Networks

This work focuses on a Bayesian approach to learning causal PDAG representations of signaling, where experimentalists (1) start with background knowledge about the signaling network, such as likely pathways or motifs, (2) use the background knowledge to construct a *prior* distribution on graph structures, (3) collect experimental measurements of the signaling states of proteins, and (4) estimate a *posterior* distribution on structures based on the prior and the experimental measurements. The Bayesian approach is advantageous, as it reveals the “rewiring” of signaling between conditions by examining differences between the prior and the posterior distributions.

More formally, the approach uses the background knowledge δ to construct a prior probability distribution of possible DAG structures $\pi(G|\delta)$ in the space of possible structures \mathbb{G} . The experimentalist collects a data set D of measurements on the proteins $V \in G$, acquired under a batch of targeted interventions S . A statistical likelihood function $p(D|S, G)$ quantifies the likelihood that the observations were generated from a graph G . Finally, inference of the DAG structure relies on a posterior probability distribution $\pi(G|D, S, \delta)$

$$\pi(G|D, S, \delta) \propto p(D|S, G)\pi(G|\delta) \quad (1)$$

In many biological applications, inferring a single DAG with high posterior probability is not of the main interest. Instead, we are interested in local features of the graph, such as the presence of particular edges or network motifs. This is expressed through a function f on a graph that quantifies the feature of interest, and through its posterior expectation $E\{f|D, S, \delta\}$ across all graphs.

$$E\{f|D, S, \delta\} = \sum_{\mathbb{G}} f(G)\pi(G|D, S, \delta) \quad (2)$$

For example, if f is an indicator of the presence of an edge in the network, then $E\{f|D, S, \delta\}$ is the posterior probability of the presence of that edge. In this work, our feature of interest quantifies the causal insight provided by an intervention.

Bayesian inference of DAG structures relies on a *Bayesian scoring function* $Score(G, D, S, \delta)$ that takes as arguments a DAG, a dataset quantifying protein activity, a set of interventions, and the structured prior knowledge. It returns values proportional to the posterior distribution $\pi(G|D, S, \delta)$. Algorithms searching for DAGs with high $Score(G, D, S, \delta)$ must explore the combinatorially large search space of possible DAGs [25, 26, 36]. For computational tractability, some algorithms approximate the posterior probabilities $\pi(G|D, S, \delta)$ (see [24, 40] for background, and [8, 18] for an example). Since the full search space over all possible DAGs \mathbb{G} is combinatorially large, the common practice is to sample a set of DAGs from their posterior distribution through a random process such as bootstrap ([14, 21]) or MCMC [15], and keep a sample of high scoring networks. $E\{f|D, S, \delta\}$ is then approximated as

$$E\{f|D, S, \delta\} \approx \sum_{G \in \Omega} f(G) \frac{Score(G, D, S, \delta)}{\sum_{G, G \in \Omega} Score(G, D, S, \delta)} \quad (3)$$

where Ω denotes a sample of high scoring DAGs.

2.3 PDAG Representation of Uncertainty in Causal Effects

Experiments with incomplete sets of interventions lack information to fully infer the causal orientation of the edges in a DAG. In order to characterize this uncertainty, algorithms take a single DAG and determine the space of DAGs having the same conditional independence structure, topology and likelihood $p(D|S, G)$, but different edge orientations as the input DAG [5, 43]. These DAGs form a *Markov equivalence class*. The algorithms return a PDAG, which represents the class by preserving the shared edge structure, while presenting edges with conflicting orientations as undirected.

In Bayesian inference of causal networks, we are interested in Markov equivalent graphs with not only the same likelihood, but also the same posterior probability and the same score $Score(G, D, S, \delta)$ [5]. As seen from Eq. 1, Markov

equivalent graphs have the same posterior only if they have the same prior probability $\pi(G|\delta)$. This last condition does not hold when the prior probabilities encode available causal information, such that for some edges orientation in one direction is more probable than the other. A contribution of this manuscript is an implementation of a DAG-to-PDAG conversion algorithm that accounts for informative prior probabilities of causal direction of edges, and outputs a PDAG representing a class of DAGs that are Markov equivalent and have the same posterior and score.

2.4 Active Learning for the Optimal Design of Causal Network Inference Experiments

In the context of causal inference from experiments, active learning is the task of including targeted interventions in the design, to optimize the inference of edge orientation. For example, in Fig. 1 an intervention on Mek compels both edges in the graph, and is thus more valuable than an intervention on Erk, which only orients one edge.

Previous work used active learning to distinguish members of statistically equivalent graphs [10, 17, 27, 33, 43]. However, these approaches work with either a single PDAG, or a selection of highly-likely PDAGs. The approach in this manuscript differs by working with the entire probability distribution of PDAGs, characterizing each PDAG by its posterior probability of containing the causal truth. Also use a Bayesian approach to represent graph uncertainty. There approach uses.

Alternative Bayesian approaches to active learning of causal networks also exist. Some of these require the experimentalists to represent their background knowledge in terms of topological orderings [28, 44], i.e. an ordering of nodes such that the “from” node for every edge occurs earlier in the ordering of the “to” node. Cho et al. [7] use Gaussian Bayesian networks with a normal-inverse gamma prior. In contrast, this manuscript represents background knowledge in terms of probabilities of edge presence and orientation. This more intuitive approach simplifies the experimentalists’ work with pathways. The proposed active learning approach also works with Gaussian continuous, discrete, and general structural assumptions on the Bayesian network joint probability distribution.

Finally, the proposed approach is similar in spirit to other methods in the bioinformatics literature that use historic data to inform experimental design. E.g., Rossel and Muller [35] used a sequential Bayesian method to plan sample size. Guan et al. [16] used available data to find optimal orderings of high-throughput experiments. King et al. [23] constructed a “robot scientist” that applied an active-learning strategy to functional genomics. To our knowledge, this manuscript is the first to apply active learning to inferring the structure of cell signaling pathways.

3 Methods

3.1 Prior Knowledge for Causal Graph Structure Learning

Quantifying Causal Knowledge with Edge Probabilities. We propose to use probabilities of edge presence and edge orientation as a means of modeling signaling events. A set of signaling proteins, represented by nodes in V , has up to $\frac{|V|(|V|-1)}{2}$ possible edges. *Presence probability* of an edge between nodes $\{u, v\}$, denoted $P(u - v)$ or $\bar{\pi}_{uv}$ as a shorthand, quantifies the confidence that the edge is present in G . *Orientation probability* for the edge from u to v , denoted $P(u \rightarrow v|u - v)$ or $\vec{\pi}_{uv}$ as a shorthand, is the conditional probability of this orientation, *given* that the edge is present. Since only two orientations are possible, $P(u \rightarrow v|u - v) = 1 - P(u \leftarrow v|u - v)$. The goal of targeted interventions is to resolve the orientations of the edges, i.e. coerce orientation probabilities towards 0 or 1.

Let δ be a set of edge probabilities $\delta_{u,v}$, where

$$\delta_{u,v} \stackrel{\text{def}}{=} \{\bar{\pi}_{uv}, \vec{\pi}_{uv}\} \quad (4)$$

In Bayesian setting, we wish to use the edge probabilities to quantify prior causal knowledge. Let $\bar{I}_{uv}(G)$ be an indicator function for the presence an edge between u and v in G , and $\vec{I}_{uv}(G)$ be an indicator function for the orientation of the edge from u to v . We map edge probabilities δ to a probability distribution on DAG structures using an *edge-wise prior*

$$\pi(G|\delta) = c \prod_{uv \in G} (1 - \bar{\pi}_{uv})^{1 - \bar{I}_{uv}(G)} (\bar{\pi}_{uv} \vec{\pi}_{uv})^{\bar{I}_{uv}(G) \vec{I}_{uv}(G)} \quad (5)$$

where c is a normalizing function on the space of graphs that corrects for the acyclicity constraint [3].

When nothing is known about the presence or orientation of the edges, we specify the *uninformative* edge probabilities [40], where

$$\bar{\pi}_{uv} \approx \frac{1}{2} + \frac{1}{2(|V| - 1)}, \quad \text{and} \quad \vec{\pi}_{uv} = \frac{1}{2} \quad (6)$$

The intuition behind Eq. 6 is that two nodes are more likely to be linked in a small network than in a large network. Therefore, the uninformative presence probability approaches .5 as network size increases. The uninformative orientation probability is .5 for either direction. These uninformative edge-wise priors correspond to the marginal probabilities of an edge in the case when there is a uniform probability distribution on the space of graphs [40].

Upon conducting an experiment with interventions S and collecting a dataset D , the next step is to *update* the DAG probability distribution with condition-specific information in the data using Bayes rule

$$\pi(G|D, S, \delta) \propto p(D|S, G)\pi(G|\delta) \quad (7)$$

Note that the Bayes rule is agnostic of the process that selected the interventions in S . S can be selected by any approach, such as applying the available inhibitors, or using the proposed active learning approach below.

The updated probability distribution $\pi(G|D, S, \delta)$ maps back to edge probabilities using the approach in Eq. 2. Let $f(\cdot)$ in Eq. 2 be an indicator function $\bar{I}_{uv}(G)$ for the presence, and an indicator function $\vec{I}_{uv}(G)$ for the orientation, of an edge between nodes u and v in a DAG. Then, the updated presence and orientation probabilities of an edge after observing data D is defined as

$$\begin{aligned}\bar{\pi}_{uv|D,S} &= \sum_{\mathbb{G}} \bar{I}_{uv}(G) \pi(G|D, S, \delta) \\ \vec{\pi}_{uv|D,S} &= \frac{1}{\bar{\pi}_{uv}} \sum_{\mathbb{G}} \vec{I}_{uv}(G) \pi(G|D, S, \delta)\end{aligned}\quad (8)$$

In other words, these are average frequencies of edge presence and orientation over all the DAGs, weighted by the posterior probabilities of the DAGs.

Incorporating Pathway Knowledge and Historic Data. When prior information is available, we propose to construct informative edge probabilities $\delta_{u,v}$. In the simplest case, a directed edge between u and v in the canonical pathway is viewed as a hypothesis that an edge linking these nodes is also present under the condition of interest, and is oriented from u to v . Denoting the set of edges in the canonical pathway as \mathbb{K} , the background knowledge $\delta_{u,v}$ is defined as in Eq. 4 where

$$\begin{aligned}\bar{\pi}_{uv} &= \begin{cases} \sim 1 & \text{if } u - v \in \mathbb{K} \\ \sim 0 & \text{otherwise} \end{cases} \\ \vec{\pi}_{uv} &= \begin{cases} \sim 1 & \text{if } u - v \text{ is oriented } u \rightarrow v \in \mathbb{K} \mid u - v \in \mathbb{K} \\ \sim 0 & \text{if } u - v \text{ is oriented } u \leftarrow v \in \mathbb{K} \mid u - v \in \mathbb{K} \end{cases}\end{aligned}\quad (9)$$

The notation ~ 1 (probability near 1) and ~ 0 (probability near 0) emphasizes that the Bayesian approach avoids the boundary probabilities of 0 and 1.

In some cases, experimentalists may wish to use alternative specifications. For example, in absence of canonical pathway information it may be inappropriate to assign $\bar{\pi}_{uv} \sim 0$ to each edge, and subjective edge probabilities may be a better choice. For edges where no assessments can be made, we use the uninformative edge probabilities in Eq. 6.

In addition to incorporating prior knowledge from canonical pathways, we also seek to make use of information in historic datasets. We define historic data D_0 as previous experiments, which quantified the activity of the proteins in the same network, under the same signaling conditions as the pending causal inference experiment, but lacking targeted interventions. We update the canonical knowledge $\pi(G|\delta)$ with the condition-specific information in the historic data

$$\pi(G|D_0, \delta) \propto p(D_0|G) \pi(G|\delta) \quad (10)$$

Here $p(D_0|G)$ is the likelihood that the historic data came from the graph G , and $\pi(G|D_0, \delta)$ is the updated distribution on graph structures, which now captures the full state of our prior information before the interventions.

Sampling DAGs from a DAG Distribution. Similarly to Eqs. 2 and 3, the proposed Bayesian inference on causal networks relies on sampling a set of DAGs Ω from a distribution $\pi(G|D, S, \delta)$. Our implementation uses Bayesian bootstrap sampling from a distribution of DAGs [14, 21] with random graph starts [19] and greedy search, and the posterior distributions are derived using Bayesian Dirichlet approximation [18]. When the signal-to-noise ratio in the historic data is high and/or the edge-wise prior is informative, the sampling concentrates on a smaller set of most probable DAGs. When the signal-to-noise ratio is low, weight is distributed more evenly among graphs, and the sampling must cover a larger number of graphs with similar $\pi(G|D, S, \delta)$.

Representing Uncertainty in Causal Effects with PDAGs. We incorporate the informative edge orientation probabilities in δ to express the uncertainty in edge orientation using PDAGs. We view the conversion of a DAG to a PDAG as a the function f in Eqs. 2 and 3. Applying these equations requires that DAG members of the same equivalence class have the same posterior probability, otherwise different instances of the same PDAG would have different probabilities, i.e. the same f would have different probabilities in Eq. 2 and scores in 3. Current conversion algorithms are not compatible with edge-wise prior probabilities.

Algorithm 1 is a DAG-to-PDAG conversion algorithm that incorporates informative edge-wise prior probabilities. It starts with a DAG from Ω . Next, every directed edge is converted to an undirected edge if it meets three conditions. The first two conditions are the same as in the prior literature [5, 43]. First (lines 4 and 10 in Algorithm 1), reversing edge direction should not change the number of immoral v-structures (i.e., 3-node motifs with one child and two parents, with no edge between parents). Second (line 6 of Algorithm 1), the child node of the edge should not be targeted by an intervention in S . In this manuscript we introduce a third condition (lines 8 in Algorithm 1), stating that the edge should have an uninformative prior orientation probability of .5. These conditions create an equivalence class P and its PDAG, where all the members have the same value for $\pi(G|D_0, \delta)$. If the algorithm is applied to two Markov equivalent DAGs with different causal information in their informative edge-wise priors, two different PDAGs are returned.

3.2 Bayesian Active Learning with Causal Information Gain

Overview. The strategy for selecting optimal interventions is overviewed in Fig. 2. The active learning algorithm takes as input a sample of DAGs from a DAG distribution, and a set of candidate interventions. The algorithm sequentially evaluates the expected causal information gain of the interventions, and outputs a minimally-sized batch of interventions that maximizes the expected information gain. We detail the components of the algorithm below.

Algorithm 1. *DAG to PDAG Algorithm*

Inputs: A DAG G , an (optional) set of selected intervention targets S , a set of edge probabilities δ .

```

1: procedure PDAG( $G, S, \delta$ )
2:   for edge  $e$  in  $G$  do
3:     if  $e$  is in an immoral v-structure then
4:       Fix direction of  $e$ 
5:     if  $e$ 's child is targeted by  $S$  then
6:       Fix direction of  $e$ 
7:     if  $e$ 's orientation probability in  $\delta \neq .5$  then
8:       Fix direction of  $e$ 
9:   for edge  $e$  in  $G$  do
10:    if  $e$  is not fixed then
11:      if reversing  $e$ 's direction
12:        will not add a new v-structure
13:        or introduce a cycle then
14:          Make  $e$  undirected
15:    $P \leftarrow G$ 
16:   return ( $P$ )
  
```

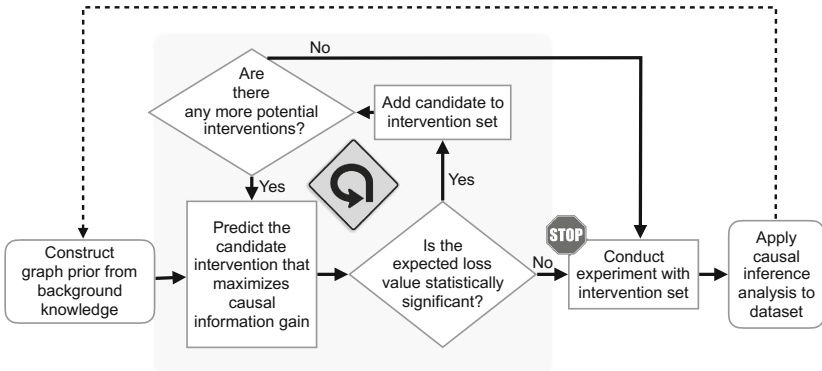


Fig. 2. Overview of the proposed method. A probability distribution of possible graph structures is constructed from canonical pathways and historic data. Interventions are iteratively added to the design until the expected causal information gain we can expect from an additional intervention becomes small. We then stop adding interventions to the design and use the newly acquired data to infer the causal network.

Defining the Causal Information Gain of an Intervention. Suppose that the true causal DAG G were known. Let $S \subseteq V$ denote a batch of candidate interventions. As discussed in Sect. 2.3, a PDAG is derived directly from a DAG’s topology and a set of interventions, and can be determined before collecting data. Therefore, we could devise an algorithm $H(G, S)$ that first determines a PDAG, then counts the number of oriented edges in the PDAG, i.e. the number of oriented edges in G that could be inferred from data with interventions S .

Suppose that we consider an additional intervention on node v , which leads to $H(G, \{S, v\})$ edges correctly oriented edges. We define the causal *information gain* $IG(G, S, v)$ of the intervention on v as the increase in correctly oriented edges, i.e. $IG(G, S, v) = H(G, \{S, v\}) - H(G, S)$. $IG(G, S, v)$ is non-negative, and can be zero if v fails to orient any edges beyond those oriented by S . Note that an equivalent definition of information gain is the reduction in the number of unoriented edges. This definition parallels the information theory notation of entropy, where the information gain is viewed as entropy reduction.

Selecting Interventions that Maximize the Expected Information Gain. Of course in practice the true causal DAG G is unknown. Therefore, similarly to Eq. 2 we consider the *expected information gain*, which averages the information gain over all possible graphs, weighted by their prior distribution $\pi(G|D_0, \delta)$

$$EIG_{S,v} = \sum_{G} IG(G, S, v) \pi(G|D_0, \delta) \quad (11)$$

Moreover, similarly to Eq. 3, we approximate the expected information gain as

$$EIG_{S,v} \approx \sum_{G \in \Omega} IG(G, S, v) \frac{Score(G, D_0, S, v, \delta)}{\sum_{G; G \in \Omega} Score(G, D_0, S, v, \delta)} \quad (12)$$

where Ω denotes a sample of high scoring DAGs, and *Score* is a Bayesian scoring function returning a value proportional to $\pi(G|D_0, \delta)$. We then select the candidate v that maximizes the approximated expected information gain. Algorithm 2 details these steps. Note that in Bayesian decision theory, $-IG$ is a loss function, and we select the candidate v that minimizes the expected loss [2].

Algorithm 2. *Expected Information Gain*

Inputs: A set of DAGs Ω , Bayesian scoring function *Score*, a set of pre-selected intervention targets S , a candidate for next intervention v , and a set of edge probabilities δ .

- 1: **procedure** EIG(Ω , *Score*, S , v , δ)
 - 2: Initialize array IG of size $|\Omega|$
 - 3: **for** i in $1:|\Omega|$ **do**
 - 4: $P_S \leftarrow$ PDAG(G_i, S, δ)
 - 5: $P_{S,v} \leftarrow$ PDAG($G_i, \{S, v\}, \delta$)
 - 6: $H_S \leftarrow$ num. of directed edges in P_S
 - 7: $H_{S,v} \leftarrow$ num. of directed edges in $P_{S,v}$
 - 8: $IG_i \leftarrow H_{S,v} - H_S$
 - 9: **return** WeightedMean(IG , *Score*)
-

Algorithm 3. *Active Learning*

Inputs: A set of DAGs Ω , Bayesian scoring function $Score$, a set U of candidates for next intervention, and a set of edge probabilities δ .

Parameter: α , stopping criterion for the information gain.

```

1: procedure ACTIVELEARNING( $\Omega, Score, U, \alpha, \delta$ )
2:    $S \leftarrow \text{null}$ 
3:   while length( $U$ ) > 0 do
4:     TopCandidate  $\leftarrow \text{null}$ 
5:     MaxEIG  $\leftarrow 0$ 
6:     for node  $v \in U$  do
7:       EIG $_{S,v} \leftarrow \text{EIG}(\Omega, Score, S, v, \delta)$ 
8:       if EIG $_{S,v} > \text{MaxEIG}$  then
9:         TopCandidate  $\leftarrow v$ 
10:        MaxEIG  $\leftarrow \text{EIG}_{S,v}$ 
11:       if  $\neg \text{Stop}(\text{TopCandidate}, \Omega, \alpha)$  then
12:          $S \leftarrow \text{cat}(S, \text{TopCandidate})$ 
13:          $U \leftarrow U[-\text{TopCandidate}]$ 
14:       else
15:         return ( $S$ )
    
```

Starting Point, Iterations and Stopping Criteria. The proposed active learning strategy is summarized in Algorithm 3. It starts with the empty set of selected interventions $S = \emptyset$, a set of candidate interventions U , and a set of DAGs Ω . For each intervention $v \in U$ and each DAG G in Ω , the $\text{EIG}(\Omega, Score, S, v)$ algorithm calculates P_S , $P_{S,v}$ and $IG(G, S, v)$, and returns the expected information gain. The candidate with the maximum expected information gain is added to the batch S . In the next iteration, the expected information gain for the remaining candidates is evaluated, while accounting for the effect of the interventions that are already in the batch.

After a certain point, additional interventions to the batch become counterproductive. For example, in Fig. 1 an intervention on Mek would orient the *Mek – Erk* edge. Including an additional intervention on Erk would provide no additional information. In the case of Fig. 1 the true graph is known, and we stop adding interventions when the information gain is 0. Since in real-life situations the structure of the true DAG is unknown, we stop adding interventions when the probability that at least some information gain occurs is below a parameter α . Similarly to Eqs. 3 and 12, the probability that at least some information gain occurs is

$$q_{S,v} = \sum_{G \in \Omega} I_{\{IG(G,S,v) > 0\}} \frac{Score(G, D_0, S, v, \delta)}{\sum_{G \in \Omega} Score(G, D_0, S, v, \delta)} \quad (13)$$

where v is the candidate that maximizes EIG, and $I_{\{\cdot\}}$ is the indicator function. We add v to the set of interventions if $q_{S,v} > \alpha$, and stop if $q_{S,v} \leq \alpha$. Higher values of α will result in a smaller intervention batch. Setting probability threshold α to ~ 0 (i.e., stopping when the probability of at least some information gain is near 0) is equivalent to stopping when expected information gain is near 0.

When the signal-to-noise ratio in the historic data is low, there is greater weight on graphs where the most optimal intervention candidates have no information gain. This leads to the triggering of the stopping criteria with a smaller batch of interventions. Thus when there is more uncertainty in the data, the procedure avoids the risk of wasteful use of interventions, instead favoring running the experiment with a smaller batch and then relying on the new experimental data to evaluate unused interventions.

3.3 Inference of Causal Network from Data Acquired Post-intervention

The next step in the investigation is to apply the selected interventions, and collect a new dataset D . The new dataset updates Eq. 7 as

$$\pi(G|S, D_0, D, \delta) \propto p(D|S, G)\pi(G|D_0, \delta) \quad (14)$$

The updated edge probabilities are obtained as in Eqs. 8 and 9. They balance the canonical representation of the signaling with the condition-specific signaling behavior quantified under the condition of interest, after the interventions. A large deviation of the posterior edge probability from the prior in δ indicates network rewiring or deregulation.

The active learning procedure is iterative in nature, in that the results of the intervention experiment can be viewed as a new instance of historic data. They can inform the selection of new interventions by substituting $\pi(G|S, D_0, D, \delta)$ in Eqs. 11 and 12, and repeating the overall procedure.

3.4 Implementation and Computational Complexity

The proposed strategy is implemented in an open-source R package *bninfo*, available on Github. The edge-wise prior is constructed as a data frame in R, either manually or through an interface to the API of KEGG provided by *bninfo*. Historic data is represented as a data frame and pre-selected interventions S as an array. The package *bninfo* implements the algorithms for converting a DAG and a edge-wise to PDAG, calculating the expected gain, and selecting the optimal batch of interventions. The Bayesian network structure learning is performed with the existing R package *bnlearn* [39].

The main scalability bottleneck in the proposed strategy is the selection of interventions in the while loop in Algorithm 3. The complexity of calculating the expected information gain for a single intervention (Algorithm 2 and line 7 in Algorithm 3) is in the order of the number of edges in the input DAG. However, the interventions in a batch are selected sequentially (i.e., the selection of the j th member of the batch depends on the $j - 1$ previously selected interventions). Therefore, the selection of j candidate interventions requires up to $j!$ calculations of the expected gain. The running time can be reduced by parallelization of Algorithm 2, or by limiting the number of candidate interventions. Moreover, sampling Ω from a distribution of DAGs has well-known scalability challenges

in Bayesian literature. In the worst case, the computational time scales exponentially with the number of proteins. Bayesian bootstrap sampling can be split among nodes on a cluster, and sped up by parallelization. For the datasets described below, the generation of intervention batches took 70 min (17 protein DREAM4 network with 14 candidate interventions and 500 sampled graphs) and 20 min (11 protein T-cell network with 5 candidate interventions and 500 sampled graphs) on a 16 node cluster.

3.5 Metrics Used for Performance Evaluation

We evaluated the proposed strategy using datasets containing some notion of “ground truth”, i.e. situations where the true structure of the causal graph is known. We used the proposed active learning strategy to determine an optimal intervention batch S . To evaluate the performance of S , we considered the data D that would be experimentally acquired with the selected interventions. We then inferred causal networks from D , and derived posterior edge probabilities as described in Sect. 3.3. Finally, we evaluated whether S can lead to network inference that correctly detects edges in the “ground truth” graph.

We evaluated the performance of edge detection using two metrics. The first is the *true positive rate of edge detection*, i.e. the proportion of correctly detected edges among the edges in the ground truth network [14]. Our cutoff for edge’s detection is presence and orientation probabilities greater than uninformative prior probabilities described in Eq. 6. The second metric is the L_1 edge error, which quantifies the overall probability of prediction error, i.e. the probability of either discovering a false edge or missing a true edge [28,44]. Given the set of interventions S and the ground truth network, the L_1 edge detection error is defined as

$$\begin{aligned} L_1(G, S) = & \sum_{u,v} \vec{I}_{uv}(G)(1 - \vec{\pi}_{uv|D,S}) \\ & + (1 - \vec{I}_{uv}(G))(\vec{\pi}_{uv|D,S}) \\ & + (1 - \bar{I}_{uv}(G))(\bar{\pi}_{uv|D,S}) \end{aligned} \quad (15)$$

where $\vec{I}_{uv}(G)$ and $\bar{I}_{uv}(G)$ are as in Eq. 9.

4 Datasets

There are currently no publicly available datasets that implement active learning approach to causal network inference. We therefore use two publicly available datasets, adapted to provide a measure of “ground truth”.

4.1 DREAM4 Network

“Ground Truth” Network. The 17-node network in Fig. 3A was used in the DREAM4 Predictive Signaling Network Challenge [34]. The network contains

canonical pathways downstream of four receptors (dark grey nodes in Fig. 3A): two inflammatory (TNF α , IL1 α), one insulin (IGF-I), and one growth factor receptor (TGF α). We use this network to evaluate the relative advantages of active learning, and the importance of the use of prior information.

Prior Information Regarding the Network Structure. We used as the background information the fact that TNF α , IL1 α , IGF-I and TGF α are receptors, i.e. proteins that receive signals from the environment, and activate downstream proteins. This presents causal information, in that the remaining proteins in the pathway are downstream of the receptors.

Historic Data. We use the DREAM4 challenge as a historic dataset. The challenge provided antibody-based measurements (sandwich immunoassays with the Luminex xMAP platform) from hepatocellular carcinoma cell lines (HepG2), which quantified the activity levels of the signaling proteins at bulk (i.e., non-single-cell) resolution. The dataset is comprised of samples with 5 stimulus conditions, namely no stimulus, stimulus on TNF α , on IL1 α , on IGF1, and on TGF α . The dataset also includes 5 intervention conditions, namely no inhibition, inhibition on ikk, on mek12, on pi3k, and on p38. In total there were 25 samples, one for each stimulus and intervention pair.

We processed the historic dataset as follows. We imputed missing values using a neural network model that predicted missing values of a protein given the values of the protein’s neighbors in the ground truth network. Several proteins from the pathway (map3k7, ras, map3k1, and mkk4) were not quantified in the historic dataset. Approaches exist for learning Bayesian network structure with hidden variables [6, 13], but these are beyond the scope of this manuscript. So to eliminate this artifact we applied a model that predicts a protein’s values given the values of its parents in the model and common biochemical assumptions on signaling dynamics [41], and used the model to predict the values for the hidden nodes. The publicly available challenge data is normalized to the 0–1 range, we then discretized the quantification values to binary on/off variables using .5 as a cutoff.

Candidate Interventions. All the non-receptor nodes in the network were considered as candidates for interventions. Due to the small number of samples and inhibitions, it was not possible to set aside a portion of this dataset for performance evaluation. Therefore, we fit a causal network model to the challenge data and the ground truth network, and used the model fit to generate synthetic post-intervention datasets. The simulation mimicked the design of the challenge data, in that it contained one biological sample for each intervention. We then evaluated the performance of the selected interventions on these synthetic datasets.

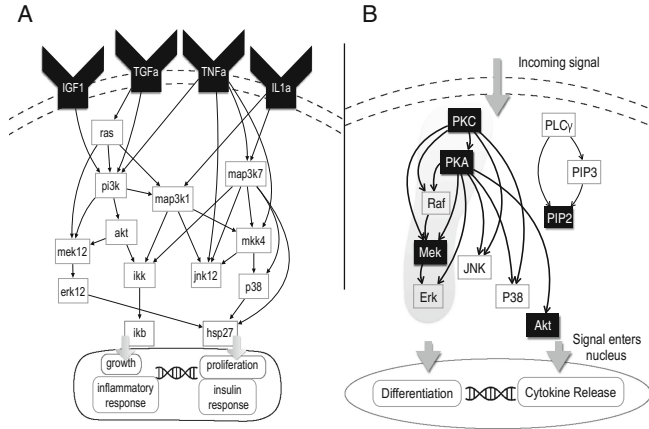


Fig. 3. The “ground truth” networks in the experimental datasets. A: The DREAM4 Predictive Signaling Network Challenge network. Dark nodes indicate receptors. B: Native T-cell signaling network in response to antigen. The edges in the PKC \rightarrow Raf \rightarrow Mek \rightarrow Erk cascade, and the edge PKA \rightarrow Raf belong to the canonical MAPK pathway. Dark nodes indicate targets of experimental interventions.

4.2 Flow Cytometry Measurements of T-Cell Signaling

“Ground Truth” Network. The network in Fig. 3B contains 11 phosphoproteins and phospholipids involved in the native CD4+ T-cell signaling response to antigen, and their canonical edges. The network was used to validate causal inference from an experimental dataset [38], and has been subsequently used as a benchmark in multiple causal inference studies [9, 11].

Prior Information Regarding the Network Structure. We used as the background knowledge the edges in the canonical MAPK pathway. Although CD4+ T-cell signaling has been extensively studied, we assumed that no prior information is available regarding the remaining edges. This assumption allowed us to compare the case of a minimally informative prior to the case of a uninformative prior, and focus performance evaluation on detection edges that were not addressed in the background knowledge.

Historic Data. The experimental dataset in [38] contains single cell fluorescence-based quantifications of 11 phosphoproteins and phospholipids in human primary naive CD4+ T-cells. These analytes are downstream of the receptors CD3 and CD28 that provide co-stimulatory signals required for T-cell activation. We used as the “historic data” a portion of this dataset that was acquired without any targeted interventions. This portion of the dataset contained 11672 cells.

Candidate Interventions. The dataset in [38] also contained single cell quantifications, acquired after activating or inhibiting five signaling proteins (dark nodes in Fig. 3B). These five proteins were considered as candidates for interventions in this manuscript. The post-intervention experimental datasets were used to compare the information gain projected in our approach with the actual information gain after the interventions.

5 Results

5.1 Informative Prior Edge Probabilities Reduced the Required Number of Interventions in the DREAM4 Dataset

In the DREAM4 dataset, we compared (1) random ordering of interventions, and (2) proposed active learning strategy with uninformative prior probabilities of edge presence (Eq. 6), and (3) the same active learning strategy, but with informative priors. The informative priors encoded the fact that the receptor nodes have upstream positions in the network. All the receptor-originating edges were assigned the prior orientation probability of ~ 1 , all receptor-terminating edges were assigned the presence probability of ~ 0 , and the remaining edges were assigned the uniformed prior presence probability in Eq. 6. All the non-receptor nodes in the network were candidate targets for interventions. In order to exhaust the information in the prior and historic data, the active learning approach with both priors used the most liberal stopping criteria for growing a batch. It only stopped adding interventions when all the additional candidates has expected information gain of ~ 0 .

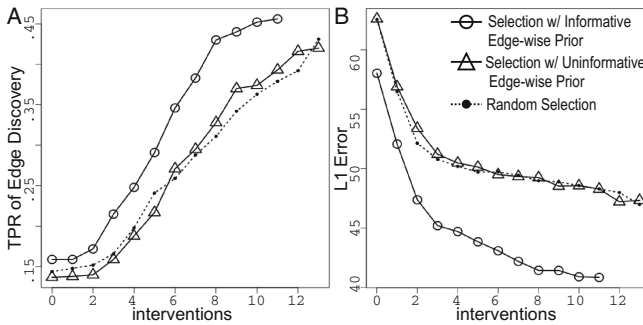


Fig. 4. Performance of the proposed strategy on the DREAM4 dataset. Dotted line: uninformative edge-wise priors, randomly selected interventions. Solid line with triangles: uninformative edge-wise priors, interventions selected with active learning. Solid line with circles: informative edge-wise priors, interventions selected active learning. A: True Positive Rate (TPR) of detecting ground truth edges. B: L_1 error of detecting ground truth edges.

Figure 4 summarizes the results. With uninformative edge-wise priors, random selection of interventions performed similarly to the proposed active learning in both True Positive Rate of edge detection and L_1 loss. Both metrics depend on correct detection of edge presence as well as edge orientation. The contribution to edge detection of the added samples provided by each intervention experiment overshadowed the selection strategies prioritization of interventions that better resolve edge orientation.

At the same time, the results demonstrate the efficiency gain in selecting the interventions, brought by encoding the knowledge of receptor identities into the prior. For example, while with the uninformative prior the true positive rate of more than .35 could be achieved with on average 9 interventions, the informative prior only required on average 6 interventions. Table 1 shows the specific interventions that were selected at a conservative cutoff ($q_{S,v} \leq 0.01$). The results indicate that informative edge-wise priors are important for such bulk experiments with a small number of replicate samples. The prior knowledge removed uncertainty in edge presence, increasing the contribution of improved detection of edge orientation to overall performance.

Table 1. Intervention targets selected by active learning in the DREAM4 dataset. The informative prior edge probabilities required a smaller intervention batch.

Prior	Intervention targets selected by active learning
Informative	(1) hsp27, (2) mek12, (3) map3k1, (4) jnk12, (5) pi3k, (6) mkk4, (7) ikk, (8) akt, (9) p38, (10) erk12, (11) ikb
Uninformative	(1) jnk12, (2) hsp27, (3) mkk4, (4) mek12, (5) pi3k, (6) map3k1, (7) map3k7, (8) ras, (9) ikk, (10) akt, (11) ikb, (12) p38, (13) erk12

5.2 The Ordering of T-Cell Interventions by Active Learning Matched Their Contribution to Causal Inference

As above, for the T-cell dataset we compared (1) random ordering of interventions, (2) proposed active learning strategy with uninformative prior probabilities of edge presence (Eq. 6), and (3) the same active learning strategy, but with informative priors. In the latter case we assumed the prior knowledge of the canonical MAPK pathway, and assigned a high prior probability (~ 1) to the edges in the PKC \rightarrow Raf \rightarrow Mek \rightarrow Erk cascade, and to the edge PKA \rightarrow Raf. The remaining potential edges were assigned the uninformative prior probability of both presence and orientation. We then considered the five interventions in [38] as the set of candidate interventions.

Figure 5 summarizes the results. Selection with an uninformative edge-wise prior did not outperform random selection of interventions in terms of True Positive Rate of detecting edges. However, it had a smaller L_1 error for the first three selected interventions. With this experiment, the intervention datasets

served not just to resolve causality, but to improve edge detection by adding variation in signaling activity not present in preceding datasets. As with the DREAM4 data, this led to performance gains due to improved edge detection overshadowed gains owed to the selection strategy.

In contrast, active learning with the edge-wise prior encoding the MAPK edges outperformed random selection both in terms of greater True Positive Rate, and smaller L_1 error. Table 2 shows the specific interventions that were selected at a conservative cutoff ($q_{S,v} \leq 0.01$). For example, while with the uninformative prior the true positive rate of .75 could be achieved with on average 5 interventions, the informative prior only required on average 3 interventions.

The network structure provides some insight into the role of informative edge-wise priors in improving the performance. Since the orientation probability of an edge depends on the orientation probability of its neighbors, the edge-wise prior reduced error by reducing uncertainty in the orientation of edges neighboring the MAPK edges. In addition, the edge-wise prior enabled the causal inference procedure to down-weight graphs where the MAPK edges were not present or had the wrong orientation, increasing sensitivity. Finally, additional causal information encoded in the prior made interventions on Mek and Akt interventions less useful, enabling the stopping criteria to eliminate them from the batch.

Table 2. Intervention targets selected by active learning in the T-cell dataset. The use of an informative edge-wise prior eliminates two interventions from the batch.

Prior	Intervention targets selected by active learning
Informative	(1) PKA, (2) PKC, (3) PIP2
Uninformative	(1) PKA, (2) PIP2, (3) PKC, (4) Akt, (5) Mek

6 Discussion

Our results showed that an active learning strategy, combined with informative priors, is the most effective at suggesting the smallest batch of target interventions for inference of causal networks. It optimizes the causal information in the intervention experiments, while controlling the experiment time and cost.

The background information comes in the form of prior knowledge on the presence and orientation of edges in the system available, e.g., in pathway databases such as KEGG, as well as from historic datasets available, e.g. from repositories such as Cytobank. The proposed strategy can be used with any level of prior information or uncertainty. When prior information and historic data indicate an intervention is potentially wasteful, it will tend to omit it from the batch, electing rather to reserving it for potential use in future experiments.

The active learning strategy in this manuscript suggests interventions based on the prior information and on the topology of the network. In practice, however, other factors can affect the utility of an intervention. For example, small

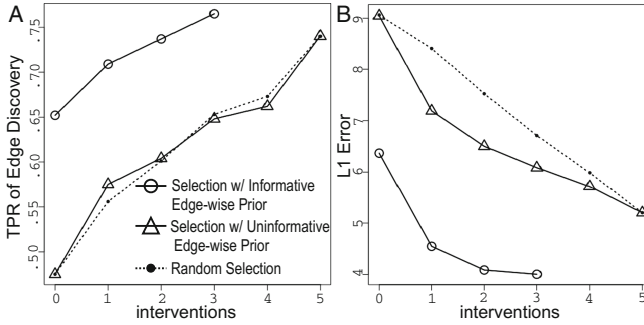


Fig. 5. Performance of the proposed strategy on the T-cell signaling dataset. Lines and panels are as in Fig. 4.

molecule inhibitors vary both in cost and efficacy. Their inhibitory effects may only occur with some probability, and may also have off-target effects. The proposed framework can be easily extended to incorporate this type of “soft intervention” [9], as well as cost considerations into the expected information gain. Alternatively, it can produce a batch of interventions with a fixed pre-specified number of targets, while selecting the targets with the most expected causal information gain.

The proposed methodology relies on signaling network modeling by means of direct acyclic graphs. In practice, however, cell signaling often displays feedback loops. This can be addressed by refining the biological interpretation of the graph structures. In particular, cycles in signaling often involve regulatory feedback loops that involve transcription. In this case, an activation of the signaling pathways causes transcription, which then results in the translation of new signaling proteins which then change the initial signaling pathway. Since the time scale of signaling is in seconds and minutes, and the time scale of transcription is in minutes and hours, collecting the data at an appropriate time point can help resolve the confounding between the initial causal effect of the signaling, and the feedback.

This work addressed both the inference of causal networks from bulk experiments and single cell experiments. Inferring an edges orientation depends of first detecting its presence, and since many edges between the proteins can potentially exist, bulk experiments often lack replicates to confidently detect the edges. Therefore, in bulk experiments it is often useful to add interventions not only to improve the detection of edge orientation, but also to improve the detection of edge presence through increased sample size.

In contrast, single cell experiments characterize protein signaling activity in thousands to millions of individual cells, and increase our confidence in the inferred edge. If the historic data was collected under a minimal number of conditions, intervention data may resolve both edge orientation and presence by virtue of adding variation in signaling activity. Moreover, single-cell experiments do not eliminate the need for true biological replication. The proposed approach

can be extended to population level inference by modeling subjects as additional nodes in the network [37].

Overall, we believe that the proposed strategy is an important step towards an informed experimental design for inference of causal networks, and advocate its practical use.

Acknowledgements. We thank M. Scutari for guidance in using the R package *bnlearn*. This work was supported in part by the NSF CAREER award DBI-1054826, and by the Sy and Laurie Sternberg award to OV.

References

1. Bandura, D.R., Baranov, V.I., Ornatsky, O.I., Antonov, A., Kinach, R., Lou, X., Pavlov, S., Vorobiev, S., Dick, J.E., Tanner, S.D.: Mass cytometry: technique for real time single cell multitarget immunoassay based on inductively coupled plasma time-of-flight mass spectrometry. *Anal. Chem.* **81**(16), 6813–6822 (2009)
2. Berger, J.O.: *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media, New York (2013)
3. Castelo, R., Siebes, A.: Priors on network structures. Biasing the search for Bayesian networks. *Int. J. Approx. Reason.* **24**(1), 39–57 (2000)
4. Chen, T.J., Kotecha, N.: Cytobank: providing an analytics platform for community cytometry data analysis and collaboration. In: Fienberg, H.G., Nolan, G.P. (eds.) *High-Dimensional Single Cell Analysis*. Current Topics in Microbiology and Immunology, vol. 377, pp. 127–157. Springer, Heidelberg (2014). doi:[10.1007/82_2014_364](https://doi.org/10.1007/82_2014_364)
5. Chickering, D.M.: A transformational characterization of equivalent Bayesian network structures. In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 87–98. Morgan Kaufmann Publishers Inc. (1995)
6. Chickering, D.M., Heckerman, D.: Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Mach. Learn.* **29**(2–3), 181–212 (1997)
7. Cho, H., Berger, B., Peng, J.: Reconstructing causal biological networks through active learning. *PLoS ONE* **11**(3), e0150611 (2016)
8. Cooper, G.F., Yoo, C.: Causal discovery from a mixture of experimental and observational data. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 116–125. Morgan Kaufmann Publishers Inc. (1999)
9. Eaton, D., Murphy, K.P.: Exact Bayesian structure learning from uncertain interventions. In: *International Conference on Artificial Intelligence and Statistics*, pp. 107–114 (2007)
10. Eberhardt, F., Glymour, C., Scheines, R.: On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables (2012). arXiv preprint: [arXiv:1207.1389](https://arxiv.org/abs/1207.1389)
11. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical LASSO. *Biostatistics* **9**(3), 432–441 (2008)
12. Friedman, N.: Inferring cellular networks using probabilistic graphical models. *Science* **303**(5659), 799–805 (2004)
13. Friedman, N., et al.: Learning belief networks in the presence of missing values and hidden variables. *ICML* **97**, 125–133 (1997)

14. Friedman, N., Goldszmidt, M., Wyner, A.: Data analysis with Bayesian networks: a bootstrap approach. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 196–205. Morgan Kaufmann Publishers Inc. (1999)
15. Friedman, N., Koller, D.: Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Mach. Learn.* **50**(1–2), 95–125 (2003)
16. Guan, Y., Dunham, M., Caudy, A., Troyanskaya, O.: Systematic planning of genome-scale experiments in poorly studied species. *PLoS Comput. Biol.* **6**(3), e1000698 (2010)
17. He, Y.-B., Geng, Z.: Active learning of causal networks with intervention experiments and optimal designs. *J. Mach. Learn. Res.* **9**(11), 2523–2547 (2008)
18. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: the combination of knowledge and statistical data. *Mach. Learn.* **20**(3), 197–243 (1995)
19. Ide, J.S., Cozman, F.G.: Random generation of Bayesian networks. In: Bittencourt, G., Ramalho, G.L. (eds.) *SBIA 2002. LNCS (LNAI)*, vol. 2507, pp. 366–376. Springer, Heidelberg (2002). doi:[10.1007/3-540-36127-8_35](https://doi.org/10.1007/3-540-36127-8_35)
20. Ideker, T., Krogan, N.J.: Differential network biology. *Mol. Syst. Biol.* **8**(1), 565 (2012)
21. Imoto, S., Kim, S.Y., Shimodaira, H., Aburatani, S., Tashiro, K., Kuhara, S., Miyano, S.: Bootstrap analysis of gene networks based on Bayesian networks and nonparametric regression. *Genome Inform.* **13**, 369–370 (2002)
22. Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M., Tanabe, M.: Kegg as a reference resource for gene and protein annotation. *Nucleic Acids Res.* **44**(D1), D457–D462 (2016)
23. King, R.D., Whelan, K.E., Jones, F.M., Reiser, P.G.K., Bryant, C.H., Muggleton, S.H., Kell, D.B., Oliver, S.G.: Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* **427**(6971), 247–252 (2004)
24. Koller, D., Friedman, N., *Models, P.G.:* Principles and Techniques. MIT Press, Cambridge (2009)
25. Korb, K.B., Nicholson, A.E.: *Bayesian Artificial Intelligence*. CRC Press, Boca Raton (2010)
26. Margaritis, D.: Learning Bayesian network model structure from data. Ph.D. thesis, U.S. Army (2003)
27. Meganck, S., Leray, P., Manderick, B.: Learning causal Bayesian networks from observations and experiments: a decision theoretic approach. In: Torra, V., Narukawa, Y., Valls, A., Domingo-Ferrer, J. (eds.) *MDAI 2006. LNCS (LNAI)*, vol. 3885, pp. 58–69. Springer, Heidelberg (2006). doi:[10.1007/11681960_8](https://doi.org/10.1007/11681960_8)
28. Murphy, K.P.: Active learning of causal Bayes net structure (2001)
29. Ness, R.O., Sachs, K., Vitek, O.: From correlation to causality: statistical approaches to learning regulatory relationships in large-scale biomolecular investigations. *J. Proteome Res.* **15**, 683–690 (2016)
30. Pawson, T., Warner, N.: Oncogenic re-wiring of cellular signaling pathways. *Oncogene* **26**(9), 1268–1275 (2007)
31. Pearl, J.: *Causality: Models, Reasoning and Inference*, vol. 29. Cambridge University Press, Cambridge (2000)
32. Perez, O.D., Nolan, G.P.: Simultaneous measurement of multiple active kinase states using polychromatic flow cytometry. *Nat. Biotechnol.* **20**(2), 155–162 (2002)
33. Pournara, I., Wernisch, L.: Reconstruction of gene networks using Bayesian learning and manipulation experiments. *Bioinformatics* **20**(17), 2934–2942 (2004)

34. Prill, R.J., Saez-Rodriguez, J., Alexopoulos, L.G., Sorger, P.K., Stolovitzky, G.: Crowdsourcing network inference: the DREAM predictive signaling network challenge. *Sci. Signal.* **4**(189), mr7 (2011)
35. Rossell, D., Müller, P.: Sequential stopping for high-throughput experiments. *Biostatistics* **14**(1), 75–86 (2013)
36. Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M., Edwards, D.D.: *Artificial Intelligence: A Modern Approach*, vol. 2. Prentice Hall, Upper Saddle River (2003)
37. Sachs, K., Gentles, A.J., Youland, R., Itani, S., Irish, J., Nolan, G.P., Plevritis, S.K.: Characterization of patient specific signaling via augmentation of Bayesian networks with disease and patient state nodes. In: 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 6624–6627. IEEE (2009)
38. Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D.A., Nolan, G.P.: Causal protein-signaling networks derived from multiparameter single-cell data. *Sci. (N.Y., NY)* **308**(5721), 523–529 (2005)
39. Scutari, M.: Learning Bayesian networks with the bnlearn R package. *J. Stat. Softw.* **35**(3), 1–22 (2010)
40. Scutari, M.: On the prior and posterior distributions used in graphical modelling. *Bayesian Anal.* **8**(3), 505–532 (2013)
41. Terfve, C., Cokelaer, T., Henriques, D., MacNamara, A., Goncalves, E., Morris, M.K., van Iersel, M., Lauffenburger, D.A., Saez-Rodriguez, J.: CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. *BMC Syst. Biol.* **6**(1), 1 (2012)
42. Terfve, C., Saez-Rodriguez, J.: Modeling signaling networks using high-throughput phospho-proteomics. In: Goryanin, I., Goryachev, A. (eds.) *Advances in Systems Biology. Advances in Experimental Medicine and Biology*, vol. 736, pp. 19–57. Springer, New York (2012). doi:[10.1007/978-1-4419-7210-1_2](https://doi.org/10.1007/978-1-4419-7210-1_2)
43. Tian, J., Pearl, J.: Causal discovery from changes. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 512–521. Morgan Kaufmann Publishers Inc. (2001)
44. Tong, S., Koller, D.: Active learning for structure in Bayesian networks. In: *International Joint Conference on Artificial Intelligence*, vol. 17, pp. 863–869. Lawrence Erlbaum Associates Ltd. (2001)
45. Werhli, A.V., Husmeier, D.: Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat. Appl. Genet. Mol. Biol.* **6**(1), 15 (2007)

***BBK** (Branch and Bound over K^*): A Provable and Efficient Ensemble-Based Algorithm to Optimize Stability and Binding Affinity over Large Sequence Spaces**

Adegoke A. Ojewole^{1,3}, Jonathan D. Jou¹, Vance G. Fowler⁴,
and Bruce R. Donald^{1,2}✉

¹ Department of Computer Science, Duke University, Durham, NC, USA
brd+recomb17@cs.duke.edu

² Department of Biochemistry, Duke University Medical Center, Durham, NC, USA

³ Computational Biology and Bioinformatics Program, Duke University,
Durham, NC, USA

⁴ Division of Infectious Diseases, Duke University Medical Center, Durham, NC, USA

Abstract. Protein design algorithms that compute binding affinity search for sequences with an energetically favorable free energy of binding. Recent work shows that the following design principles improve the biological accuracy of protein design: *ensemble-based design* and *continuous conformational flexibility*. Ensemble-based algorithms capture a measure of entropic contributions to binding affinity, K_a . Designs using backbone flexibility and continuous side-chain flexibility better model conformational flexibility. A third design principle, *provable guarantees of accuracy*, ensures that an algorithm computes the best sequences defined by the *input model* (i.e. input structures, energy function, and allowed protein flexibility). However, previous provable methods that model ensembles and continuous flexibility are *single-sequence* algorithms, which are very costly: linear in the number of sequences and thus exponential in the number of mutable residues. To address these computational challenges, we introduce a new protein design algorithm, *BBK**, that retains all aforementioned design principles yet provably and efficiently computes the tightest-binding sequences. A key innovation of *BBK** is the *multi-sequence* (MS) bound: *BBK** efficiently computes a single provable upper bound to approximate K_a for a *combinatorial number of sequences*, and entirely avoids single-sequence computation for all provably suboptimal sequences. Thus, to our knowledge, *BBK** is the first provable, ensemble-based K_a algorithm to run in time *sublinear* in the number of sequences. Computational experiments on 204 protein design problems show that *BBK** finds the tightest binding sequences while approximating K_a for up to 10^5 -fold fewer sequences than exhaustive enumeration. Furthermore, for 51 protein-ligand design problems, *BBK** provably approximates K_a up to 1982-fold faster than the previous state-of-the-art iMinDEE/ A^*/K^* algorithm. Therefore, *BBK** not only accelerates protein designs that are possible with previous provable algorithms, but also efficiently performs designs that are too large for previous methods.

A.A. Ojewole and J.D. Jou contributed equally to this work.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 157–172, 2017.

DOI: 10.1007/978-3-319-56970-3_10

1 Introduction

Protein design is the prediction of protein sequences with desired biochemical functions, which often involve binding to a target ligand. Computational protein design casts functional design into a structural optimization problem whose goal is to find amino acid sequences that fold into specified three-dimensional structures. Protein design algorithms search a space defined by a biophysical *input model*, which defines the sequence and structural search space (i.e. the input structure, allowed amino acid mutations, and allowed protein flexibility); the optimization objective (e.g. design for binding affinity); and the energy function [1]. Protein design algorithms [5,9] have successfully predicted protein sequences that fold and bind desired targets *in vitro* and *in vivo*. For example, these algorithms have been used, with experimental validation, to predict drug resistance [7,37,41] and to design enzymes [3,13,32,49], new drugs [19], inhibitors of protein-protein interactions [15,42], epitope-specific antibody probes [12], and even neutralizing antibodies [17,45].

Computational methods can potentially search a large number of sequences to predict the proteins that bind most tightly to a target ligand in less time and with fewer resources than *in vitro* methods such as phage display [2,38]. However, four computational challenges have prevented protein design algorithms from realizing this potential. First, for each binding interface, an exponentially large number of conformations in each binding partner's ensemble must be pruned or considered to accurately predict binding affinity [5,15,18,32]. Second, for each sequence, finding the lowest energy conformations that most influence binding affinity is NP-hard [26,39,40,55,56], making algorithms that guarantee optimality expensive for larger designs. Third, mutating a protein sequence induces conformational changes in the protein structure. Since such conformational changes occur over many continuous degrees of freedom, algorithms that model continuous flexibility must search over a large, continuous conformation space. Fourth, the number of protein sequences (i.e. the *sequence space*) grows exponentially with the number of simultaneously mutable residues in the design. Therefore, previous algorithms either focus on accurately modeling smaller designs or attempt larger designs by making simplifications that (a) ignore the ensemble nature of proteins, (b) disregard continuous conformational flexibility, or (c) return heuristic solutions with no guarantees. A discussion of these simplifications and their ramifications for protein design follows; see also Supplementary Information [36] (SI) Sect. 2.

Global Minimum Energy Conformation (GMEC)-based algorithms [4,10,20,43,53] assume that the lowest energy conformation accurately predicts binding affinity. However, GMEC-based designs cannot accurately model entropic change due to binding [6] and can disproportionately favor sequences with energetically favorable GMECs over sequences with tight binding affinity [3,15,32,42,46,47]. Many protein design algorithms [29,48,50,51,53] rely on a simplified, discrete model of side-chain flexibility. However, discrete rotamers model a small subset of protein energetics, which are sensitive to small atomic movements not permitted by the discrete model. To overcome this limitation, researchers have developed

provable algorithms [10, 15, 21, 22, 42, 43] that incorporate continuous side-chain flexibility [10, 15]. Another important aspect of design algorithms is the quality of the computed results. Whereas GMEC-based design is NP-hard [26, 40, 55, 56], computation of thermodynamic ensembles and associated partition functions is #P-hard [35, 52, 53]. Provable protein design algorithms either return the optimal sequences or conformations [4, 10, 23, 28, 43, 48, 50, 51, 53] with respect to the input model, or return provably good approximate solutions [15, 21, 22, 32, 42]. Non-provable algorithms such as Metropolis Monte Carlo methods [27, 29, 30] instead use stochastic methods to rapidly sample the space described by the input model. These algorithms return solutions without any guarantees. Thus, ensemble-based design, a realistic model of structural flexibility, and provable optimality of the computed sequences with respect to the input model improve the predictive power of protein design algorithms. However, each of these design principles also increases the cost of protein design (as a function of the number of sequences).

Single-sequence algorithms, which explicitly evaluate each possible sequence, are powerful and versatile. Molecular dynamics [31, 57], for instance, is frequently applied to design for binding affinity. The approach models ensembles and continuous flexibility. Provable algorithms have also been developed to model these two phenomena. The K^* algorithm [15, 32, 42] in OSPREY [10, 11, 13–15, 20–23, 25, 32, 42, 43] uses a combination of dead-end elimination pruning [10, 15] and A^* [24, 28, 44] gap-free conformation enumeration to provably and efficiently approximate K_a . K^* and all previous provable ensemble-based algorithms that model continuous side-chain flexibility [21, 22] are single-sequence algorithms. The empirical and asymptotic runtime complexity of single-sequence algorithms is linear in the number of possible sequences, and therefore exponential in the number of mutable residues. Thus, designs with many mutable residues rapidly become intractable when using single-sequence algorithms. The COMETS algorithm [20] in OSPREY is the only provable multi-state design algorithm that is more efficient than single-sequence algorithms. However, its binding predictions are GMEC-based rather than ensemble-based. Additional background and references are located in **SI** [36], Sect. 1.

To efficiently search large sequence spaces while retaining all the benefits of provable guarantees, ensemble-based design, and continuous side-chain flexibility, we present a new, provable algorithm: Branch and Bound over K^* (BBK^*). The key innovation of BBK^* is the *multi-sequence* (MS) bound: BBK^* efficiently computes upper and lower bounds on the binding affinities of *partial sequences*, which are shared by a combinatorial number of full sequences. By avoiding costly single-sequence computation, BBK^* runs in time *sublinear* in the number of sequences. To our knowledge, BBK^* is the first provable, ensemble-based algorithm to do so. BBK^* not only avoids explicitly computing all possible sequences, but also provably and efficiently enumerates sequences in a gap-free decreasing order of binding affinity. Therefore, BBK^* provides a vast performance improvement over the previous state-of-the-art, by not only accelerating protein designs that were possible with previous provable algorithms, but also

efficiently handling large designs that previous algorithms could not compute in a reasonable amount of time.

By presenting BBK^* , our paper makes the following contributions:

1. A novel, ensemble-based algorithm that provably computes the same results as the previous state of the art (exhaustive search over sequences) but is empirically combinatorially faster, returns a gap-free list of sequences in decreasing order of binding affinity, and runs in time sublinear in the number of sequences.
2. Proofs of correctness for multi-sequence bounds, a key innovation in BBK^* .
3. A new two-pass bound that more efficiently computes a provable ε -approximation to the desired partition functions.
4. 255 protein designs showing that BBK^* approximates binding affinity for full sequences up to 1982-fold faster than the best previous algorithm and that BBK^* computes the best binding sequences in a large sequence space up to 10^5 -fold more efficiently than exhaustive search.
5. Support for both continuous side-chain and backbone flexibility, demonstrating the ability of BBK^* to handle multiple modes of protein flexibility in addition to large conformation and sequence spaces.
6. An implementation of BBK^* in our laboratory’s open-source OSPREY [10, 11, 13–15, 20–23, 25, 32, 42, 43] protein design software package, available for download as free software.

2 Computing the Partition Function

To successfully design for improved binding affinity K_a , design algorithms must consider the energy of more than just the GMEC. In particular, all algorithms that design for improved K_a optimize the ratio of partition function Z for the bound and unbound states of the protein and ligand (Eq. 2). Protein design can thus be cast as an optimization problem [5]. For an n -residue protein design problem with at most a amino acids per mutable residue, let P , L , and PL denote the unbound protein, unbound ligand, and bound protein-ligand complex, respectively. For each sequence s , let $\mathbf{Q}(s)$ be the set of discrete rigid rotamer conformations defined by the allowed amino acids for each mutable residue of s . For a rigid rotamer conformation c , let E_x be a pairwise energy function with respect to input structure X , which may be one of P , L , or PL . In particular, we will consider the case of design with *continuous rotamers* [10, 15]. We define $E_x(c)$ to be the energy of c for structure X after energy-minimizing the side-chains of mutable residues.

2.1 K^*

We define the Boltzmann-weighted partition function $Z_x(s)$ as:

$$Z_x(s) = \sum_{c \in \mathbf{Q}(s)} \exp(-E_x(c)/RT). \quad (1)$$

We define the K^* score, a partition function ratio that approximates binding affinity K_a , as:

$$K^*(s) = \frac{Z_{PL}(s)}{Z_P(s)Z_L(s)}. \quad (2)$$

As stated in Sect. 1, the K^* approximation and, by extension, the full partition function, are #P-hard to compute [35,52,53]. Therefore, researchers have not only developed heuristic algorithms that rapidly compute loose partition function bounds, but also developed efficient, provable algorithms that compute ε -approximations to the partition function. Probabilistic algorithms bound the partition function either provably [54] or non-provably [8]. An efficient ε -approximation to $Z_X(s)$ is computed in [15,32,42]. However, these methods are designed to compute partition functions for single sequences. For an n -residue design with at most t possible amino acids at each residue and q rotamers per amino acid, provable single-sequence methods must compute or bound the partition functions of all t^n sequences, each with q^n conformations. Thus, previous single-sequence algorithms for protein design for binding affinity take time exponential in n ($\mathcal{O}(t^n)$) when computing the sequence with the best predicted binding affinity.

Therefore, to provably find the best binding sequences, new, efficient provable algorithms are needed to search over an exponentially large sequence space, in which each sequence represents an exponentially large conformation space. *BBK** addresses this need. *BBK** compares *partial sequences* (for which some mutable residues have not been assigned an amino acid identity) without computing the partition functions for all *full sequences* (which assign a specific amino acid to each mutable residue). *BBK** computes bounds on the free energies of partial sequences, and avoids enumerating conformations from sequences with poor binding affinity, by pruning sequences during search. As we will describe in Sect. 3, pruning these sequences circumvents prohibitive computational costs required to compute many single-sequence K^* scores.

3 A* Search over Sequences, with Multi-sequence (MS) Bounds

It may at first seem counter-intuitive to compute the sequence with optimal binding affinity, along with its predicted K^* score, without explicitly computing the K^* scores of all possible sequences. Indeed, all previous ensemble-based provable methods, as well as many heuristic methods, are *single-sequence* methods: they must individually evaluate and compare each sequence to provably return the optimal sequence. In contrast, *BBK** bounds the K^* ratios of a combinatorial number of sequences efficiently and can prune these sequences without computing any single-sequence bounds. The key to this improvement is the observation that a partial sequence s' with poor predicted binding affinity adversely affects the K^* score of the combinatorial set of sequences that contain s' . That is, the best possible K^* score consistent with s' limits the K^* score of all sequences

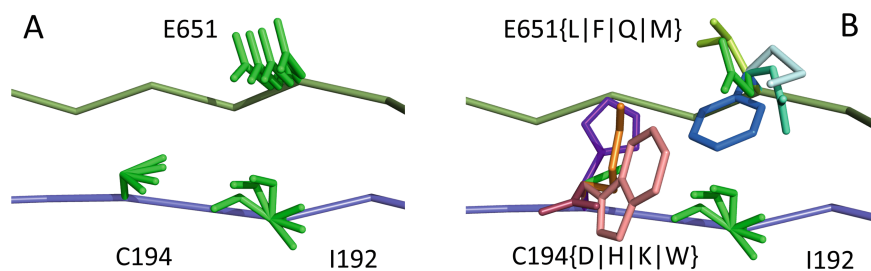


Fig. 1. A toy protein design problem in which conformational ensembles (A) and optimal mutations (B) must be computed at 3 residues. Residues of the fibronectin F1 module (Fn, blue ribbon), and of a fragment of *S. aureus* fibronectin binding protein A (FNBPA-5, green ribbon) are shown (PDB id: 2RL0). Side-chain conformations, labeled with amino acid identity, are also shown per residue. (A) Previous provable methods require a *fully defined sequence* to compute a *single-sequence* (SS) ε -approximation bound on binding affinity (i.e. a K^* score, Eq. 2). (B) **A key innovation in this paper is the *multi-sequence* (MS) bound for binding affinity in protein design.** An MS bound is a provable bound on the binding affinity of a *partial sequence*. Unassigned residues, whose amino acid identities are not defined by the partial sequence, adopt side-chain conformations from *multiple* amino acids, shown as the blue, purple, pink, and light blue ensemble. Thus, an MS bound is a provable upper bound on the binding affinity of all sequences containing that partial sequence, and is obtained without computing any SS bounds. The Fn:FNBPA-5 design problem is described in Sect. 4.3.

consistent with s' . Henceforth, we will refer to a bound on the binding affinity for a sequence as a *bound on the sequence*. To compute a bound on all sequences consistent with s' , BBK^* computes the partition function for an ensemble that contains conformations from multiple sequences. Figure 1 illustrates the difference between single-sequence and multi-sequence ensembles. The K^* ratio of a multi-sequence ensemble is a provable upper bound on the best possible K^* ratio of all sequences that contain s' . This *multi-sequence* bound (MS bound) is not only cheaper to compute, but it also allows BBK^* to compare a combinatorial number of sequences without computing any single-sequence bounds. By bounding every possible sequence consistent with a partial sequence, BBK^* can provably eliminate those sequences, and prune a combinatorial number of sequences without performing any single-sequence computation. Figure 2 illustrates the combinatorial speedup provided by MS bound pruning, whereby pruning the partial sequence obviates computation of all single sequences containing the partial sequence. Details of the algorithm, proofs of its space and time complexity, and comparison to $iMinDEE/A^*/K^*$ are provided in Appendix A of the SI [36]. An additional enhancement, pruning by fold stability compared to wild type, is described in Appendix A.7 of the SI [36].

The improvement of BBK^* over single-sequence methods can be measured using *cost per sequence*. We show the improvement is threefold: BBK^* (a) reduces the cost to compute a bound on a combinatorial number of sequences,

(b) eliminates all computational costs once a sequence is pruned, and (c) when it must compute a bound for a single sequence, computes a bound that is in many cases cheaper than the bounds computed by previous single-sequence algorithms. To guarantee that the first sequence returned is optimal, an algorithm must either compute or bound the partition function for all possible sequences. Previous provable algorithms compute a provable *single-sequence bound* of the partition function, called an ε -approximation (SS- ε bound), for each sequence [15, 32, 42]. These SS- ε bounds are guaranteed to be within a user-specified ε of the K^* score for a sequence. BBK^* also provably returns the optimal sequences, but does so without enumerating all possible sequences. Instead of SS- ε bounds, BBK^* computes an MS bound, which is an upper bound on the best possible K^* score of multiple sequences that share a common partial sequence.

We will now compare the cost of bounding sequences with single-sequence algorithms to the cost with BBK^* . Consider an n -residue protein design: we are given an initial partial sequence s' , which fixes amino acid identity (but not the rotamer) for a residues, and u residues do not have a fixed amino acid identity ($a+u = n$). If the design problem allows at most t amino acids per unassigned (u) residue and at most q rotamers for any amino acid, there are t^u sequences containing s' , and q^a partial conformations defined by s' . A complete sequence would still have q^n conformations, and computing the energy of a conformation takes $\mathcal{O}(n^2)$ time using a pairwise energy function. Thus, a single-sequence algorithm would spend $\mathcal{O}(t^u q^n n^2)$ worst-case time individually computing the K^* scores of all t^u sequences. In contrast, the cost of an MS bound is $\mathcal{O}(q^a(a^2 + q^2 t^2 un))$, which includes $\mathcal{O}(q^a a^2)$ time to compute the pairwise energy of the a assigned residues of all q^a partial conformations, and $\mathcal{O}(q^{a+2} t^2 un)$ time to compute a bound on the energy of each partial conformation. By reducing two exponentials from t^u to t^2 , and from q^n to q^{a+2} , BBK^* computes an MS bound in time sub-linear in the number (t^u) of sequences. The cost to compute a single, provable MS bound (that holds for all t^u sequences) is therefore significantly smaller than the cost to compute t^u single-sequence bounds. Furthermore, these MS bounds are used to prune *partial sequences* containing *combinations of mutations*: for a pruned partial sequence s' , all t^u sequences containing s' are provably eliminated from search without any additional computation. That is, BBK^* provably, *combinatorially* prunes the search space. Finally, MS bounds are in many cases inexpensive to compute when compared to the $\mathcal{O}(q^n n^2)$ complexity of computing an SS- ε bound for a single-sequence. Since there are q^a partial conformation energy bounds to compute, the cost of an MS bound increases exponentially as a increases. Obviously, when $a \ll n$, $q^{a+2} \ll q^n$. This is very advantageous for A^* search, because a is initially very small: when BBK^* begins search, $a = 1$, and increases one at a time. Furthermore, $a + u = n$, and a never exceeds n . Thus in many cases $a \ll n$, and MS bound costs of $\mathcal{O}(q^{a+2} t^2 un)$ are significantly smaller than the SS- ε costs of $\mathcal{O}(q^n n^2)$ for a single sequence. Use of MS bounds enables BBK^* to efficiently bound and prune sequences that would otherwise require $\mathcal{O}(q^n n^2)$ time *each* to evaluate.

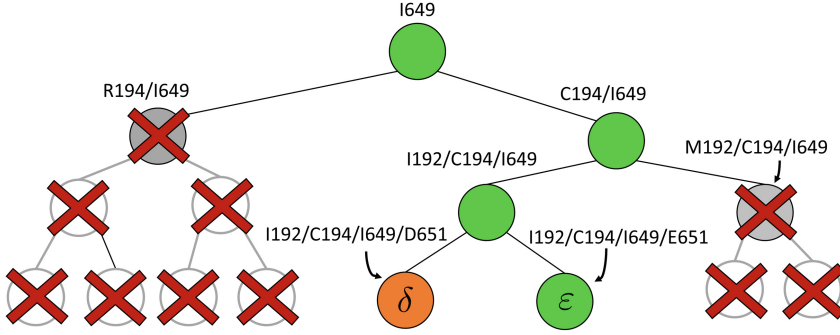


Fig. 2. *BBK pruning efficiently explores the sequence space.** An example design of residues 192 and 194 of the fourth fibronectin F1 module, and residues 649 and 651 of a fragment *S. aureus* fibronectin binding protein A 5 is shown (Fig. 1, PDB Id: 2RL0). As *BBK** searches the sequence space (tree above) its multi-sequence bounds provably prune sub-trees from the sequence space. All sequences containing R194/I649 are pruned (red crosses) after computing exactly one multi-sequence bound: the bound on the partial sequence R194/I649, which is an upper bound for all sequences containing R194/I649. Sequences containing M192/C194/I649 are pruned (red crosses) after computing only the multi-sequence bound for the partial sequence M192/C194/I649. All pruned sequences and partial sequences, shown as empty gray circles, have no additional computation performed. Even though single-sequence bounds are *initiated* for both I192/C194/I649/E651 and I192/C194/I649/D651, the latter is pruned early, after computing a mere δ -approximation bound (orange leaf node), which is cheaper, and not as tight as an ϵ -approximate bound. A provable ϵ -approximation bound (green leaf node) is computed for only the optimal sequence, I192/C194/I649/E651. In contrast, single-sequence methods compute separate ϵ -approximate bounds (which are expensive) for all 8 possible sequences, shown as leaf nodes in the tree.

The algorithmic advances that make MS bounds possible are new bounds on partial and full sequences. We denote the design states unbound protein, unbound ligand, and bound complex as P , L , and PL respectively. The following definitions of these new bounds are sufficient for the theorems provided in the main paper – the precise definitions involve some subtleties, which are deferred to Appendix A of the **SI** [36]. Given a sequence s and a state $X \in \{P, L, PL\}$, the function $L_X(s)$ is a provable lower bound of the partition function for s in state X , and $U_X(s)$ is a provable upper bound on the partition function for s in state X . For a partial sequence s' , $L_X(s')$ and $U_X(s')$ are, respectively, partition function lower and upper bounds for the combinatorial number of sequences containing s' . These lower- and upper-bounding functions are combined into an upper-bounding function $K_a^+(s')$ on the partition function ratio of s' .

Definition 1. Let s be a sequence. $K_a^+(s)$ is defined as follows:

$$K_a^+(s) = \frac{U_{PL}(s)}{L_P(s)L_L(s)}. \quad (3)$$

The following theorem establishes the relationship between the partition function ratio of a partial sequence and the partition function ratio of any sequence containing the partial sequence:

Theorem 1. *Let s be a partial or full sequence. For any partial sequence $s' \subset s$, $K_a^+(s')$ bounds $K_a^+(s)$ from above:*

$$K_a^+(s') \geq K_a^+(s) \geq \frac{Z_{PL}(s)}{Z_P(s)Z_L(s)} = K^*(s). \quad (4)$$

A proof is provided in Appendix A.3 of the SI [36]. Theorem 1 shows that the bounds used by BBK^* are *admissible*. That is, they never underestimate the K^* ratio of any partial sequence. Thus, BBK^* uses $K_a^+(s')$ as the optimistic bounding function for A^* search. Previously, A^* search has been used to provably enumerate conformations within some energy window E_w of the GMEC [28] and to provably approximate the partition function of single sequences [5, 15, 32, 42]. Since Eq. (4) defines an admissible bound over sequences, all of the provable guarantees of A^* apply to BBK^* . With these guarantees, BBK^* provably searches over *sequences* rather than conformations, and is guaranteed to return a gap-free list of sequences in order of decreasing binding affinity.

3.1 Algorithm Overview

BBK^* bounds all possible sequences either with the MS bounds described in Sect. 3, or by computing a single-sequence bound as described in [13, 32, 42]. In brief, to bound a single sequence, BBK^* computes a gap-free list of conformations whose statistical mechanical energies (Eq. 1) are used to bound the K^* ratio. The algorithm reports an error bound δ such that the computed bound is guaranteed to be no more than a $(1 + \delta)$ factor greater than the true K^* ratio. We will refer to these single-sequence, δ -approximate bounds [15, 16] as *SS- δ bounds*. As the gap-free list of conformations used for an SS- δ bound grows in size, the computed single-sequence bound becomes tighter (δ decreases). Eventually, $\delta \leq \varepsilon$, and the single-sequence bound becomes a provable ε -approximation, which we will refer to as an *SS- ε bound*. We will refer to an SS- δ bound constructed this way as a *running bound*, which BBK^* incrementally tightens as it enumerates additional conformations [15].

BBK^* maintains a max heap whose node values correspond to either full or partial sequences and whose node keys are an upper bound on all K^* scores (Eq. 2) in the sequence space represented by the node. The heap is initialized with a node representing the entire sequence space. BBK^* then repeatedly removes the max node x of the queue, and performs one of the following operations:

1. *Branch.* If x contains a partial sequence s' , then s' is expanded. Expansion creates t new child nodes by selecting an unassigned residue r in s' , and creating a new child node for each allowed amino acid a at r . Each child node contains s' plus the additional mutation of a assigned at r . These nodes are bounded with MS bounds or SS- δ bounds, and reinserted into the heap.

2. *Update*. If x contains a complete sequence s , whose bound is an SS- δ bound, then BBK^* enumerates m additional conformations ($m = 8$ in our study), tightening the SS- δ bound. x is reinserted into the heap, with the updated SS- δ bound as its key. Computing this tighter SS- δ bound is important for pruning sequences, as shown by our computational experiments in Sect. 4.2.
3. *Return*. If node x contains a full sequence, whose bound is an SS- ε bound, then the sequence in x has the best K^* score of all unenumerated sequences (as with A^* , all better sequences are guaranteed to have already been enumerated). The sequence of x is returned.

BBK^* terminates when it has enumerated the top k sequences (by SS- ε bound), where k is a user-specified number. A detailed description of the algorithm is provided in Appendix A.8 of the SI [36].

4 Computational Experiments

We implemented BBK^* in our laboratory’s open source OSPREY [11] protein design package and compared our algorithm to the previous state-of-the-art single-sequence iMinDEE/ A^*/K^* algorithm [15,32,42]. We computed the five best binding sequences using both BBK^* and iMinDEE/ A^*/K^* for 204 different protein design problems from 51 different protein-ligand complexes. For each protein-ligand interface, we created four design problems spanning the wild-type sequence and all sets of single, double, triple, and quadruple mutants, respectively. In each design problem, we modeled either 8 or 9 residues at the protein-ligand interface as mutable and flexible. Each mutable residue was allowed to assume its wild-type identity or mutate to 13–19 other amino acids. The size of the resulting design problems ranged from 10 to 2.6×10^6 sequences and 10^5 to 10^{11} conformations (over all sequences). In all cases, we modeled continuous side-chain flexibility using continuous rotamers [10,43]. As in [10,15], rotamers from the Penultimate Rotamer Library [33] were allowed to minimize to any conformation within 9° of their modal χ -angles. For all design problems, we performed minimized dead-end elimination pruning (minDEE) [15], followed by either iMinDEE/ A^*/K^* or BBK^* . The initial pruning window [10] was 0.1 kcal/mol, and the SS- ε bound accuracy was 0.683 (details are provided in Appendix A.9 of the SI [36]). Each design either provably returned the optimal sequences or was terminated after 30 days. A detailed description of the 51 protein-ligand systems in our experiments, the 204 protein design problems based on these systems, and our experimental protocol is provided in Appendix C.1 of the SI [36].

4.1 Performance Comparison

As the size of the sequence space increases, so does the efficiency of BBK^* over iMinDEE/ A^*/K^* (Fig. 3), demonstrating that the complexity of BBK^* is in practice sublinear in the number of sequences. We first measured the efficiency of BBK^* using the number of SS- ε bounds computed. Next, we measured efficiency

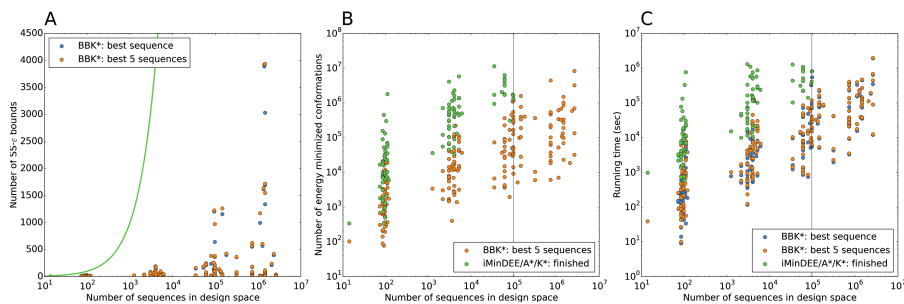


Fig. 3. BBK^* is up to five orders of magnitude more efficient than $iMinDEE/A^*/K^*$. BBK^* completed all 204 designs within a 30 day limit, while $iMinDEE/A^*/K^*$ completed only 107. (A) The number of SS- ϵ bounds performed vs. the number of sequences in the design space. Results are shown for computing only the the best sequence (blue) and computing the best five sequences (orange). Single-sequence algorithms, including the best previous algorithm $iMinDEE/A^*/K^*$, must compute binding affinity for all possible sequences (green curve). BBK^* required up to 6×10^5 -fold fewer SS- ϵ bounds to find the best sequences. (B) The number of energy-minimized conformations by BBK^* and $iMinDEE/A^*/K^*$ vs. the number of sequences in the design space. $iMinDEE/A^*/K^*$ completed only 107 of 204 designs (left of the vertical line) before the 30-day limit. For these designs, BBK^* was up to 1700-fold more efficient. (C) BBK^* and $iMinDEE/A^*/K^*$ running times vs. the number of sequences in the design space. For the 107 designs completed by $iMinDEE/A^*/K^*$ within 30 days (left of the vertical line), BBK^* was up to 800-fold more efficient than $iMinDEE/A^*/K^*$.

using the number of conformation energy minimizations performed. Last, we compared the running times of BBK^* to those of $iMinDEE/A^*/K^*$.

We divide the design problem sizes into three categories: the smallest problems have between 10 and 10^2 sequences; medium-sized problems contain between 10^2 and 10^4 sequences; and the largest problems contain between 10^4 and 10^7 sequences. After 30 days, BBK^* completed all 204 designs, but $iMinDEE/A^*/K^*$ completed only 107 of 204 designs: all 39 of the smallest designs, 54 of 63 medium-sized designs, and only 14 of the 111 largest designs. We now discuss results for the 107 designs completed by $iMinDEE/A^*/K^*$. Because $iMinDEE/A^*/K^*$ computes individual sequence binding energies as SS- ϵ bounds, we first measured the efficiency of BBK^* using the number of SS- ϵ bounds computed (Fig. 3(A)). For small, medium, and large designs, respectively, BBK^* was on average 17-fold, 162-fold, and 2568-fold more efficient than $iMinDEE/A^*/K^*$. Next, we measured efficiency using the number of conformation energy minimizations performed (Fig. 3(B)). Here, BBK^* minimized, on average, 10-fold, 43-fold, and 113-fold fewer conformations for small, medium, and large-sized designs, respectively, compared to $iMinDEE/A^*/K^*$. Last, we compared empirical running times for both methods (Fig. 3(C)). On average, BBK^* was 36-fold, 67-fold, and 97-fold faster than $iMinDEE/A^*/K^*$ for small, medium, and large-sized designs, respectively.

Based on the 107 designs that iMinDEE/ A^*/K^* was able to complete within 30 days, we conclude that BBK^* provides a combinatorial speedup over iMinDEE/ A^*/K^* . Crucially, BBK^* is not only more efficient, but also retains the provability guarantees and biophysical modeling improvements (viz. ensemble-based design, continuous flexibility) employed by *single-sequence* iMinDEE/ A^*/K^* . In one large design (2.6×10^6 sequences), involving a camelid single-domain V_{HH} antibody fragment in complex with RNase A (PDB id: 2P49), BBK^* pruned more than 99.9% of sequences to provably find the best 5 sequences. Therefore, BBK^* can design over similarly sized sequence spaces to high throughput experimental screening methods such as phage display [2, 38].

4.2 Sequence Space Pruning

BBK^* owes its efficiency to two complementary modes of sequence pruning: MS bound pruning and SS- δ bound pruning. Figure 4(A) illustrates the efficiency gains in BBK^* due to MS bound pruning. In small, medium, and large design problems, respectively, BBK^* pruned up to 90%, 99% and 99.9% of the sequence design space using MS bound pruning. These data show that the amount of MS pruning increased significantly with the size of the design space. Figure 4(B) illustrates the efficiency gains in BBK^* due to SS- δ bound pruning. In small, medium, and large design problems, respectively, SS- δ pruning eliminates up to 98%, 99.9% and 99.99% of the sequences not pruned by MS pruning. These data show that the amount of SS- δ pruning increased with the size of the design problem. Further details are provided in Appendices A.10 and B.1 of the SI [36].

Importantly, MS bound pruning and SS- δ bound pruning have multiplicative synergy, producing a combined pruning effect of up to 99.99999% of the original sequence space while provably finding the five best-binding sequences. In one example, we re-designed the protein-protein interface of a camelid affinity-matured single-domain V_{HH} antibody fragment (PDB id: 2P4A). The sequence space, 2.6×10^6 sequences, consisted of all quadruple mutants in the 9-residue protein-protein interface. BBK^* pruned all but 2078 sequences using MS pruning and then pruned 2071 sequences from these remaining 2078 sequences using SS- δ bound pruning. These data show how BBK^* prunes a combinatorial number of sequences from the design space, producing dramatic efficiency gains over single-sequence methods. See SI [36] Sect. 5.2 for details.

4.3 Design with Coupled Continuous Side-Chain and Backbone Flexibility

To determine whether design with a *fixed* backbone and continuous rotamers predicts tight binding in the same sequences as does a model with both *local backbone flexibility* and continuous rotamers, we used BBK^* to redesign the Human Fibronectin F1:*Staphylococcus aureus* FNBPA-5 interface [34] (PDB id: 2RL0) for binding affinity. As we will discuss below, the flexible backbone model favors binding in different sequences than the fixed backbone model does. Details of our experimental protocol are provided in Appendix C.3 of the SI [36].

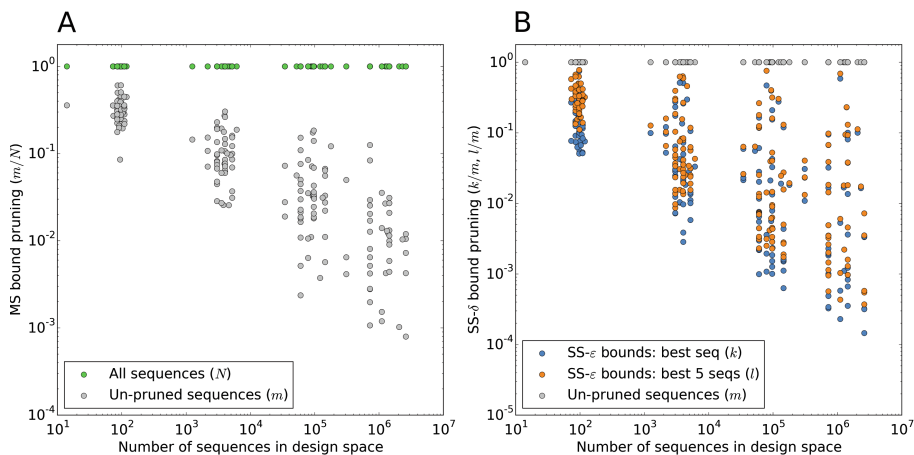


Fig. 4. *BBK used MS and SS- δ bounds to prune up to 99.99999% of the sequence space.** (A) Sequence space reduction due to MS pruning. The fraction of un-pruned sequences (gray) normalized to the total number of sequences (green). *BBK** used MS bounds to provably prune up to 99.9% of the sequence space. *BBK** does not compute SS- ϵ bounds for pruned sequences. (B) The fraction of *BBK** SS- ϵ bounds (blue for the best sequence and orange for the best 5 sequences) normalized to the number of sequences not pruned by MS bound pruning (gray). To compute the best sequences, *BBK** calculated SS- ϵ bounds for as few as 0.01% of the un-pruned sequences. The remaining 99.99% of these sequences were pruned at mere SS- δ accuracy.

In the first experiment, we re-designed the Fibronectin F1:FNBPA-5 interface for binding affinity over the wild-type sequence and 15 single amino-acid polymorphisms. Our results showed that using the flexible backbone model versus the fixed backbone model increased the size of the design conformation space by 1417-fold but only increased the running time by 4-fold in *BBK**. By comparison, iMinDEE/*A**/*K** required 48-fold more time than *BBK** to complete the flexible backbone design. Our results also showed that the *BBK** sequence rankings between the two input models had a Spearman correlation coefficient of only $\rho = 0.53$. Thus, the flexible backbone model favors binding in different sequences than the fixed backbone model does. For instance, the FNBPA-5 D650E mutant is predicted to bind less tightly than the wild-type in the fixed backbone model (Fig. 5(A)) but more tightly than WT in the flexible model (Fig. 5(B)). In our second experiment, the sequence design space consisted of the wild-type sequence and 25 single amino-acid polymorphisms. The *BBK** sequence rankings produced by the two input models had a Spearman correlation coefficient of $\rho = 0.82$ (additional details are provided in Section B.2 of the **SI** [36]). Relative to the fixed backbone model, the flexible backbone model increased the size of the design conformation space by 8447-fold but only increased the running time by only 1.7-fold in *BBK**. iMinDEE/*A**/*K** required

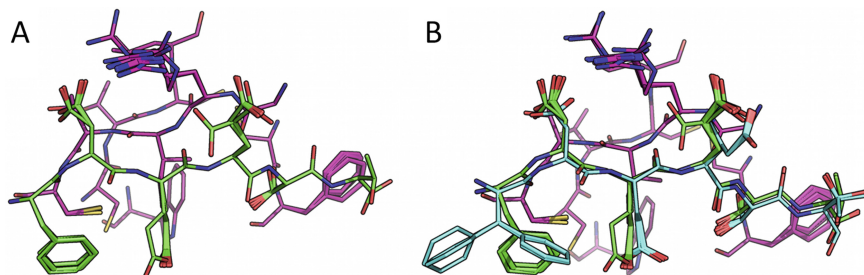


Fig. 5. BBK^* efficiently handles coupled continuous side-chain and local backbone flexibility. Selected residues from ensembles, computed by BBK^* , of human fibronectin F1 modules 4–5 (magenta) in complex with a fragment of *S. aureus* fibronectin binding protein A 5 (FNBPA-5, PDB id: 2RL0, [34]). The design space consisted of the wild-type sequence and either 15 or 25 single amino-acid mutants. (A) Ensemble of the wild-type sequence based on the original crystal structure. The design used a fixed FNBPA-5 backbone (green) and continuous side-chain flexibility. (B) Ensemble of the wild-type sequence using two backbones: the original FNBPA-5 backbone (green) and a second backbone (PDB id: 2RKY, cyan) with RMSD 1.3 Å from the original (found using the MASTER program [58]). The sequence rankings (by K^* score, Eq. 2) from the fixed and flexible backbone models had Spearman correlation coefficients of $\rho=0.53$ and $\rho=0.82$ in the 15 and 25 mutant designs, respectively. This shows that the flexible backbone model favors binding in very different sequences than the fixed backbone model does.

89-fold more time than BBK^* to complete the design using the flexible backbone model.

It is important to note that these experiments are only possible with provable algorithms. Without the provable guarantees of BBK^* , it would be difficult and perhaps unsound to compare the results of computational protein design with and without coupled continuous side-chain and backbone flexibility, since difference induced by the fixed backbone and rotamer model cannot be deconvolved from differences stemming from undersampling or inadequate stochastic optimization. Thus, BBK^* provides provable methods to analyze the difference in predicted sequences between different models of side-chain and backbone flexibility.

5 Conclusion

BBK^* fills an important *lacuna* in protein design: we presented a novel algorithm that can search not over the energies of single-conformations, but instead over the binding affinity of sequences. BBK^* is, to our knowledge, the first provable, ensemble-based algorithm to search over binding affinity and run in time *sublinear* in the number of sequences. Previously, protein designers either employed heuristic algorithms to compute locally optimal sequences, or computed provably accurate approximations of binding affinity for each sequence

individually. *BBK** not only computes the globally optimal sequences, it does so while combinatorially pruning the sequence space. Our experiments show that *BBK** can search over sequence spaces of up to 2.6×10^6 sequences, a capacity comparable to high-throughput experimental screening methods such as phage display. Thus, *BBK** liberates binding affinity-based protein design from the efficiency barrier imposed by exhaustive search. Ensemble-based design for affinity over large sequence spaces was previously possible only with heuristic algorithms (with no guarantees), or using high-throughput wet-bench experiments. *BBK** enables computational protein design by providing new K_a algorithms, with provable guarantees, for these large-scale protein designs.

Acknowledgments. We thank Drs. Mark Hallen and Pablo Gainza for helpful discussions and for providing useful protein-ligand binding problems; Dr. Jeffrey Martin for software optimizations; Hunter Nisonoff, Anna Lowegard and all members of the Donald lab for helpful discussions; and the NSF (GRFP DGF 1106401 to AAO) and the NIH (R01-GM78031 to BRD, R01-HL119648 to VGF) for funding.

References

1. Boas, F.E., Harbury, P.B.: *Curr. Opin. Struct. Biol.* **17**, 199 (2007)
2. Carmen, S., Jermutus, L.: *Brief Funct. Genomic Proteomic* **1**, 189 (2002)
3. Chen, C.-Y., et al.: *Proc. Natl. Acad. Sci. USA* **106**, 3764 (2009)
4. Desmet, J., et al.: *Nature* **356**, 539 (1992)
5. Donald, B.R.: *Algorithms in Structural Molecular Biology*. MIT Press, Cambridge (2011)
6. Fleishman, S.J., et al.: *Protein Sci.* **20**, 753 (2011)
7. Frey, K.M., et al.: *Proc. Natl. Acad. Sci. USA* **107**, 13707 (2010)
8. Fromer, M., Yanover, C.: *Bioinformatics* **24**, i214 (2008)
9. Gainza, P., Nisonoff, H.M., Donald, B.R.: *Curr. Opin. Struct. Biol.* **39**, 16 (2016)
10. Gainza, P., Roberts, K.E., Donald, B.R.: *PLoS Comput. Biol.* **8**, e1002335 (2012)
11. Gainza, P., et al.: *Methods Enzymol* **523**, 87 (2013). Program, user manual, and source code are available at www.cs.duke.edu/donaldlab/software.php
12. Georgiev, I., et al.: *Retrovirology* **9**(Suppl. 2), P50 (2012)
13. Georgiev, I., Donald, B.R.: *Bioinformatics* **23**, i185 (2007)
14. Georgiev, I., Lilien, R.H., Donald, B.R.: *Bioinformatics* **22**, e174 (2006)
15. Georgiev, I., Lilien, R.H., Donald, B.R.: *J. Comput. Chem.* **29**, 1527 (2008)
16. Georgiev, I.S.: *Novel algorithms for computational protein design, with applications to enzyme redesign and small-molecule inhibitor design*. Ph.D. thesis, Duke University (2009). <http://hdl.handle.net/10161/1113>
17. Georgiev, I.S., et al.: *J. Immunol.* **192**, 1100 (2014)
18. Gilson, M.K., et al.: *Biophys. J.* **72**, 1047 (1997)
19. Gorczynski, M.J., et al.: *Chem. Biol.* **14**, 1186 (2007)
20. Hallen, M.A., Donald, B.R.: *J. Comput. Biol.* **23**, 311 (2016)
21. Hallen, M.A., Gainza, P., Donald, B.R.: *J. Chem. Theory. Comput.* **11**, 2292 (2015)
22. Hallen, M.A., Jou, J.D., Donald, B.R.: *J. Comput. Biol.* Epub ahead of print (2016)
23. Hallen, M.A., Keedy, D.A., Donald, B.R.: *Proteins* **81**, 18 (2013)
24. Hart, P., Nilsson, N., Raphael, B.: *IEEE Trans. SSC* **4**, 100 (1968)
25. Jou, J.D., et al.: *J. Comput. Biol.* **23**, 413 (2016)

26. Kingsford, C.L., Chazelle, B., Singh, M.: *Bioinformatics* **21**, 1028 (2005)
27. Kuhlman, B., Baker, D.: *Proc. Natl. Acad. Sci. USA* **97**, 10383 (2000)
28. Leach, A.R., Lemon, A.P.: *Proteins* **33**, 227 (1998)
29. Leaver-Fay, A., et al.: *Methods Enzymol.* **487**, 545 (2011)
30. Lee, C., Levitt, M.: *Nature* **352**, 448 (1991)
31. Leech, J., Prins, J.F., Hermans, J.: *Comput. Sci. Eng.* **3**, 38 (1996)
32. Lilien, R.H., et al.: *J. Comput. Biol.* **12**, 740 (2005)
33. Lovell, S.C., et al.: *Proteins* **40**, 389 (2000)
34. Lower, S.K., et al.: *Proc. Natl. Acad. Sci. USA* **108**, 18372 (2011)
35. Nisonoff, H., Thesis, B.S.: Department of Mathematics, Duke University (2015). <http://hdl.handle.net/10161/9746>
36. Ojewole, A.A., et al.: Supplementary information: BBK* (Branch and Bound over K*): a provable and efficient ensemble-based algorithm to optimize stability and binding affinity over large sequence spaces for sparse approximations of computational protein design (2015). <http://www.cs.duke.edu/donaldlab/Supplementary/recomb17/bbkstar>
37. Ojewole, A., et al.: *Methods Mol. Biol.* **1529**, 291 (2017)
38. Pál, G., et al.: *J. Biol. Chem.* **281**, 22378 (2006)
39. Peng, J., et al.: [q-bio.BM] (2015). [arXiv:1504.05467](https://arxiv.org/abs/1504.05467)
40. Pierce, N.A., Winfree, E.: *Protein Eng* **15**, 779 (2002)
41. Reeve, S.M., et al.: *Proc. Natl. Acad. Sci. USA* **112**, 749 (2015)
42. Roberts, K.E., et al.: *PLoS Comput. Biol.* **8**, e1002477 (2012)
43. Roberts, K.E., Donald, B.R.: *Proteins* **83**, 1151 (2015)
44. Roberts, K.E., et al.: *Proteins* **83**, 1859 (2015)
45. Rudicell, R.S., et al.: *J. Virol.* **88**, 12669 (2014)
46. Sciretti, D., et al.: *Proteins* **74**, 176 (2009)
47. Silver, N.W., et al.: *J. Chem. Theory Comput.* **9**, 5098 (2013)
48. Simoncini, D., et al.: *J. Chem. Theory. Comput.* **11**, 5980 (2015)
49. Stevens, B.W., et al.: *Biochemistry* **45**, 15495 (2006)
50. Traoré, S., et al.: *Bioinformatics* **29**, 2129 (2013)
51. Traoré, S., et al.: *J Comput. Chem.* **37**, 1048 (2016)
52. Valiant, L.G.: *Theoret. Comput. Sci.* **8**, 189 (1979)
53. Viricel, C., et al.: The 22nd International Conference on Principles and Practice of Constraint Programming (2016)
54. Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: CoRR abs/1301.0610 (2013)
55. Xu, J.: 9th Annual International Conference, RECOMB, vol. 3500, p. 423 (2005)
56. Xu, J., Berger, B.: *J. ACM* **53**, 533 (2006)
57. Zheng, F., et al.: *J. Am. Chem. Soc.* **130**, 12148 (2008)
58. Zhou, J., Grigoryan, G.: *Protein Sci* **24**, 508 (2015)

Superbubbles, Ultrabubbles and Cacti

Benedict Paten¹(✉), Adam M. Novak¹, Erik Garrison², and Glenn Hickey¹

¹ UC Santa Cruz Genomics Institute, University of California Santa Cruz,
Santa Cruz, CA 95064, USA
benedict@soe.ucsc.edu

² Wellcome Trust Sanger Institute, Cambridge, UK

Abstract. A superbubble is a type of directed acyclic subgraph with single distinct source and sink vertices. In genome assembly and genetics, the possible paths through a superbubble can be considered to represent the set of possible sequences at a location in a genome. Bidirected and bidedged graphs are a generalization of digraphs that are increasingly being used to more fully represent genome assembly and variation problems. Here we define snarls and ultrabubbles, generalizations of superbubbles for bidirected and bidedged graphs, and give an efficient algorithm for the detection of these more general structures. Key to this algorithm is the cactus graph, which we show encodes the nested decomposition of a graph into snarls and ultrabubbles within its structure. We propose and demonstrate empirically that this decomposition on bidirected and bidedged graphs solves a fundamental problem by defining genetic sites for any collection of genomic variations, including complex structural variations, without need for any single reference genome coordinate system. Furthermore, the nesting of the decomposition gives a natural way to describe and model variations contained within large variations, a case not currently dealt with by existing formats, e.g. VCF.

1 Introduction

Graphs are used extensively in biological sequence analysis, where they are often used to represent uncertainty about, or ensembles of, potential nucleotide sequences. Several subtypes have become especially prominent for sequence representation, in particular the De Bruijn graph [4, 13], the string graph [10], the breakpoint graph [1, 14] and the bidirected graph (aka variation graph or sequence graph) [6, 9].

In the context of de novo sequence assembly several characteristic types of subgraph are recognised, in particular the *bubble* [16], a pair of paths that start and end at common source and sink nodes but are otherwise disjoint. In the context of sequence analysis, a bubble can represent a potential sequencing error or a genetic variation within a set of homologous molecules. An efficient algorithm for bubble detection was proposed by [2].

A generalization of the notion of a bubble, the superbubble is a more complex subgraph type in which a set of (not necessarily disjoint) paths start and end at common source and sink nodes. This problem was initially proposed by [11],

who gave a quadratic solution. [3] recently provided a linear time algorithm for superbubbles on directed acyclic graphs (DAGs). This result, when paired with a previous linear time transformation of the problem of superbubbles on directed graphs to superbubbles on DAGS [15], yields a linear cost solution for computing superbubbles on digraphs. For a review of superbubbles and their use in sequence analysis see [8]. In this paper we generalize the idea of superbubble to the more general case of a bidirected graph, connect a slight generalization of the superbubble, which we call the ultrabubble, and show how it relates to the decomposition of the graph into 2- and 3-edge connected components.

2 Methods

2.1 Directed, Bidirected and Biedged Graphs

A *bidirected graph* $D = (V_D, E_D)$ is a graph in which each endpoint of every edge has an independent orientation (denoted either “left” or “right”), indicating if the endpoint is incident with the left or right *side* of the given vertex. The sides of D are therefore the set $V_D \times \{left, right\}$, and each edge in E_D is a pair set of two sides (Fig. 1). We say for all $x \in V_D$, $(x, left)$ and $(x, right)$ are *opposite sides*.

Any digraph is a special case of a bidirected graph in which each edge connects a left and a right side (by convention we here consider the right side to be the outgoing side and the left side the incoming side, so that the conversion from a digraph to a bidirected graph is determined; see Fig. 1).

A *biedged graph* is a graph with two types of edges: *black edges* and *grey edges*, such that each vertex is incident with at most one black edge (Fig. 1(C)).

For any bidirected graph D there exists an equivalent biedged graph $B(D) = (V_{B(D)}, E_{B(D)})$ where:

- $V_{B(D)} = V_D \times \{left, right\}$, the sides of V_D .
- $E_{B(D)} = S_{B(D)} \cup E_D$, where E_D are the grey edges,
- and $S_{B(D)} = \{(x, left), (x, right)\} | x \in V_D\}$ are the black edges.

For a vertex $x \in V_{B(D)}$ we use the notation \hat{x} to denote the opposite side to x , e.g. for $x = (x', left) \in V_{B(D)}$, $\hat{x} = (x', right)$.

Clearly the bidirected and biedged representations are essentially equivalent, and the choice to use either one is largely a stylistic consideration. For the remainder of this paper we will mostly use the biedged representation. As any digraph is a special case of a bidirected graph and any bidirected graph has an equivalent biedged graph, so any digraph has an equivalent biedged graph.

2.2 Directed Walks on Biedged and Bidirected Graphs

A directed walk on a bidirected graph is a walk that at each visited vertex exits the opposite side to that which it enters. On a biedged graph a directed walk is equivalent to a walk that alternates between black and grey edges. A bidirected or biedged graph is acyclic if it contains no directed cycles.

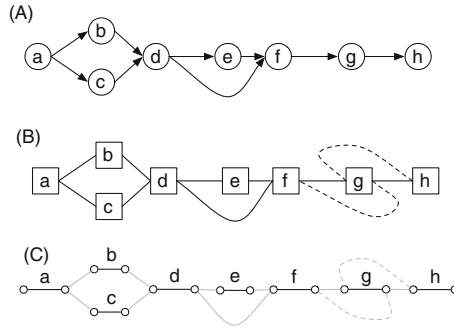


Fig. 1. (A) A digraph. (B) A bidirected graph. Each node is drawn as a box and the orientation for each edge endpoint is indicated by the connection to either the left or right side of the node. The graph excluding the dotted edges is the equivalent bidirected graph for the digraph in (A); the dotted edges encode an inversion that cannot be expressed in the digraph representation. (C) A biedged graph equivalent to the bidirected graph shown in (B).

These definitions are a generalization of a directed walk on a digraph. In a bidirected representation of a digraph all edges in a directed walk are all left-to-right or all right-to-left. A directed walk on a general bidirected (or biedged) graph can mix these two types and additionally include edges that do not alternate the orientation of their endpoints (e.g. left-right and right-right and left-left edges).

Given these generalizing relationships, clearly a digraph \mathbf{D} is acyclic iff $B(\mathbf{D})$ is acyclic. Note that any acyclic biedged graph can also be converted into an equivalent directed acyclic graph (DAG):

Lemma 1. *For any acyclic biedged graph $B(D)$ there exists an isomorphic biedged graph $B(\mathbf{D})$ such that \mathbf{D} is a DAG.*

Proof. Use a depth first search (DFS) beginning at side x to label the sides of $B(D)$ either ‘red’ or ‘white’: If x is not already labelled then label x red and \hat{x} white. For each grey edge incident with \hat{x} , if the connected side is not labeled, label the connected side red and continue recursively via DFS. In this way all the sides in the connected component containing x will be labeled in a single DFS. If during the recursion the connected side encountered is already labelled then it must be labeled red, else there would exist a cycle in the DFS, a contradiction. Use the labelling to create $B(\mathbf{D})$, isomorphic to $B(D)$ but replacing the orientation of the sides so that each side labeled white is a left side and each side labeled red is a right side. All edges in $B(\mathbf{D})$ connect a left and a right side.

2.3 Superbubbles, Snarls and Ultrabubbles

Repeating the definition from [11], any pair of distinct vertices (x, y) in a digraph \mathbf{D} is called a *superbubble* (Fig. 2(A)) if:

- *reachability*: y is reachable from x .
- *matching*: The set of vertices, X , reachable from x without passing through y is equal to the set of vertices from which y is reachable without passing through x (passing through here means to enter and then exit a vertex on the path).
- *acyclicity*: The subgraph induced by X is acyclic.
- *minimality*: No vertex in X other than y forms a pair with x that satisfies the criteria defined above, and similarly for y .

We call the subgraph induced by X the *superbubble subgraph*.

To generalize superbubbles for biedged graphs we introduce the notion of a snarl, a minimal subgraph in a biedged graph whose vertices are at most 2-black-edge-connected (2-BEC) to the remainder of the graph (two vertices in a biedged graph are k -black-edge-connected (k -BEC) if it takes the deletion of at least k black edges to disconnect them). In a biedged graph $B(D)$ a pair set of distinct, non-opposite vertices $\{x, y\}$ are a *snarl* (Fig. 2(B)) if:

- *seperable*: The removal of the black edges incident with x and y disconnects the graph, creating a *separated component* X containing x and y and not \hat{x} and \hat{y} .
- *minimality*: No node z in X exists such that $\{x, z\}$ fulfils the above criteria, and similarly for y .

We call a vertex not incident with a grey edge a *tip* [16]. In a biedged graph $B(D)$ a snarl is an *ultrabubble* if its separated component is acyclic and contains no tips.

The following shows that a superbubble in a digraph is an ultrabubble in the equivalent biedged graph.

Lemma 2. *For any superbubble (x, y) in a digraph D , the pair set $\{x' = (x, right), y' = (y, left)\}$ is an ultrabubble in $B(D)$.*

Proof. Let d and e be the black edges incident with x' and y' , respectively, and let X be the superbubble subgraph of (x, y) .

We start by proving that $\{x', y'\}$ satisfies the separable criteria. As y is reachable from x by definition there exists a directed path in $B(D)$ between x' (the right side of x) and y' (the left side of y) that excludes d and e . After the deletion of these black edges x' and y' therefore remain connected. If the separable criteria is not satisfied the deletion of d and e must therefore not disconnect x' and y' from either or both \hat{x}' and \hat{y}' , without loss of generality assume x' (and therefore y') remains connected to \hat{x}' .

If \hat{x}' is on a directed walk from x' that excludes d then the addition of d to this walk defines a directed cycle in $B(D)$. As all nodes reachable from x are in the separated component X , the existence of this cycle in $B(D)$ implies the existence of a corresponding directed cycle in X , a contradiction.

If there exists a non-directed walk from x' to \hat{x}' then let z' be the last node on the walk from x' such that the subwalk between x' and z' is a directed walk.

By definition, there exists directed walk from z' to y' . The next node on the walk from x' to \hat{x}' after z' is, by definition, not reachable from x' but y' must be reachable from this node. This implies a contradiction of the matching criteria for the corresponding nodes in X .

We have therefore established that $\{x', y'\}$ fulfills the seperable criteria. We have already established that iff a digraph is acyclic its equivalent biedged graph is acyclic, therefore the seperated component of $\{x', y'\}$ is acyclic. As every node in X is both reachable from x and on a path from y , the seperated component clearly contains no tips.

It remains to prove that $\{x', y'\}$ fulfills the minimality criteria. If $\{x', y'\}$ do not satisfy the minimality criteria without loss of generality there exists a node z' in the seperated component of $\{x', y'\}$ such that $\{x', z'\}$ are separable. It follows that all directed paths from x' to y' that exclude d and e visit z' , and for the node z in \mathbf{D} contained in z' , (x, z) fulfills (clearly) all the superbubble criteria, a contradiction.

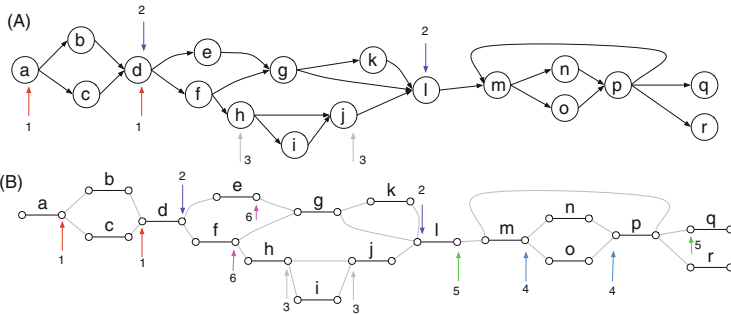


Fig. 2. (A) Superbubbles in a digraph. The superbubbles are indicated by pairs of numbered arrows. (B) A biedged graph representation of the digraph in (A). The ultrabubbles are illustrated, as are two snarls that are not ultrabubbles (of several; pairs 5 and 6, whose seperated components contain cycles).

2.4 Cactus Graphs

A cactus graph is a graph in which any two vertices are at most two-edge connected [7]. In a cactus graph each edge is part of at most one simple cycle, and therefore any two simple cycles intersect at most one vertex.

For a graph $G = (V_G, E_G)$ let $G' = (V_{G'}, E_{G'})$ be a multigraph created by merging subsets of the vertices, such that:

- $V_{G'}$ is a partition of V_G ,
- $E_{G'} = \{\{a_{G'}(x), a_{G'}(y)\} | \{x, y\} \in E_G\}$ is a multiset.

where $a_{G'} : V_G \rightarrow V_{G'}$ is a graph homomorphism that maps each vertex in V_G to the set in $V_{G'}$ that contains it.

Merging all equivalence classes of 3-edge connected (3-EC) vertices in a graph results in a cactus graph [12].

For a bidedged graph $B(D)$ let $C(D)$ be the cactus graph created by first contracting all the grey edges in $B(D)$ then for each equivalence class of 3-EC vertices in the resulting graph merging together the vertices within the equivalence class (Fig. 3(A–C)). As with G' and G , $V_{C(D)}$ is a partition of the vertices of $V_{B(D)}$, and $E_{C(D)} = \{\{a_{C(D)}(x), a_{C(D)}(y)\} | \{x, y\} \in S_{B(D)}\}$ is a multiset.

For a vertex $x \in V_{B(D)}$ we call $a_{C(D)}(x)$ its *projection* (in $C(D)$). Similarly for a set of vertices $X \subset V_{B(D)}$ we call $\{a_{C(D)}(x) | x \in X\}$ the projection of X (in $C(D)$). Let $b_{C(D)}(x) = \{a_{C(D)}(x), a_{C(D)}(\hat{x})\}$, which is the projection of the black edge incident with x in $C(D)$.

Appendix 1 gives lemmas that make explicit the relationship between the edge connectivity of vertices in $B(D)$ and $C(D)$, and which we use to prove the relationship between the snarls of $B(D)$ and $C(D)$.

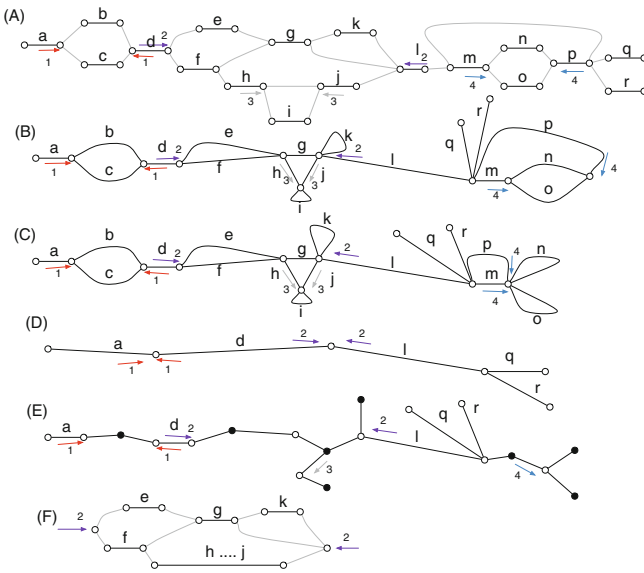


Fig. 3. (A) A bidedged graph $B(D)$ with ultrabubbles indicated by pairs of numbered arrows. (B) The graph in (A) after contracting the grey edges. (C) The cactus graph $C(D)$ for $B(D)$. (D) The bridge forest $D(D)$. (E) The cactus tree $T(D)$. (F) A net for the number 2 bridge pair in (A). The projection of chain pairs in $B(D)$ to the other graphs is shown using the numbered arrows, with the arrows drawn along the projecting black edge incident with the projected vertex.

2.5 Snarls and Cacti

A pair set of distinct vertices $\{x, y\}$ in $B(D)$ are a *chain pair* if they project to the same vertex in $C(D)$ and their incident black edges project to the same simple cycle in $C(D)$ (e.g. grey arrows and cyan arrows in Fig. 3(C)). A cyclic sequence of chain pairs within the same simple cycle in $C(D)$ and ordered according to the ordering of this simple cycle is a (*cyclic*) *chain*. Contiguous chain pairs in a chain share two opposite sides of a black edge in $B(D)$.

For a cactus graph $C(D)$, the graph $D(D)$ resulting from contracting all the edges in simple cycles in $C(D)$ is called a *bridge forest* (Fig. 3(D)).

A pair set of distinct vertices $\{x, y\}$ in $B(D)$ are a *bridge pair* if they project to the same vertex in $D(D)$ and both their incident black edges are bridges (e.g. pairs of arrows numbered 1 and 2 in Fig. 3(D)). A maximum sequence of bridge pairs within $D(D)$ connected by incident nodes with degree two is an (*acyclic*) *chain*. As with chain pairs, contiguous bridge pairs in a chain share two opposite sides of a black (bridge) edge in $B(D)$.

Theorem 1. *The set of snarls in $B(D)$ is equal to the union of chain pairs and bridge pairs.*

Proof. Follows from Lemmas 12 and 13 given in Appendix 2.

Given Theorem 1 to calculate the set of snarls for a given biedged graph it is sufficient to calculate the cactus graph to give the set of snarls that map to chain pairs and the bridge forest to calculate the set of snarls that map to bridge pairs. Constructing a cactus graph of the type described for a biedged graph is linear in the size of the biedged graph (using the algorithm described in [12]), and clearly the cost of then calculating the bridge forest from the cactus graph is similarly linear. The number of chain pairs is clearly linear in the size of the biedged graph, however, the number of bridge pairs is potentially quadratic in the number of bridge pairs, so enumerating these latter snarls has potentially worst case quadratic cost in terms of the size of the biedged graph. Below we consider ways to prune the set of snarls by using their natural nesting relationships to create a hierarchy of snarls that is at most linear in the size of the biedged graph.

2.6 Ultrabubbles and Cactus Trees

Given Theorem 1, to determine the ultrabubbles in $B(D)$ it is sufficient to check for each chain and bridge pair if the separated component is acyclic and contains no tips. As snarls can contain each other, to do this efficiently we decompose the problem into a series of smaller independent problems. We use a modification of the cactus graph called a cactus tree. For a cactus graph $C(D)$ the *cactus tree* $T(D)$ is created by, for each simple cycle S in $C(D)$, making a novel *chain* vertex, adding an edge between each vertex in S and x , and deleting the edges in S (Fig. 3(E)). We call each non-chain vertex (a member of the set $V_{C(D)}$) in $T(D)$ a *net* vertex. For each chain pair $\{x, y\}$ in $B(D)$ the edge in $T(D)$ connecting the net vertex projected to by x and y and the chain vertex representing the simple

cycle in $C(D)$ projected to by the black edges incident with x and y is the chain pair's *chain edge*. Each bridge edge in $B(D)$ projects to the other type of edge in $T(D)$, which connects two net vertices. Each pair of such edges connected by a path of edges connecting chain and net vertices represents a bridge pair. The edges of a cactus tree $T(D)$ are therefore decomposable into a set of edges representing the chain pairs in $B(D)$ and a set of edges representing the bridges in $B(D)$.

A *parent* snarl contains a distinct *child* snarl if the separated component of the child is contained entirely within the separated component of the parent. From the definition it follows that a snarl that is a bridge pair cannot be contained within another a snarl.

For a snarl $\{x, y\}$ in $B(D)$, let X be the path in $T(D)$ connecting $a_{T(D)}(x)$ and $a_{T(D)}(y)$. X starts and ends with net vertices and alternates between chain and net vertices. If $\{x, y\}$ is a chain pair then X consists only of the net vertex which both x and y project to in $T(D)$. The *net graph* Y for $\{x, y\}$ is a bidedged graph as follows (Fig. 3(E)):

- The vertices V_Y are the subset of vertices in $B(D)$ that project to net vertices in X .
- The grey edges in E_Y are the subset of grey edges in $E_{B(D)}$ that connect members of V_Y .
- There is a black edge e connecting each pair $\{x', y'\}$ of distinct vertices in V_Y not equal to $\{x, y\}$ whose incident black edges in $B(D)$ both project to the same simple cycle in $V_{C(D)}$ and are connected by a path that starts and ends with black edges.

We can use net graphs to determine if snarls are ultrabubbles. The net graph for a snarl $\{x, y\}$ is *bridgeless* if it does not contain a vertex other than x or y without an incident black edge.

Lemma 3. *A snarl $\{x, y\}$ in $B(D)$ is an ultrabubble iff its net graph and the net graph of each snarl contained in $\{x, y\}$ is acyclic and bridgeless.*

Proof. IF: If the separated component of $\{x, y\}$ is not acyclic then it contains a directed cycle S . Let $e = \{x', y'\}$ be a grey edge in S . By definition e is contained within exactly one net graph X . If x' is not incident with a black edge in X then its incident black edge in $B(D)$ is a bridge and x' cannot be a member of a directed cycle in $B(D)$, therefore let $\{x', z'\}$ be the black edge in X incident with x' . If S does not include z' then there must exist a directed walk in $B(D)$ from \hat{x}' to y' that excludes z' , but as the black edges incident with x' and z' project to the same simple cycle in $C(D)$, by Lemmas 6 and 11, the deletion of these black edges disconnects $B(D)$, separating \hat{x}' and y' , and implying no such directed walk excluding z' can exist. S therefore contains x', y', z' and, by the same logic, the node w' in X connected by a black edge to y' . If w' and z' are not connected by a grey edge then add the nodes in S that they are connected to by a grey edge in X to this set. Continuing this set extension we must ultimately define a directed cycle in X , therefore any cycle

S in the separated component must define one or more cycles in a net graph. If the separated component contains a tip vertex x' then the incident black edge is by definition a bridge, therefore the net graph containing x is not bridgeless.

ONLY IF: Each black edge $\{x', y'\}$ in a net graph X represents a portion of a simple cycle in $C(D)$, there therefore exists a directed path between x' and y' in $B(D)$ that starts and ends with the black edges incident with x' and y' . If there exists a directed cycle S in X then for each black edge in S we can replace it with a corresponding directed path in $B(D)$ and so define a valid directed cycle in $B(D)$. If there exists a node x' in X not equal to x or y and without an incident black edge then $\{x', \hat{x}'\}$ is a bridge edge in $B(D)$. It is easily verified that either there exists a tip or a directed cyclic walk in the separable component.

For a chain pair $\{x, y\}$ in $B(D)$ let the *contained chain pairs* be the chain pairs whose chain edges in $T(D)$ are:

- reachable from $a_{T(D)}(x)$ without passing through the chain edge of $\{x, y\}$ and which,
- on the path from $a_{T(D)}(x)$, first visit the incident chain vertex and then the incident net vertex.

Similarly for a bridge pair $\{x, y\}$ in $B(D)$ let the *contained chain pairs* be the chain pairs whose chain edges in $T(D)$ are:

- reachable from a vertex on the path X between $a_{T(D)}(x)$ and $a_{T(D)}(y)$ without passing through an edge in X or the projection in $T(D)$ of the bridge edges incident with x or y , and which,
- on the path from a vertex in X , first visit the incident chain vertex and then the incident net vertex.

Lemma 4. *For a chain pair or bridge pair $\{x, y\}$ in $B(D)$ the set of contained snarls is equal to its contained chain pairs.*

Proof. Let X be the component of $C(D)$ containing the projection of x and y after the deletion of the projection of the black edges incident with x and y . From Theorem 1 and given that homomorphisms preserve connectedness, it follows that the vertex induced subgraph in $B(D)$ of vertices that project to a vertex in X is the separated component of x and y . It follows that only snarls whose separated components' vertex projections are contained in X can be contained in $\{x, y\}$. It is easily verified from the definitions that this is equal to the set of the contained chain pairs for $\{x, y\}$.

Theorem 2. *A snarl $\{x, y\}$ in $B(D)$ is an ultrabubble iff its net graph and the net graph of each its contained chain pairs is acyclic and bridgeless.*

Proof. Follows from Lemmas 3 and 4.

Given Theorem 2 we now sketch an algorithm to compute the set of ultrabubbles for a given bidirected graph $B(D)$:

1. Calculate $C(D)$ (e.g. using the algorithm described in [12]).
2. Calculate $T(D)$.
3. For each chain pair label its chain edge in $T(D)$ with whether the chain pair's net graph is acyclic and bridgeless.
4. For each chain pair, traversing from its chain edge in $T(D)$, use depth first search to determine if its net graph and the net graph of each its contained chain pairs is acyclic and bridgeless, using the labels of the chain edges, and reporting the chain pair as an ultrabubble if so. (By recording if a chain pair is an ultrabubble as it is visited it is easily verified the complete traversal can be calculated by visiting each chain edge only once).
5. Calculate $D(D)$.
6. For each vertex x in $D(D)$ incident with exactly two edges let $\{x', y'\}$ be the bridge pair whose members project to x in $D(D)$. (There can be at most $|E_{B(D)}| - 1$ such bridge pairs). Calculate if the net graph and the contained chain pairs of $\{x, y\}$ are acyclic and bridgeless, reporting the bridge pair as an ultrabubble if true. (As an element in $T(D)$ can be contained in at most one such bridge pair the cost for this step is $O(|V_{B(D)}| + |E_{B(D)}|)$ for all such bridge pairs).

The computational complexity of steps 1, 2, 4, 5 and 6 is less than or equal to $O(|V_{B(D)}| + |E_{B(D)}|)$. For each chain vertex in step 3 the acyclicity of n net graphs is calculated, where n is the number of simple cycles incident with the vertex in $C(D)$. In the worst case, this step has a complexity of $O(|E_{B(D)}||V_{B(D)}|)$, which occurs when all vertices in $B(D)$ are 3-BEC. The cost of step 3 therefore dominates and the worst case complexity of the entire algorithm is $O(|E_{B(D)}||V_{B(D)}|)$. However, if the size of the largest net subgraph is bounded by a constant (which in practice it likely is) the expected running time will be linear in the size of graph.

2.7 Rooted Cactus Trees, Ultrabubbles and Genetic Sites

One particularly attractive feature of superbubbles is that they have a nested containment relationship, so that a digraph is partitioned into a set of top level superbubble components and other graph members not contained in a superbubble component, and each top level superbubble component then contains one or more child superbubbles, forming a tree structure. The situation is more complex for snarls and ultrabubbles, in that the separated component of snarls can overlap (Fig. 4), such that each partially contains the other. To create a properly nested hierarchy of snarls it is therefore necessary to exclude some snarls.

Given Lemma 4, to define a hierarchy of snarls that are chain pairs it is sufficient to pick a chain vertex as the root in each component of the cactus forest (e.g. Fig. 4) and only including the chain pairs contained by the root chain, using the definition of chain pair containment defined above. Note that this naturally orients the cyclic chains of chain pairs, breaking them by the chosen chain edge nearest the root.

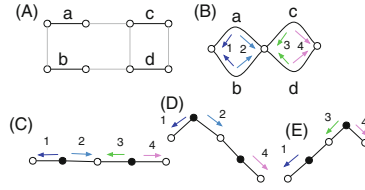


Fig. 4. Overlapping snarls. (A) A bidirected graph, its corresponding (B) cactus graph and (C) cactus tree. The snarl numbered 2 contains the snarl numbered 4, similarly the snarl numbered 3 contains the snarl numbered 1. The snarls numbered 2 and 3 overlap. (D–E) Two possible rootings for the cactus tree are shown, each of which defines a properly nested set of snarls.

Snarls that are bridge pairs are not naturally organized hierarchically. However, if the objective is to get a decomposition that contains the maximum number of ultrabubbles then the presence of bridge edges actually simplifies the problem, because a bridge edge cannot be contained within an ultrabubble. Hence if a bidged graph contains bridge edges, we can pick a subset of bridge pairs, and use each bridge pair to define a hierarchy of its contained chain pairs.

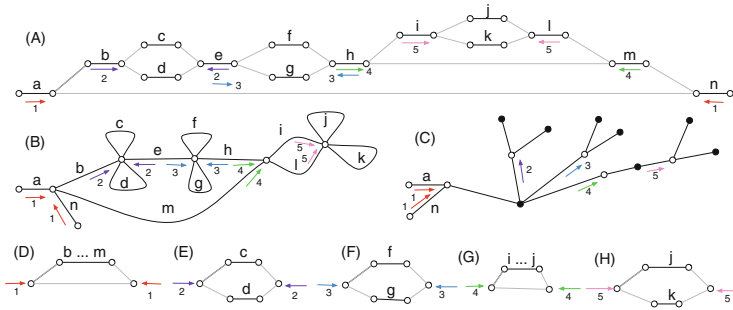


Fig. 5. (A) A bidged graph $B(D)$ with nested ultrabubbles indicated by pairs of numbered arrows. (B) $C(D)$ for $B(D)$. (C) $T(D)$ for $B(D)$. (D–H) The net graphs for the ultrabubbles in $B(D)$.

One of our motivations for investigating ultrabubbles was to define a decomposition of a bidirected graph representing genome variations into ‘sites’, groupings of paths representing subsequences into ‘alleles’, each representing an alternative at a particular location within a genome. Given a nesting of the ultrabubbles, we can envision that this nesting structure could play a powerful role in decomposing genotyping problems. This is illustrated in Fig. 5, which shows a bridge pair (arrows numbered 1) defining a *top-level* ultrabubble. Within this ultrabubble are a series of nested ultrabubbles and chains (*second-level* chain containing snarls numbered 2, 3 and 4 and a *third-level* ultrabubble numbered 5). In a genome problem, such a structure can easily arise with nested indels

and substitutions. The genotype problem can be viewed as the problem of establishing a consistent genotype of each ultrabubble’s net graph in a coordinated fashion.

3 Results

We implemented the ultrabubbles algorithm described above within the `vg` software package (<http://github.com/vgteam/vg>), where it is used to decompose graphs into sites for variant calling. Ultrabubbles can also be computed directly by running `vg stats -u`. To root the decomposition we picked the largest top-level chain, which consists of bridge pairs. Here we present the results of running this decomposition on a graph for human chromosome 1 constructed from the (roughly 6.5 million) variant calls from phase 3 of the 1000 Genomes Project [5]. The graph contained 19,917,881 nodes and 26,782,661 edges and the runtime was 23 min using a maximum of 49 G RAM on a single 2.27 GHz Intel Xeon core (4 min and 30 G of RAM were spent loading the graph into memory, a process that can be made an order of magnitude more efficient by switching the implementation to use `xg`, `vg`’s succinct representation).

Table 1 shows the relative proportion of each of these structures. The first three rows describe the top-level ultrabubble decomposition, which covers exactly every base in the input graph. The second three rows display the same statistics but for structures that are entirely contained within top-level ultrabubbles or snarls. The remaining rows describe the third and deepest nesting level, which is contained within second level ultrabubbles or snarls. Every base within the graph is part of either a top level chain, ultrabubble or snarl in this decomposition.

Table 1. Coverage statistics for the ultrabubble decomposition of the human chromosome 1 variant graph.

Structure	Nesting level	Count	Coverage (bp)	Coverage (pct)
Chains	Top	1	221,715,143	86.60
Ultrabubbles	Top	5,554,903	12,539,619	4.90
Snarls	Top	75	21,775,387	8.50
Chains	Second	919	20,594,450	8.04
Ultrabubbles	Second	533,252	1,199,777	0.47
Snarls	Second	0	0	0
Chains	Third	67	495	0.00
Ultrabubbles	Third	694	1,623	0.00
Snarls	Third	0	0	0

Figure 6 shows the size distribution of the top-level ultrabubble and snarl sizes. All but 22 top-level ultrabubbles (totaling 3,251 bases) are 100 bases long

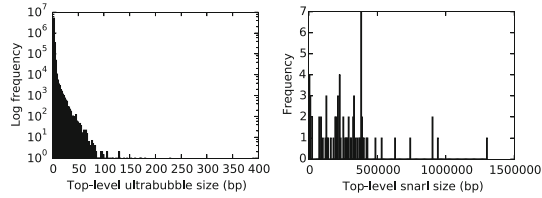


Fig. 6. Histograms of top-level ultrabubble and snarl sizes in number of bases, as found in the 1000 Genomes graph for chromosome 1.

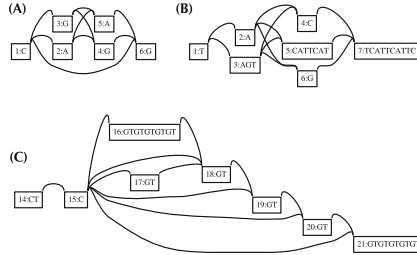


Fig. 7. Ultrabubbles found in the 1000 Genomes-derived graph for chromosome 1. (A) Two adjacent SNPs inside a deletion (chr1:209,887,366). (B) A more complex combination of SNP and indel events (chr1:237,977,845). (C) Copy number changes in a GT repeat (chr1:1,200,943).

or shorter. If we consider such sites “easy” to call, along with top-level chains, then we can assign roughly 91.5% of chromosome 1 into this category. Figure 7 displays three examples of such small ultrabubbles. The remaining 9.5% of cases are found in a small number of relatively large snarls.

4 Discussion and Conclusion

We have presented a partial decomposition of a bidirected graph into a set of nested snarls and ultrabubbles. We believe this solves an important problem in using graphs for representing arbitrary genetic variations by defining a decomposition that determines sites and alleles.

As the decomposition is only partial, not all elements in a graph will necessarily fit into one of the ultrabubbles. However, we demonstrate that for an existing large library of variation (1000 Genomes) the large majority of sites are either invariant or described by simple, top-level ultrabubbles.

For bases outside of these easy sites it is possible to imagine further subclassification. For example, classifying snarls that contain tips but are acyclic might define a useful class of subgraph common in some subproblems (e.g. sequence assembly). Similarly, characteristic structures representing genomic phenomena, such as inversions and translocations, are imaginable. Beyond our initial investigation, a more thorough evaluation of how much of a graph fits within a snarl, ultrabubble, or one of these more complex structures would be a useful exercise.

As an alternative to further subclassification, in the context of assembly, various error correction algorithms have been proposed to remove graph elements and reduce the complexity of the graph, and therefore correspondingly increase the fraction of the graph that is contained within an ultrabubble structure. We foresee the cactus graph structure providing a useful basis for exploring such algorithms.

Acknowledgements. This work was supported by the National Human Genome Research Institute of the National Institutes of Health under Award Number 5U54HG007990 and grants from the W.M. Keck foundation and the Simons Foundation.

Appendix 1

Lemma 5. *A pair of vertices x, y are in the same component of $B(D)$ iff their projections are in the same component of $C(D)$.*

Proof. IF: Follows given that by definition no pair of vertices not connected in $B(D)$ project to the same vertex in $C(D)$. ONLY IF: Follows given that $a_{C(D)}$ is a graph homomorphism from $B(D)$ to $C(D)$ and graph homomorphisms preverse connectedness.

Lemma 6. *For a subset of edges $X \subset E_{B(D)}$, if the removal of the projection of X disconnects $C(D)$, then the removal of X disconnects $B(D)$.*

Proof. Follows given that graph homomorphisms preverse connectedness.

Lemma 7. *The vertices in $C(D)$ are the equivalence classes of 3-BEC in $B(D)$.*

Proof. Each pair of vertices $B(D)$ that project to the same vertex in $C(D)$ are either/or-both connected by a path of grey edges (and hence 3-BEC) or connected by at least three black-edge-disjoint paths (using Menger's theorem).

Lemma 8. *A black edge in $B(D)$ is a bridge edge iff its projection in $C(D)$ is a bridge edge.*

Proof. Let $e = \{x, \hat{x}\} \in E_{B(D)}$.

ONLY IF: Suppose e is a bridge. As e is a bridge the vertices X reachable from x without visiting \hat{x} are black edge connected only by e to the vertices X' reachable from \hat{x} without visiting x . Given Lemma 7, it follows that the projection of X and the projection of X' are disjoint, therefore the projection of e is a bridge.

IF: Suppose e is not a bridge but its projection is. By definition there exists a path in $B(D)$ from x to \hat{x} that does not include e . As $a_{C(D)}$ is a homomorphism, the projection of that path connects $a_{C(D)}(x)$ and $a_{C(D)}(\hat{x})$ without traversing $b_{C(D)}(x)$ implying that it is not a bridge, a contradiction.

Lemma 9. *A maximal set of vertices in $C(D)$ is 2-EC iff the union of its members is a 2-BEC equivalence class of vertices in $B(D)$.*

Proof. Delete the black bridge edges in $B(D)$ and the bridge edges in $C(D)$ to create $B(D)'$ and $C(D)'$, respectively. Each component in $B(D)'$ is, by definition 2-BEC, and similarly each component in $C(D)'$ is 2-EC. The proof follows from Lemmas 5 and 8, by showing there exists a bijection between components in $B(D)'$ and $C(D)'$ such that for each component X in $B(D)'$ all the vertices in X project to vertices in the same component in $C(D)'$.

A *cut pair* is a pair of edges whose deletion disconnects the graph.

Lemma 10. *A pair of edges in a 2-EC component of a cactus graph is a cut pair iff both edges are contained within the same simple cycle.*

Proof. By definition, a 2-EC component of a cactus graph is a set of simple cycles connected by articulation (cut) vertices. It is easily verified that such a graph is and can only be disconnected by a pair of edges if they occur within one such simple cycle.

Lemma 11. *A pair of black edges (d, e) in a 2-BEC component X of $B(D)$ is a cut pair iff its projection is a cut pair in $C(D)$.*

Proof. Let X' be a vertex induced subgraph of the projection of X . By Lemma 9, X' is a 2-EC component in $C(D)$.

IF: If the deletion of the projection of d and e disconnects X' then, using Lemma 6, the deletion of d and e disconnects X .

ONLY IF: If the projection of d and e are not a cut pair, by the definition of a cactus graph and Lemma 10 the projection of d and e in X' are each members of two distinct simple cycles. If the projection of d (similarly e) were a self loop then its endpoints are 3-BEC, implying that after the deletion of d and e its endpoints remain connected. This is impossible if the deletion of d and e disconnect the 2-EC component, hence each simple cycle containing the projection of d or e has length greater than one. For any pair of distinct vertices x, y in $B(D)$ that project to the same vertex in $C(D)$, there exists a path in $B(D)$ that connects them that excludes their incident black edges, because by Lemma 7 they are 3-BEC, and are therefore either connected by a path of grey edges or, by Menger's theorem, connected by at least three edge disjoint paths containing black edges. From this observation it is easily verified that the endpoints of d (and similarly e) must be connected by a path Y in $B(D)$ that includes the black edges that project to the simple cycle containing d , in the order of the cycle, and which excludes both d and e . This implies the endpoints of d (similarly e) remain connected after the deletion of d and e , contradicting the claim they are a cut pair.

Appendix 2

Lemma 12. *Each snarl $\{x, y\}$ in $B(D)$ is either a chain pair or bridge pair.*

Proof. Using Lemma 5, both x and y must project to a vertex in the same component of $C(D)$ as they are connected in $B(D)$.

Let d and e be the black edges incident with x and y , respectively. If d is a bridge then e must be a bridge, or else by definition e connects two vertices in a 2-EC component X , the removal of d and e cannot therefore disconnect X , and therefore y and \hat{y} , violating the snarl separation criteria. Using Lemma 8, in this case the projections of d and e must therefore also be bridges. If d and e are both bridge edges but x and y do not project to the same vertex in $D(D)$ (and are therefore not a bridge pair) there exists an intermediate bridge edge $b_{D(D)}(z, \hat{z})$ on the path between $a_{D(X)}(x)$ and $a_{D(X)}(y)$. The deletion d , e and $\{z, \hat{z}\}$ for $B(D)$ disconnects $B(D)$ into distinct components, one contains x and z , one contains \hat{z} and y , one contains \hat{x} and one contains \hat{y} . This implies $\{x, z\}$ and $\{\hat{z}, y\}$ each fulfill the separation criteria, contradicting the minimality of $\{x, y\}$.

If d and e are not bridges both must be in the same 2-BEC component or contradict the separation criteria, by the same reasoning as earlier. In this case, Lemma 9 implies both d and e must project edges in the same 2-EC component in $C(D)$. Lemmas 10 and 11 further imply they must project to edges in the same simple cycle. If x and y do not project to the same vertex in $C(D)$ (and are therefore not a chain pair) then there exists an intermediate black edge $b_{C(D)}(z, \hat{z})$ on the path between $a_{C(D)}(x)$ and $a_{C(D)}(y)$ that excludes $d_{D(D)}(\hat{x})$ and $d_{D(D)}(\hat{y})$. As with the case that both d and e were bridge edges, this similarly contradicts the minimality of $\{x, y\}$.

Lemma 13. *Each chain pair or bridge pair $\{x, y\}$ in $B(D)$ is a snarl.*

Proof. Lemmas 6 and 10 imply that $\{x, y\}$ meet the separation criteria. It remains to prove that $\{x, y\}$ is minimal. If $\{x, y\}$ is not minimal then there must exist an intermediate edge $b_{C(D)}(z, \hat{z})$ on a path between $a_{C(D)}(x)$ and $a_{C(D)}(y)$ that excludes $d_{C(D)}(\hat{x})$ and $d_{C(D)}(\hat{y})$, and which, using Lemma 12, forms chain or bridge pairs with $a_{C(D)}(x)$ and $a_{C(D)}(y)$. As $a_{C(D)}(x) = a_{C(D)}(y)$ if $\{x, y\}$ is a chain pair, or $a_{D(D)}(x) = a_{D(D)}(y)$ if $\{x, y\}$ is a bridge pair, this is clearly impossible.

References

1. Alekseyev, M.A., Pevzner, P.A.: Breakpoint graphs and ancestral genome reconstructions. *Genome Res.* **19**(5), 943–957 (2009). <http://genome.cshlp.org/cgi/content/abstract/19/5/943>
2. Birmelé, E., Crescenzi, P., Ferreira, R., Grossi, R., Lacroix, V., Marino, A., Pisanti, N., Sacomoto, G., Sagot, M.-F.: Efficient bubble enumeration in directed graphs. In: Calderón-Benavides, L., González-Caro, C., Chávez, E., Ziviani, N. (eds.) SPIRE 2012. LNCS, vol. 7608, pp. 118–129. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34109-0_13
3. Brankovic, L., Iliopoulos, C.S., Kundu, R., Mohamed, M., Pissis, S.P., Vayani, F.: Linear-time superbubble identification algorithm for genome assembly. *Theor. Comput. Sci.* **609**, 374–383 (2015). <http://linkinghub.elsevier.com/retrieve/pii/S0304397515009147>

4. de Bruijn, N.G.: A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen* **1**(49), 758–764 (1946)
5. Consortium, G.P., et al.: A global reference for human genetic variation. *Nature* **526**(7571), 68–74 (2015)
6. Edmonds, J., Johnson, E.L.: Matching: a well-solved class of integer linear programs. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) *Combinatorial Optimization — Eureka, You Shrink!*. LNCS, vol. 2570, pp. 27–30. Springer, Heidelberg (2003). doi:[10.1007/3-540-36478-1_3](https://doi.org/10.1007/3-540-36478-1_3)
7. Harary, F., Uhlenbeck, G.E.: On the number of husimi trees: I. *Proc. Natl. Acad. Sci. U.S.A.* **39**(4), 315–322 (1953). http://www.ncbi.nlm.nih.gov/sites/entrez?Db=pubmed&Cmd=Retrieve&list_uids=16589268&dopt=abstractplus
8. Iliopoulos, C.S., Kundu, R., Mohamed, M., Vayani, F.: Popping superbubbles and discovering clumps: recent developments in biological sequence analysis. In: Kaykobad, M., Petreschi, R. (eds.) *WALCOM 2016*. LNCS, vol. 9627, pp. 3–14. Springer, Heidelberg (2016). doi:[10.1007/978-3-319-30139-6_1](https://doi.org/10.1007/978-3-319-30139-6_1)
9. Medvedev, P., Brudno, M.: Maximum likelihood genome assembly. *J. Comput. Biol.: J. Comput. Mol. Cell Biol.* **16**(8), 1101–1116 (2009). <http://www.liebertonline.com/doi/abs/10.1089/cmb.2009.0047>
10. Myers, E.W.: The fragment assembly string graph. *Bioinformatics* **21**(2), ii79–ii85 (2005). http://bioinformatics.oxfordjournals.org/content/21/suppl_2/ii79.long. (Oxford, England)
11. Onodera, T., Sadakane, K., Shibuya, T.: Detecting superbubbles in assembly graphs. In: Darling, A., Stoye, J. (eds.) *WABI 2013*. LNCS, vol. 8126, pp. 338–348. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-40453-5_26](https://doi.org/10.1007/978-3-642-40453-5_26)
12. Paten, B., Diekhans, M., Earl, D., John, J.S., Ma, J., Suh, B., Haussler, D.: Cactus graphs for genome comparisons. *J. Comput. Biol.: J. Comput. Mol. Cell Biol.* **18**(3), 469–481 (2011). <http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pubmed&id=21385048&retmode=ref&cmd=prlinks>
13. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A.* **98**(17), 9748–9753 (2001). <http://www.pnas.org/cgi/content/full/98/17/9748>
14. Pevzner, P.: *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, Cambridge (2000)
15. Sung, W.K., Sadakane, K., Shibuya, T., Belorkar, A., Pyrogova, I.: An $O(m \log m)$ -time algorithm for detecting super bubbles. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **12**(4), 770–777 (2015). <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6998850>
16. Zerbino, D.R., Birney, E.: Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18**(5), 821–829 (2008). <http://www.genome.org/cgi/content/full/18/5/821>

EPR-Dictionaries: A Practical and Fast Data Structure for Constant Time Searches in Unidirectional and Bidirectional FM Indices

Christopher Pockrandt^{1,2(✉)}, Marcel Ehrhardt¹, and Knut Reinert¹

¹ Department of Computer Science and Mathematics, Freie Universität Berlin, Berlin, Germany

christopher.pockrandt@fu-berlin.de

² International Max Planck Research

School for Computational Biology and Scientific Computation, Berlin, Germany

<http://reinert-lab.de>

Abstract. The unidirectional FM index was introduced by Ferragina and Manzini in 2000 and allows to search a pattern in the index in one direction. The bidirectional FM index (2FM) was introduced by Lam et al. in 2009. It allows to search for a pattern by extending an infix of the pattern arbitrarily to the left or right. If σ is the size of the alphabet then the method of Lam et al. can conduct one step in time $\mathcal{O}(\sigma)$ while needing space $\mathcal{O}(\sigma \cdot n)$ using constant time rank queries on bit vectors. Schnattinger and colleagues improved this time to $\mathcal{O}(\log \sigma)$ while using $\mathcal{O}(\log \sigma \cdot n)$ bits of space for both, the FM and 2FM index. This is achieved by the use of binary wavelet trees.

In this paper we introduce a new, practical method for conducting an exact search in a uni- and bidirectional FM index in $\mathcal{O}(1)$ time per step while using $\mathcal{O}(\log \sigma \cdot n) + o(\log \sigma \cdot \sigma \cdot n)$ bits of space. This is done by replacing the binary wavelet tree by a new data structure, the *Enhanced Prefixsum Rank dictionary* (EPR-dictionary).

We implemented this method in the SeqAn C++ library and experimentally validated our theoretical results. In addition we compared our implementation with other freely available implementations of bidirectional indices and show that we are between ≈ 2.2 – 4.2 times faster. This will have a large impact for many bioinformatics applications that rely on practical implementations of (2)FM indices e.g. for read mapping. To our knowledge this is the first implementation of a constant time method for a search step in 2FM indices.

Keywords: FM index · Bidirectional · BWT · Bit vector · Rank queries · Read mapping

1 Introduction

It is seldom that new data structures or algorithms have such a large practical impact as full text indices had for biological sequence analysis. The so-called

next-generation sequencing (NGS) allows to produce billions of small DNA strings called *reads*, usually of length 100–250. It is an invaluable technology for a multitude of applications in biomedicine. In many of these applications finding the positions of the strings in a reference (i.e., a large string of length $\approx 10^7 - 10^{10}$) is the first fundamental step preceding downstream analyses. The strings in question can be over different alphabets like DNA (size 4), proteins (size 27 if ambiguity characters are counted), or reduced protein alphabets (like Murphy10 of size 10 which is used in transcriptome search in [9, 22]).

This has triggered a plethora of work in the field to implement fast and accurate read mappers or local search tools. Many of the popular programs like Bowtie2 [13], BWA [15], BWA-Mem [14], Masai [21], Yara [20], GEM [18], Lambda [9], and Rapsearch [22] use as their main data structure a version of the FM index [5] that was introduced by Ferragina and Manzini in 2000. The FM index is based on the Burrows-Wheeler transform (BWT) [3] of the given text, i.e., the genomes at hand, and conceptually some lookup tables containing counts of characters in prefixes of the text. In its original form it allows to search exactly for a pattern in *one direction* by matching the characters of the pattern with characters in the BWT [3] (i.e., extending a suffix of the pattern character by character to the left). It was later extended to the 2FM index by Lam et al. [11] and Schnattinger et al. [19]. The 2FM index allows to search in both directions, that means we can extend an infix of a pattern arbitrarily to the left or to the right. In order to reduce its space requirements, the count tables are in practice replaced by efficient bit vector data structures with rank support (see for example [10]). The search method of Lam et al. can conduct one search step in a 2FM index in time $\mathcal{O}(\sigma)$ while needing space $\mathcal{O}(\sigma \cdot n)$ using constant time rank queries on bit vectors. Schnattinger et al. improved this time to $\mathcal{O}(\log \sigma)$ while using $\mathcal{O}(\log \sigma \cdot n)$ bits of space for both, the FM and 2FM index. This is achieved by the use of binary wavelet trees introduced by Grossi et al. [8]. In the last years several theoretical results appeared that improved on this. However, none of those has found a way into a practical implementation.

In this paper we introduce a new method for conducting an exact search in a uni- and bidirectional FM index that needs $\mathcal{O}(1)$ time per step while using $\mathcal{O}(\log \sigma \cdot n) + o(\log \sigma \cdot \sigma \cdot n)$ bits of space. This is done by replacing the binary wavelet tree by a new data structure, the *Enhanced Prefixsum Rank dictionary* (EPR-dictionary). To our knowledge this is the first implementation of a constant time method for 2FM indices. We will show, that the method outperforms other implementations by several factors at the expense of a slight increase in memory usage resulting in a very practical method independent of the size of the alphabet used in the application.

In the following paragraph we will review the concepts of the FM and 2FM index as well as constant time rank queries very shortly (readers unfamiliar with this can find a more detailed description in the appendix).

1.1 Introduction to the FM and 2FM Index

Given a text T of length n over an ordered, finite alphabet $\Sigma = \{c_1, \dots, c_\sigma\}$ with $\forall 1 \leq i < \sigma : c_i <_{lex} c_{i+1}$, let $T[i]$ denote the character at position i in T , \cdot the concatenation operator and $T[1..i]$ the prefix of T up to the character at position i . T^{rev} represents the reversed text. We assume that T ends with a sentinel character $\$ \notin \Sigma$ that does not occur in any other position in T and is lexicographically smaller than any character in Σ . The FM index needs the Burrows-Wheeler transform (BWT) of T . The BWT is the concatenation of characters in the last column of all lexicographically sorted cyclic permutations of the string T . We will refer to the BWT as L .

In contrast to suffix trees or suffix arrays, where a prefix P of a pattern is extended by characters to the right (referred to as forward search $P \rightarrow Pc$ for $c \in \Sigma$), the FM index can only be searched using backward searches, i.e., extending a suffix P' by characters to the left, $P' \rightarrow cP'$. Performing a single character backward search of c in the FM index will require two pieces of information. First, $C(c)$, the number of characters in L that are lexicographically smaller than c , second, $Occ(c, i)$, the number of c 's in $L[1..i]$. Given a range $[a, b]$ for P ; i.e., the range in the sorted list of cyclic permutations that start with P , we can compute the range $[a', b']$ for cP as follows: $[a', b'] = [C(c) + Occ(c, a - 1) + 1, C(c) + Occ(c, b)]$.

The 2FM index maintains two FM indices \mathcal{I} and \mathcal{I}^{rev} , one for the original text T and one for the reversed text T^{rev} . Searching a pattern left to right on the original text (i.e., conducting a forward search) corresponds to a backward search in \mathcal{I}^{rev} ; searching a pattern right to left in the original text corresponds to a backward search in \mathcal{I} . The difficulty is to keep both indices *synchronized* whenever a search step is performed. W.l.o.g. we assume that we want to extend the pattern to the right, i.e., perform a forward search $P \rightarrow Pc_j$ for some character c_j . First, the backward search $P^{rev} \rightarrow c_j P^{rev}$ is carried out on \mathcal{I}^{rev} and its new range $[a'_{rev}, b'_{rev}] = [C(c_j) + Occ(c_j, a_{rev} - 1) + 1, C(c_j) + Occ(c_j, b_{rev})]$ is computed. The new range in \mathcal{I} can be calculated using the interval $[a, b]$ for P in \mathcal{I} and the range size of the reversed texts index $[a', b'] = [a + smaller, a + smaller + b'_{rev} - a'_{rev}]$. To compute *smaller*, e.g. Lam et al. [11] propose to perform $\mathcal{O}(\sigma)$ backward searches $P^{rev} \rightarrow c_i P^{rev}$ for all $1 \leq i < j$ and sum up the range sizes, i.e., $smaller = \sum_{1 \leq i < j} Occ(c_i, b_{rev}) - \sum_{1 \leq i < j} Occ(c_i, a_{rev} - 1)$ leading to a total running time of $\mathcal{O}(\sigma)$.

The implementation of the occurrence table Occ is usually *not* done by storing explicitly the values of the entire table. Instead of storing the entire $Occ : \Sigma \times \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ one uses the more space-efficient *constant time rank dictionary*: for every $c \in \Sigma$ a bit vector $B_c[1..n]$ is constructed such that $B_c[i] = 1$ if and only if $L[i] = c$. Thus the occurrence value equals the number of 1's in $B_c[1..i]$, i.e., $Occ(c, i) = rank(B_c, i)$. Jacobson [10] showed that rank queries can be answered in constant time using only $o(n)$ additional space per bit vector by employing a sum of two count arrays (i.e., *blocks* and *superblocks*) and a final *in-block* count. Since then many other constant time rank query data structures have been proposed. For an overview we refer the reader to [17]

containing a comparison of various implementations. For readers unfamiliar with *2-level rank dictionaries*, an explanation is given in the appendix.

1.2 Recent Improvements on the FM and 2FM Index

For large alphabets, it is not practical to maintain for each character a bit vector with rank support. In 2003 Grossi et al. [8] proposed the use of a more space efficient data structure for the FM index, called the *(binary) wavelet tree* (WT) that was later used by Schnattinger [19] for an implementation of bidirectional FM indices. It is a binary tree of height $\mathcal{O}(\log \sigma)$ with a bit vector of length n with rank support at each level. This reduces the space consumption by a factor of $\mathcal{O}(\frac{\log \sigma}{\sigma})$ in trade-off for an increased running time of $\mathcal{O}(\log \sigma)$. Schnattinger used the fact that not only the rank query for a given character c can be computed in $\mathcal{O}(\log \sigma)$ but also the *smaller* value can be computed in the same asymptotic time which is quite convenient for the 2FM index. Ferragina et al. proposed a new data structure in 2007 [6], the multi-ary wavelet tree, which could be used to speed up the needed rank queries of 2FM indices. In 2013 Belazzougui et al. proposed the first constant-time bidirectional FM index [1] using minimal perfect hashing, of which to our knowledge no implementation exists (see also [2] for an extended version). Our solution is based on the use of support data structures of bit vectors with rank support and the direct use of the BWT. This proved so far to be very fast in practice, in particular due to the `popcount` machine operation. The resulting data structure is implemented in the latest release 2.3 of the SeqAn library.

2 Theoretical Results

In this section we present the main results of this paper. They are based on a simple observation and a new bit vector data structure with rank query support which allows us to improve upon the results of Lam and Schnattinger. Our proposed method runs in constant time per step while using $\mathcal{O}(\log \sigma \cdot n) + o(\log \sigma \cdot \sigma \cdot n)$ bits of space for small alphabets (i.e., $\sigma < \log(n)/\log \log(n)$) which is in theory inferior in space consumption to the results of mentioned above (see [1]), but in practice very fast, and presents to our knowledge the first constant time implementation of 2FM indices with this space complexity.

Our first observation is simple. Instead of defining a bit vector for each $c \in \Sigma$ to map characters equal to c in L to 1's, we suggest using prefix sum bit vectors PB_c , i.e., $PB_c[i] = 1$ if and only if $L[i] \leq_{lex} c$ for $c \in \Sigma$.

Theorem 1. *A step in a bidirectional search can be performed in time $\mathcal{O}(1)$ using $\mathcal{O}(\sigma \cdot n)$ bits of space.*

Proof. We define $Prefix\text{-}Occ(c_j, i) = rank(PB_{c_j}, i)$; that means it counts the number of occurrences of a character lexicographically smaller or equal than c_j up to position i . $Prefix\text{-}Occ(c_j, i)$ and thus the *smaller* value for the 2FM

index can now be computed by a single rank query $rank(PB_{c_j}, i)$, the original $Occ(c_j, i)$ value for backward searches needs only two rank queries and a subtraction, namely $Occ(c_j, i) = rank(PB_{c_j}, i) - rank(PB_{c_{j-1}}, i)$ (for the lexicographically smallest character c_0 no subtraction is necessary).

Note that the bit vector for the lexicographically largest character can be omitted, since all bits will be set to 1 and thus $rank(PB_{c_\sigma}, i) = i, \forall 1 \leq i \leq n$.

Our next idea is the main result of this work and will allow us to reduce the space complexity for both the FM and the 2FM index while maintaining the optimal running time of $\mathcal{O}(1)$ per search step. Instead of using normal bit vectors we use directly the binary encoding of the BWT (an idea already used by BWT-SW [12]). We call our data structure *EPR-dictionary*, short for *Enhanced Prefixsum Rank dictionary*.

2.1 The EPR-Dictionary

The general idea of the EPR-dictionary is as follows. Assuming an ordered alphabet $\Sigma = \{c_1, \dots, c_\sigma\}$, each character c_i is encoded by the binary value $ord_2(c_i)$ of its rank i . Conceptually, we use the binary representation of the BWT to derive from it a *spaced* bit vector representation for PB_c for each $s \in \Sigma$. Then we compute the auxiliary data structures (i.e., blocks and superblocks (see also appendix)) for each of those vectors. After we have those auxiliary structures which only need $o(n)$ bits space, we *delete* the bit vectors and only retain the BWT. In practice the blocks and superblocks are computed directly by a linear scan on the BWT. For the last in-block query, we show how to derive the counts $Prefix-Occ(c_j, i)$ from the BWT in constant time using a number of logical and arithmetic operations.

W.l.o.g. we assume that the block length is an even multiple of $\log \sigma$ to avoid case distinctions in the proof. In practice this holds since the in-block query is performed with popcounts on registers the length of which is a power of 2. All bitmasks used for computing the in-block rank are exactly as long as a block. For a character c_j we define the *rank bitmask* $rb(c_j)$ to be a binary sequence of concatenations of the pattern $0^{\lceil \log \sigma \rceil - 1} 1 \cdot ord_2(c_j)$, i.e., $\lceil \log \sigma \rceil - 1$ many bits set to 0 followed by a 1 followed by the binary encoding of the character c_j . For the DNA alphabet with $\Sigma = \{A, C, G, T\}$ and its binary encodings $\{00, 01, 10, 11\}$ the rank bitmask for $G \in \Sigma$ is for example $rb(G) = 01 \cdot 10 \cdot 01 \cdot 10 \dots$

Counting the characters inside a block is done in two steps. The characters at even and odd positions are counted separately to generate space for an overflow bit. Therefore we need a bitmask M_E masking characters at even positions from the bit vector. M_E has 1s for each even block of length $\lceil \log \sigma \rceil$, i.e., $M_E = 00 \cdot 11 \cdot 00 \cdot 11 \cdot 00 \cdot 11 \dots$. Finally, we need a bitmask BM which filters out the lowest bit of each odd $\log \sigma$ block, i.e., $BM = 01 \cdot 00 \cdot 01 \cdot 00 \cdot 01 \cdot 00 \dots$ for $\sigma = 4$.

Step 1. We first take the characters at odd positions inside the corresponding block of the BWT, subtract it from $rb(c_i)$, which will result in the rightmost bit of even character positions to be set to 1 if and only if the character to the right

is smaller or equal to c_i . We then obtain exactly those bits by masking with BM .

$$B_{EPR}(c_i)_E = (rb(c_i) - (BWT \& M_E)) \& BM$$

Step 2. We then take the characters at even positions inside the corresponding block of the BWT by shifting them $\lceil \log \sigma \rceil$ bits to the right and masking with M_E . We can now continue as in step 1 by subtracting it from $rb(c_i)$, which will again result in a 1 bit in the rightmost bit of even character positions to be set to 1 if and only if the character to the right is smaller or equal to c_i . We then apply the bitmask BM to filter only these rightmost bits.

$$B_{EPR}(c_i)_O = (rb(c_i) - ((\gg_{\lceil \log \sigma \rceil} BWT) \& M_E)) \& BM$$

Finally both bit vectors are merged with one of them shifted by 1 to the left avoiding the rightmost bits to overlap. In practice this is faster than two popcount operations.

$$B_{EPR}(c_i) = B_{EPR}(c_i)_E | (\ll_1 B_{EPR}(c_i)_O)$$

Since we used the binary encoding of the BWT, note that the underlying rank queries have to be adapted to $Prefix\text{-}Occ(c_j, i) = rank(B_{EPR}(c_j), (i - 1) \cdot \lceil \log \sigma \rceil + 1)$. It follows directly that $Occ(c_j, i)$ can be computed in constant time by observing that

$$Occ(c_j, i) = \begin{cases} Prefix\text{-}Occ(c_j, i) - Prefix\text{-}Occ(c_{j-1}, i) & \text{if } j > 0 \\ Prefix\text{-}Occ(c_j, i) & \text{otherwise} \end{cases}$$

The EPR-transformed bit vector $B_{EPR}(c_j)$ is now a “normal” bit vector and thus allows us to compute the prefix sums for a string in constant time. This improves the running time of the 2FM index and makes it optimal in terms of speed.

Let us now take a look at the space consumption. For our exposition we define the block length of $\ell = \left\lfloor \frac{\log n}{2} \right\rfloor$ (if ℓ is not a multiple of $\lceil \log \sigma \rceil$ padding strategies can be applied). Given a $B_{EPR}(c_j)$, for the m -th superblock we count the number of 1’s (i.e., the number of occurrences of characters smaller or equal to c_j in the corresponding BWT) from the beginning of B_{EPR} to the end of the superblock in $M'[m][j] = rank(B_{EPR}(c_j), m \cdot \ell^2)$. As there are $\left\lfloor \frac{\lceil \log \sigma \rceil \cdot n}{\ell^2} \right\rfloor$ superblocks and σ characters, M' can be stored in

$$\mathcal{O} \left(\sigma \cdot \frac{\log \sigma \cdot n}{\ell^2} \cdot \log n \right) = \mathcal{O} \left(\sigma \cdot \log \sigma \cdot \frac{n}{\log n} \right) = o(\sigma \log \sigma \cdot n)$$

bits. For the m -th block we count the number of 1’s from the beginning of the overlapping superblock to the end of the block in $M[m][j] = rank(B_{EPR}[1 + k\ell..n](c_j), (m - k)\ell)$ where $k = \lfloor \frac{m-1}{\ell} \rfloor$ ℓ is the number of blocks left of the

$rb(G)$		01 10 01 10 01 10 01 10
$BWT \& M_E$	-	00 01 00 01 00 11 00 11
		- (C) - (C) - (T) - (T)
	=	01 01 01 01 00 11 00 11
BM	&	01 00 01 00 01 00 01 00
$B_{EPR}(G)_E$	=	01 00 01 00 00 00 00 00
(a) step 1		
$rb(G)$		01 10 01 10 01 10 01 10
$(\gg_{\lceil \log \sigma \rceil} BWT) \& M_E$	-	00 00 00 10 00 10 00 00
		- (A) - (G) - (G) - (A)
	=	01 10 01 00 01 00 01 10
BM	&	01 00 01 00 01 00 01 00
$B_{EPR}(G)_O$	=	01 00 01 00 01 00 01 00
(b) step 2		
$B_{EPR}(G)_E$		01 00 01 00 00 00 00 00
$\ll_1 B_{EPR}(G)_O$		10 00 10 00 10 00 10 00
$B_{EPR}(G)$	=	11 00 11 00 10 00 10 00
$popcount$	=	6
(c) retrieving $B_{EPR}(G)$		

Fig. 1. An example for $\Sigma = \{A, C, G, T\}$ that shows how to perform an in-block rank query for characters smaller or equal to G of the BWT substring $ACGCGTAT$. The resulting bit vector $B_{EPR}(G)$ has a 1 for each character smaller or equal to G ., i.e., all positions except those with a T .

overlapping superblock. M has $\lfloor \frac{\lceil \log \sigma \rceil \cdot n}{\ell} \rfloor$ entries for every character and can be stored in

$$\mathcal{O} \left(\sigma \cdot \frac{\log \sigma \cdot n}{\ell} \cdot \log \ell^2 \right) = \mathcal{O} \left(\sigma \cdot \log \sigma \cdot \frac{n \cdot \log \log n}{\log n} \right) = o(\sigma \log \sigma \cdot n)$$

bits.

Let P be a precomputed lookup table such that for each possible infix V of a bit vector $B_{EPR}(c_j)$ of length ℓ , $i \in \left[1.. \lfloor \frac{\ell}{\log \sigma} \rfloor \right]$ and $c_j \in \Sigma$ holds $P[V][i] = rank(V, (i-1) \cdot \lceil \log \sigma \rceil + 1)$. There are $2^\ell \cdot \lfloor \frac{\ell}{\log \sigma} \rfloor$ entries of value at most $\lfloor \frac{\ell}{\log \sigma} \rfloor$ and thus can be stored in

$$\mathcal{O} \left(2^\ell \cdot \frac{\ell}{\log \sigma} \cdot \log \frac{\ell}{\log \sigma} \right) = \mathcal{O} \left(2^{\frac{\log n}{2}} \cdot \log(n - \sigma) \cdot \log \log(n - \sigma) \right) = \mathcal{O}(\sqrt{n} \cdot \log n \cdot \log \log n) = o(n)$$

bits. Note that we do need this lookup table only *once*, since the position and counting of the bits set to 1 is the same for all characters.

Equivalent to Theorem 1, we do not need to store blocks and superblocks for c_σ since $rank(B_{EPR}(c_\sigma), i) = i, \forall 1 \leq i \leq n$.

Theorem 2 (Constant time prefix sum query). *One search step in an FM index or 2FM index can be performed in $\mathcal{O}(1)$ time using $\mathcal{O}(\log \sigma \cdot n) + o(\log \sigma \cdot n)$ bits of space.*

Proof. The BWT can be stored in $\mathcal{O}(\log \sigma \cdot n)$, all the blocks, superblocks, and lookup table P in $o(\log \sigma \cdot \sigma \cdot n)$ bits. A prefix sum rank query requires one superblock and block lookup as well as a constant number of arithmetic and logical operations on the last block which run all in constant time.

3 Experimental Results

In this Section we will conduct computational experiments to validate our theoretical findings and to compare our FM and 2FM indices to another available implementation. All of our tests were conducted on Debian GNU/Linux 7.1 with Intel® Xeon® E5-2667V2 CPUs at fixed frequency of 3.3 GHz to prevent dynamic overclocking effects. All data was stored on tmpfs, a virtual file system in main memory to prevent loading data just on demand during the search and thus effecting the speed of the search by I/O operations.

In the first part of the experiments we will test FM and 2FM indices with our new data structure (EPR) in comparison to the wavelet tree (WT) implementation which was previously the generic standard implementation in SeqAn [4]. Additionally we will run the same benchmarks for other available 2FM implementations, namely the bidirectional wavelet tree by Schnattinger et al. [19] which we will call 2SCH and the balanced wavelet tree implementation with plain bit vectors and constant-time rank support in the SDSL [7] which we will refer to as 2SDSL.

The 2BWT by Lam et al. [12] is not generic and only works for DNA alphabets. We also were not able to retrieve all hits when switching between forward and backward searches on the same pattern. Unfortunately we couldn't reach the authors and thus excluded 2BWT from our comparisons.

Another implementation that is worth mentioning is the affix array by Meyer et al. [16]. Even though the affix array implementation is generic, the construction algorithm did not terminate for alphabets other than DNA in a reasonable amount of time (several days). Unfortunately the affix array is not stand-alone but part of an application and does not provide a documented interface. Hence we were not able to include the affix array in our tests within a reasonable time frame. Meyer compares the running time of their affix array implementation with 2SCH and states that the affix array is faster by a factor of 1.26 to 2. From that we can conclude that our 2FM index implementation using the EPR-dictionary is expected to be faster than the affix array implementation (see below).

3.1 Runtime and Space Consumption

For the first benchmark we want to make a comparison with alphabets of different sizes to test the predicted independence from σ for the EPR implementation.

The alphabet sizes are inspired by bioinformatics applications and are of size 4 (DNA), 10 (reduced amino acid alphabet *Murphy10*), 16 (IUPAC alphabet) and 27 (protein alphabet).

We first generated a text of length 10^8 with a uniform distribution and sampled 1 million queries of length 200 from this text. The search in the FM and 2FM indices will determine the number of occurrences of the sampled strings. Our sampling will ensure that the text occurs at least once and the stepwise search is never prematurely stopped. This ensures that we have 200 million single steps in searches. The unidirectional FM indices perform backward searches while for 2FM indices we search the right half of the query first (using forward searches) and then extend the other half of the pattern to the left by backward searches. The results were also verified on real-world data. Running the same tests on chromosome 13 of the human genome (with unknown bases removed and a length of 10^8 bases left) did not show any measurable difference compared to the generated data sets.

In the following we will refer to WT and EPR as unidirectional FM indices and to 2WT and 2EPR as bidirectional FM indices, all part of the SeqAn library.

Table 1 gives an overview of the running times of all FM and 2FM index implementations. It shows the absolute runtimes as well as the speedup factor relative to the unidirectional resp. bidirectional wavelet tree implementation. WT, 2WT, 2SCH, 2SDSL are all based on wavelet trees. Our bidirectional wavelet tree implementation 2WT has a similar runtime compared to 2SDSL. It is slightly faster especially for small alphabets.

Table 1. Runtimes of various implementations in seconds and their speedup factors with respect to the unidirectional respectively bidirectional wavelet tree.

Index	DNA		Murphy10		IUPAC		Protein	
	Time	Factor	Time	Factor	Time	Factor	Time	Factor
WT	20.73s	1.00	52.35s	1.00	66.45s	1.00	85.55s	1.00
EPR	15.09s	1.37	22.27s	2.35	23.39s	2.84	25.31s	3.38
2WT	41.21s	1.00	66.59s	1.00	98.74s	1.00	120.96s	1.00
2EPR	20.09s	2.05	23.80s	2.80	24.39s	4.05	26.08s	4.64
2SDSL	43.54s	0.95	74.69s	0.89	89.07s	1.11	109.44s	1.11
2SCH	59.59s	0.69	91.30s	0.73	107.04s	0.92	129.98s	0.93

Compared to the wavelet tree implementations the EPR implementation is between 40% (for DNA) and 240% (Protein) faster for unidirectional indices and between 110% (for DNA) and 360% (Protein) faster for bidirectional indices.

Since we anticipate the application of 2EPR to bioinformatics applications, we also compared the runtime of all implementations using the complete human genome sequence. We again searched one million sampled strings of length 200 exactly as described above. The relative results were very similar to the ones in

Table 1, indeed even slightly better. 2SCH crashed with this data set. 2SDSL was the slowest implementation (46.69s) followed by 2WT (1.4 times as fast) and by 2EPR (2.1 times as fast as 2SDSL) which was again the fastest implementation.

Our experiments also show that we were indeed able to eliminate the $\log \sigma$ factor of wavelet trees in practice, as predicted by Theorem 2. While the runtime for the WT implementations grows for larger alphabets with $\log \sigma$ the runtime of EPR and 2EPR increases only slightly for larger alphabets which can be explained by larger indices and therefore more cache misses. This can be seen in the following Figure in which we plot the runtime for EPR for different alphabets and the runtime of WT divided by $\log \sigma$. The resulting times develop very similarly.

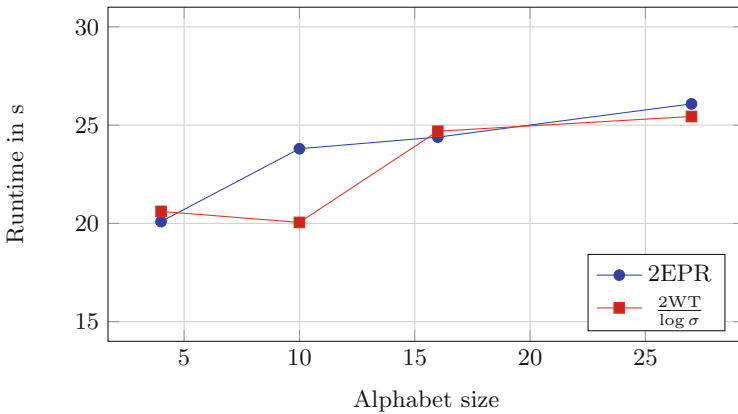


Fig. 2. Plot of the runtime for EPR for different alphabets and the runtime of WT divided by $\log \sigma$.

When we compare the runtimes of the EPR and 2EPR, they behave as expected, i.e., the unidirectional index is slightly faster, since in each step of the bidirectional index we have to synchronize two indices.

All indices implemented in SeqAn (WT, EPR, 2WT, 2EPR) support up to 3 levels for rank dictionary support: blocks, superblocks and ultrablocks. The tests presented here were performed with a 2-level rank dictionary similar to the one explained in Sect. 2.1 (or in the Appendix). Table 2 illustrates the practical space consumption for all previously discussed indices and of the affix array for DNA (larger alphabets did not finish within several days).

Please note, that the other implementations may use versions of rank dictionaries different to the simple one explained in Sect. 2.1. The numbers of FM indices given in Table 2 do neither account for storing the text itself nor for storing a compressed suffix array necessary to locate the matches in the text since the libraries use different implementations offering various space-time trade-offs. The running time of the backward and forward searches does not depend on it and

the compressed suffix array implementation is independent from the used rank dictionary and thus interchangeable. A typical compressed suffix array implementation as used in the 2SDSL takes $\frac{n}{\eta} \log n$ (when sampling on the text instead of the suffix array). For a sampling rate of 10% ($\eta = 10$) the space consumption for our experiments would be 253 MB and thus still much smaller than the affix array.

Table 2. Space consumption of the rank data structure in Megabyte of various implementations

Index	DNA	Murphy10	IUPAC	Protein
EPR	42	156	227	478
2EPR	84	311	454	955
WT	30	51	60	72
2WT	60	102	120	144
2SDSL	68	105	122	145
2SCH	75	108	123	146
AF	2670	-	-	-

The current implementation of the EPR and 2EPR in SeqAn interleaves the bit vector (i.e., the BWT) and precomputed block values but does not interleave superblock values. Reconsidering the design and storing block and superblock values close to the corresponding bit vector region could decrease the number of cache misses for one rank query to one cache miss and thus further improve the running time.

For larger alphabets one might also consider using a 3-level rank dictionary with smaller data types for blocks and superblocks which will reduce the space consumption noticeably at the expense of a higher runtime (i.e., for the protein alphabet we reduce the space consumption from 955 MB to 581 MB while increasing the runtime from 26.08 to 35.13 s). Thus it is still faster than the 2WT by a factor of 3.4. The increased running time is due to another array lookup and thus still constant-time per step.

3.2 Effect of the Low Order Terms for Space Consumption

In Table 3 we show how quickly the $o(\log \sigma \cdot \sigma \cdot n)$ data structures for rank queries can be neglected for growing n . For the WT and EPR implementations we measured the space needed for both the DNA and the IUPAC alphabet for $n = 10^4, 10^5, 10^6, 10^7, 10^8, 10^9$. We then divided the space consumption of both implementations by the factor in the \mathcal{O} -term, namely $\log \sigma \cdot n$.

For growing n the \mathcal{O} -term should dominate the low order o -term, hence we would expect the resulting number converge to a constant. This is indeed true, as can be seen in Table 3. The EPR implementation converges faster than the

WT, which is expected, since our o -term is larger than the one for the WT implementations. The effect of the o -terms falls for EPR from 10^5 to 10^6 by 35 resp. 6 percent, whereas the decline for WT is steeper with 84 and 123 percent. From size 10^7 on, the low order terms are clearly dominated by the \mathcal{O} -terms.

Table 3. Influence of the space consumption of the o -terms with increasing n .

Method (σ)	10^4	10^5	10^6	10^7	10^8	10^9
EPR (4)	2.4000	0.6000	0.4440	0.4292	0.4276	0.4274
EPR (16)	2.0000	1.2400	1.1680	1.1610	1.1602	1.1601
WT (4)	4.4000	0.6400	0.3480	0.3088	0.3056	0.3053
WT (16)	7.0000	0.8000	0.3580	0.3104	0.3057	0.3053

4 Conclusions

In this paper we have introduced a new data structure, the EPR-dictionary, that enables constant time prefix sum computations for arbitrary, finite alphabets in $\mathcal{O}(\log \sigma \cdot n) + o(\log \sigma \cdot \sigma \cdot n)$ bits of space and works directly on the BWT. This allows two important data structures, the FM and 2FM index, to perform single search steps in time $\mathcal{O}(1)$. We implemented the dictionary in the C++ library SeqAn and used it for an implementation of an FM and 2FM index. We compared its practical performance with the previous SeqAn implementation using wavelet trees and with other openly available implementations, among them the quasi standard for succinct data structures, the SDSL. We show that the EPR-dictionary implementation supports our theoretical claims, eliminates the $\log \sigma$ factor for searching in bidirectional indices, and performs between 40% and 360% faster than the wavelet tree implementation at the expense of a higher memory consumption. We compared our 2FM implementation against the available, open implementation of Schnattinger et al. (2SCH). Our implementation is between 3 to 5 times faster than 2SCH and 2.2 to 4.2 faster than the 2SDSL. We also showed that the additional space consumption is easily tolerable on normal hardware.

Acknowledgments. We would like to acknowledge Enrico Siragusa for his previous implementations of the FM index in SeqAn. The first author acknowledges the support of the International Max-Planck Research School for Computational Biology and Scientific Computing (IMPRS-CBSC). We also thank Veli Mäkinen and Simon Gog for very helpful remarks on a previous version of this manuscript during the Dagstuhl seminar 16351 “Next Generation Sequencing - Algorithms, and Software For Biomedical Applications”.

Appendix

In the appendix we give for the reader not familiar with FM and 2FM indices a short introduction.

Introduction to the FM and 2FM Index

Given a text T of length n over an ordered, finite alphabet $\Sigma = \{c_1, \dots, c_\sigma\}$ with $\forall 1 \leq i < \sigma : c_i <_{lex} c_{i+1}$, let $T[i]$ denote the character at position i in T , \cdot the concatenation operator and $T[1..i]$ the prefix of T up to the character at position i . T^{rev} represents the reversed text. We assume that T ends with a sentinel character $\$ \notin \Sigma$ that does not occur in any other position in T and is lexicographically smaller than any character in Σ . The FM index needs the Burrows-Wheeler transform (BWT) of T . The BWT is the concatenation of characters in the last column of all lexicographically sorted cyclic permutations of the string T (see Fig. 3 for an example). We will refer to the BWT as L .

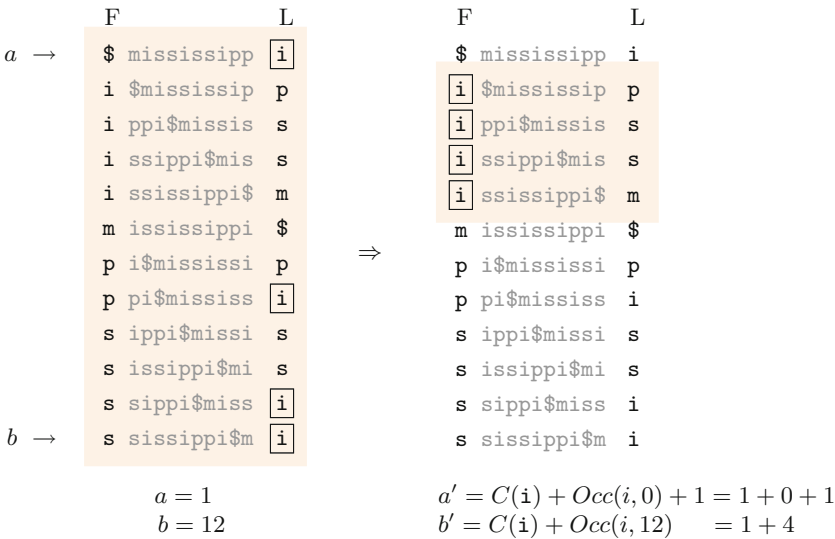


Fig. 3. First step of the backwards search for $P = ssi$ in the FM index for the text $T = mississippi\$$. The first interval $[a, b]$ is the whole range $[1, 12]$. From all matrix rows we search those beginning with the last pattern character $P[3] = i$. From $Occ(i, 1) = 0$ and $Occ(i, 12) = 4$ follows $a' = C(i) + 0 + 1 = 2$ and $b' = C(i) + 4 = 5$.

In contrast to suffix trees or suffix arrays, where a prefix P of a pattern is extended by characters to the right (referred to as forward search $P \rightarrow Pc$ for $c \in \Sigma$), the FM index can only be searched using backward search, i.e., extending a suffix P' by characters to the left, $P' \rightarrow cP'$. Performing a single character backward search of c in the FM index will require two pieces of information.

First, $C(c)$, the number of characters in L that are lexicographically smaller than c , second, $Occ(c, i)$, the number of c 's in $L[1..i]$. Given a range $[a, b]$ for P ; i.e., the range in the sorted list of cyclic permutations that starts with P , we can compute the range $[a', b']$ for cP as follows: $[a', b'] = [C(c) + Occ(c, a - 1) + 1, C(c) + Occ(c, b)]$. We will refer to the BWT together with tables C and Occ as *FM index* \mathcal{I} (see Fig. 3 for an example of one search step).

The 2FM index maintains two FM indices \mathcal{I} and \mathcal{I}^{rev} , one for the original text T and one for the reversed text T^{rev} . Searching a pattern left to right on the original text (i.e., conducting a forward search) corresponds to a backward search in \mathcal{I}^{rev} ; searching a pattern right to left in the original text corresponds to a backward search in \mathcal{I} . The difficulty is to keep both indices *synchronized* whenever a search step is performed. W.l.o.g. we assume that we want to extend the pattern to the right, i.e., perform a forward search $P \rightarrow Pc_j$ for some character c_j . First, the backward search $P^{rev} \rightarrow c_jP^{rev}$ is carried out on \mathcal{I}^{rev} and its new range $[a'_{rev}, b'_{rev}] = [C(c_j) + Occ(c_j, a_{rev} - 1) + 1, C(c_j) + Occ(c_j, b_{rev})]$ is computed. The new range in \mathcal{I} can be calculated using the interval $[a, b]$ for P in \mathcal{I} and the range size of the reversed texts index $[a', b'] = [a + smaller, a + smaller + b'_{rev} - a'_{rev}]$. To compute *smaller*, Lam et al. [11] propose to perform $\mathcal{O}(\sigma)$ backward searches $P^{rev} \rightarrow c_iP^{rev}$ for all $1 \leq i < j$ and sum up the range sizes, i.e., $smaller = \sum_{1 \leq i < j} Occ(c_i, b_{rev}) - \sum_{1 \leq i < j} Occ(c_i, a_{rev} - 1)$ leading to a total running time of $\mathcal{O}(\sigma)$ (See Fig. 4 for an illustration).

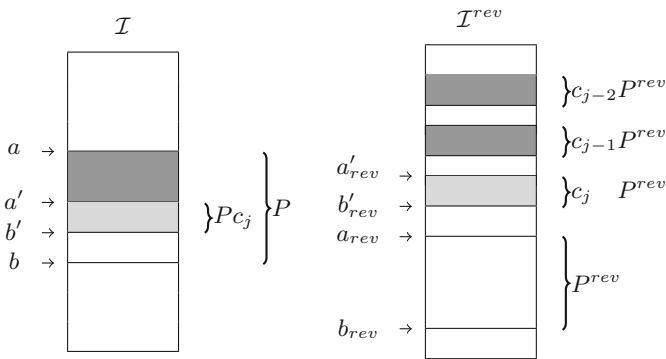


Fig. 4. When conducting a forward search $P \Rightarrow Pc_j$ we need to determine the subinterval of the suffix array interval for P which is depicted on the left. In order to determine the start, we can compute in \mathcal{I}^{rev} the size of the intervals for all characters smaller than c_j , depicted in dark gray on the right. The sum of all those sizes is exactly the needed offset from the beginning of the interval for P in \mathcal{I} .

The implementation of the occurrence table Occ is usually *not* done by storing explicitly the values of the entire table. Instead of storing the entire $Occ : \Sigma \times \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ one uses the more space-efficient *constant time rank dictionary*: for every $c \in \Sigma$ a bit vector $B_c[1..n]$ is constructed such

that $B_c[i] = 1$ if and only if $L[i] = c$. Thus the occurrence value equals the number of 1's in $B_c[1..i]$, i.e., $Occ(c, i) = rank(B_c, i)$. Jacobson [10] showed that rank queries can be answered in constant time using only $o(n)$ additional space per bit vector. Since then many other constant time rank query data structures have been proposed. For an overview we refer the reader to [17] containing a comparison of various implementations. Since we will make also use of this technique, we explain the most simple idea, namely the one for *2-level rank dictionaries* in the following paragraph.

Constant Time Rank Queries

In order to store partial prefix sums, the technique uses two levels of lookup table, called *blocks* and *superblocks*. Given a bit vector B of length n we divide it into blocks of length ℓ and superblocks of length ℓ^2 where

$$\ell = \left\lfloor \frac{\log n}{2} \right\rfloor.$$

For both, blocks and superblocks we allocate arrays M and M' of sizes $\lfloor \frac{n}{\ell} \rfloor$ and $\lfloor \frac{n}{\ell^2} \rfloor$ respectively (see Fig. 5 for an illustration).

For the m -th superblock we store the number of 1's from the beginning of B to the end of the superblock in $M'[m] = rank(B, m \cdot \ell^2)$. As there are $\lfloor \frac{n}{\ell^2} \rfloor$ superblocks, M' can be stored in $\mathcal{O}(\frac{n}{\ell^2} \cdot \log n) = \mathcal{O}(\frac{n}{\log n}) = o(n)$ bits. For the m -th block we store the number of 1's from the beginning of the overlapping superblock to the end of the block in $M[m] = rank(B[1 + k\ell..n], (m - k) \cdot \ell)$, where $k = \lfloor \frac{m-1}{\ell} \rfloor$ is the total number of blocks in all superblocks left of the current superblock. M has $\lfloor \frac{n}{\ell} \rfloor$ entries and can be stored in $\mathcal{O}(\frac{n}{\ell} \cdot \log \ell^2) = \mathcal{O}(\frac{n \cdot \log \log n}{\log n}) = o(n)$ bits.

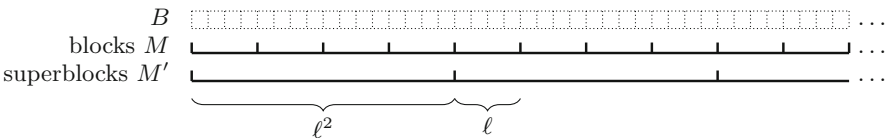


Fig. 5. 2-level dictionary. Blocks and superblocks are allocated for each character (only one shown).

Given a rank query $rank(B, i)$, one can now add the corresponding superblock and block values. But we still have to account for the 1's in the block covering position i (in case i is not at the end of a block). Let P be a precomputed lookup table such that for each possible bit vector V of length ℓ

and $i \in [1..\ell]$ holds $P[V][i] = \text{rank}(V, i)$. V has $2^\ell \cdot \ell$ entries of values at most ℓ and thus can be stored in

$$\mathcal{O}(2^\ell \cdot \ell \cdot \log \ell) = \mathcal{O}\left(2^{\frac{\log n}{2}} \cdot \log n \cdot \log \log n\right) = \mathcal{O}(\sqrt{n} \cdot \log n \cdot \log \log n) = o(n)$$

bits. We now decompose a rank query into 3 subqueries using the precomputed tables. For a position i we determine the index $p = \lfloor \frac{i-1}{\ell} \rfloor$ of next block left of i and the index $q = \lfloor \frac{p-1}{\ell} \rfloor$ of the next superblock left of block p . Then it holds:

$$\text{rank}(B, i) = M'[q] + M[p] + P[B[1 + p\ell..(p+1)\ell]][i - p\ell].$$

Since the text T of length n has to be addressed, we assume that a register has at least size $\lceil \log n \rceil$. Thus $B[1 + p\ell..(p+1)\ell]$ fits into a single register and can be determined in $\mathcal{O}(1)$ time. Therefore a rank query can be answered in $\mathcal{O}(1)$ time. In practice the precomputed lookup table P is replaced by a popcount operation on the CPU register and we have only two lookup operations.

One can now replace the occurrence table by this 2-level dictionary, i.e., by creating a bit vector for every $c \in \Sigma$ and setting it to 1 if the character occurs in the BWT L . This results in $\mathcal{O}(\sigma \cdot n) + o(\sigma \cdot n)$ bits space requirement.

References

1. Belazzougui, D., Cunial, F., Kärkkäinen, J., Mäkinen, V.: Versatile succinct representations of the bidirectional burrows-wheeler transform. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 133–144. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-40450-4_12](https://doi.org/10.1007/978-3-642-40450-4_12)
2. Belazzougui, D., Navarro, G.: Optimal lower and upper bounds for representing sequences. ACM Trans. Algorithms **11**, 31 (2015)
3. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Technical report (1994)
4. Döring, A., Weese, D., Rausch, T., Reinert, K.: SeqAn an efficient, generic C++ library for sequence analysis. BMC Bioinform. **9**, 11 (2008). <https://doi.org/10.1186/1471-2105-9-11>
5. Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: Annual Symposium on Foundations of Computer Science (2000). <https://doi.org/10.1109/SFCS.2000.892127>
6. Ferragina, P., Manzini, G., Mäkinen, V., Navarro, G.: Compressed representations of sequences and full-text indexes. ACM Trans. Algorithms (TALG) **3**, 20 (2007)
7. Gog, S., Beller, T., Moffat, A., Petri, M.: From theory to practice: plug and play with succinct data structures. In: Gudmundsson, J., Katajainen, J. (eds.) SEA 2014. LNCS, vol. 8504, pp. 326–337. Springer, Cham (2014). doi:[10.1007/978-3-319-07959-2_28](https://doi.org/10.1007/978-3-319-07959-2_28)
8. Grossi, R., Gupta, A., Vitter, J.: High-order entropy-compressed text indexes. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (2003)
9. Hauswedell, H., Singer, J., Reinert, K.: Lambda: the local aligner for massive biological data. Bioinformatics (Oxford, England) **30**, i349–i355 (2014). <https://doi.org/10.1093/bioinformatics/btu439>

10. Jacobson, G.J.: Succinct static data structures (1988)
11. Lam, T., Li, R., Tam, A., Wong, S., Wu, E.: High throughput short read alignment via bi-directional BWT. In: Proceedings of BIBM, pp. 31–36 (2009). <https://doi.org/10.1109/BIBM.2009.42>
12. Lam, T., Sung, W., Tam, S., Wong, C., Yiu, S.: Compressed indexing and local alignment of DNA. *Bioinformatics* **24**, 791–797 (2008). <https://doi.org/10.1093/bioinformatics/btn032>
13. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012)
14. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM (2013)
15. Li, H., Durbin, R.: Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009). <https://doi.org/10.1093/bioinformatics/btp324>
16. Meyer, F., Kurtz, S., Backofen, R., Will, S., Beckstette, M.: Structator: fast index-based search for RNA sequence-structure patterns. *BMC Bioinform.* **12**, 214 (2011). <https://doi.org/10.1186/1471-2105-12-214>
17. Navarro, G., Providel, E.: Fast, small, simple rank/select on bitmaps. In: International Symposium on Experimental Algorithms (2012). https://doi.org/10.1007/978-3-642-30850-5_26
18. Santiago, M., Sammeth, M., Guigo, R., Ribeca, P.: The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat. Methods* **9**, 1185–1188 (2012). <https://doi.org/10.1038/nmeth.2221>
19. Schnattinger, T., Ohlebusch, E., Gog, S.: Bidirectional search in a string with wavelet trees and bidirectional matching statistics. *Inf. Comput.* **213**, 13–22 (2012). <https://doi.org/10.1016/j.ic.2011.03.007>
20. Siragusa, E.: Approximate string matching for high-throughput sequencing. Ph.D. thesis, Freie Universität Berlin (2015)
21. Siragusa, E., Weese, D., Reinert, K.: Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic Acids Res.* **41**, e78–e78 (2013). <https://doi.org/10.1093/nar/gkt005>
22. Ye, Y., Choi, J.-H., Tang, H.: Rapsearch: a fast protein similarity search tool for short reads. *BMC Bioinform.* **12**, 1 (2011)

A Bayesian Framework for Estimating Cell Type Composition from DNA Methylation Without the Need for Methylation Reference

Elior Rahmani¹(✉), Regev Schweiger¹, Liat Shenhav¹,
Eleazar Eskin², and Eran Halperin²(✉)

¹ Tel Aviv University, Tel Aviv, Israel

elior.rahmani@gmail.com

² University of California Los Angeles, Los Angeles, CA, USA

ehalperin@cs.ucla.edu

Abstract. Genome-wide DNA methylation levels measured from a target tissue across a population have become ubiquitous over the last few years, as methylation status is suggested to hold great potential for better understanding the role of epigenetics. Different cell types are known to have different methylation profiles. Therefore, in the common scenario where methylation levels are collected from heterogeneous sources such as blood, convoluted signals are formed according to the cell type composition of the samples. Knowledge of the cell type proportions is important for statistical analysis, and it may provide novel biological insights and contribute to our understanding of disease biology. Since high resolution cell counting is costly and often logistically impractical to obtain in large studies, targeted methods that are inexpensive and practical for estimating cell proportions are needed. Although a supervised approach has been shown to provide reasonable estimates of cell proportions, this approach leverages scarce reference methylation data from sorted cells which are not available for most tissues and are not appropriate for any target population. Here, we introduce BayesCCE, a Bayesian semi-supervised method that leverages prior knowledge on the cell type composition distribution in the studied tissue. As we demonstrate, such prior information is substantially easier to obtain compared to appropriate reference methylation levels from sorted cells. Using real and simulated data, we show that our proposed method is able to construct a set of components, each corresponding to a single cell type, and together providing up to 50% improvement in correlation when compared with existing reference-free methods. We further make a design suggestion for future data collection efforts by showing that results can be further improved using cell count measurements for a small subset of individuals in the study sample or by incorporating external data of individuals with measured cell counts. Our approach provides a new opportunity to investigate cell compositions in genomic studies of tissues for which it was not possible before.

Keywords: DNA methylation · Epigenetics · Bayesian model · Cell type composition · Cell type proportions · Tissue heterogeneity

1 Introduction

Epigenome-Wide Association Studies (EWAS), where genome-wide methylation levels are measured across a population and compared to a phenotype of interest, have become ubiquitous over the last few years. Many associations between methylation sites and disease status have been reported (e.g., multiple sclerosis [1], schizophrenia [2], and type 2 diabetes [3]), suggesting an important role for DNA methylation in complex diseases. Thus, DNA methylation status holds great potential for better understanding the role of epigenetics, potentially leading to better clinical tools for diagnosing and treating patients.

In a typical EWAS, we obtain a large matrix in which each entry corresponds to the methylation level (a number between 0 and 1) at a specific genomic position for a specific individual. This level represents the fraction of the probed DNA molecules that were found to have an additional methyl group at the specific position for the specific individual. In such studies, we typically search for rows of the methylation matrix (each corresponding to one genomic position) that are significantly correlated with a phenotype of interest. The analysis of EWAS is complicated by the fact that the studied tissue is typically a mixture of different cell types. Since each cell type may have a distinct methylation pattern, the resulting DNA methylation matrix is a convolution of the signals arising from the different cell types. As a result, a large number of false discoveries may be found in the common case where the cell type composition is correlated with the phenotype [4].

In principle, one can avoid false discoveries by adding high-resolution cell counts to a regression model commonly used in an EWAS. Unfortunately, such cell counting for a large cohort may be costly and often logistically impractical (e.g., in some tissues, such as blood, reliable cell counting can be obtained from fresh samples only). In order to overcome this problem and to allow correcting methylation data for cell type composition, several statistical and computational methods have been proposed [5–9]. These methods take either a supervised approach, in which reference data of methylation patterns from sorted cells are obtained and used for predicting cell compositions [5], or an unsupervised approach (reference-free) [6–9].

The main advantage of the reference-based method is that it provides direct (absolute) estimates of the cell counts, while current unsupervised methods are only capable of inferring components that capture linear combinations of the cell counts. However, the reference-based method can only be applied when relevant reference data exist. Currently, reference data only exist for blood [10], breast [11] and brain [12], for a small number of individuals (e.g., six samples in the blood reference [10]). In addition, the individuals in most data sets do not match the reference individuals in their methylation-altering factors such as age [13] and sex [14, 15]. This problem was recently highlighted in a study showing that available blood reference collected from adults fails to estimate cell proportions of newborns [16]. It is therefore often the case that unsupervised methods are either the only option or are a better option for the analysis of EWAS.

As opposed to the supervised approach, although can be applied for any tissue in principle, the reference-free methods do not provide direct estimates of the cell type proportions. A few reference-free methods allow us to infer a set of components, or general axes, which were shown to be linearly correlated with cell type composition [8,9]. Unlike cell proportions, while linearly correlated components are useful in linear analyses such as linear regression, they cannot be used in any nonlinear downstream analysis (e.g., when studying specific cell types). Cell proportions may provide novel biological insights and contribute to our understanding of disease biology, and we therefore need targeted methods that are practical and low in cost.

Here, we propose an alternative strategy that utilizes prior knowledge about cell counts to improve upon the performance of reference-free methods, while addressing some of their limitations. We present a Bayesian semi-supervised method, BayesCCE (Bayesian Cell Count Estimation), which encodes experimentally obtained cell count information as a prior on the distribution of the cell type composition in the data. As we demonstrate here, the required prior is substantially easier to obtain compared with standard reference data from sorted cells. We can estimate this prior from general cell counts collected in previous studies, without the need for corresponding methylation data or any other genomic data.

We evaluate our method on two large data sets and on simulated data, and show that our method produces a set of components that can be used as cell composition estimates. We observe that each component of BayesCCE can be regarded as corresponding to a linear transformation of exactly one cell type (i.e. high absolute correlation with one cell type, but not necessarily good estimates in absolute terms). Considering existing reference-free methods as a baseline for estimating cell proportions, we find that BayesCCE provides improvement of up to 50% in correlation. We also consider the case where both methylation and cell count information are available for a small subset of the individuals in the sample, or for a group of individuals from external data. We show that our proposed Bayesian model can leverage such additional information, and we demonstrate that it allows us to impute missing cell counts in absolute terms. Testing this case on both real and simulated data, we find that measuring cell counts for a small group of samples (a couple of dozens) can lead to a further increase in the correlation of BayesCCE's components with the cell types composition. We therefore propose that future studies will consider measuring cell counts for at least a small number of the samples in the study, if possible, or incorporate into their analysis external data of samples with both methylation and measured cell counts from the same tissue.

2 Methods

2.1 Model

Let $O \in \mathbb{R}^{m \times n}$ be an m sites by n samples matrix of DNA methylation levels coming from heterogeneous source consisted of k cell types. For methylation

levels, we consider what is commonly referred to as beta-normalized methylation levels, which are defined for each sample in each site as the proportion of methylated probes out of the total number of probes. Put differently, $O_{ji} \in [0, 1]$ for each site j and sample i . We denote $M \in \mathbb{R}^{m \times k}$ as the cell type specific mean methylation levels for each site, and denote a row of this matrix, corresponding to the j th site, using $M_{j,\cdot}$. We denote $R \in \mathbb{R}^{n \times k}$ as the cell type proportions of the samples in the data, and denote $X \in \mathbb{R}^{n \times p}$ as a matrix of p covariates for each individual and a p -length row vector β_j as their corresponding effects in the j th site. If the measurements of O_{ji} were the true values of the methylation levels, then $O_{ji} = M_{j,\cdot} R_i^T + \beta_j X_i^T$. Due to measurement noise and other unmodeled factors, we incorporate an error term ϵ_{ji} . Thus, the full model for the observed methylation levels is

$$O_{ji} = M_{j,\cdot} R_i^T + \beta_j X_i^T + \epsilon_{ji} \quad (1)$$

$$\epsilon_{ji} \sim N(0, \sigma^2) \quad (2)$$

$$\forall i \forall h : R_{ih} \geq 0 \quad (3)$$

$$\forall i : \sum_{h=1}^k R_{ih} = 1 \quad (4)$$

$$\forall j \forall h : 0 \leq M_{jh} \leq 1 \quad (5)$$

The constraints in (3) and in (4) require the cell proportions to be positive and to sum up to one in each sample, and the constraints in (5) require the cell type specific mean levels to be in the range $[0, 1]$. We note that the above formulation of the problem is similar to the one previously suggested in the context of reference-based estimation of cell proportions from DNA methylation by Houseman et al. [5]. The reference-based method first obtains an estimate of M from reference methylation data collected from sorted cells of the cell types composing the studied tissue. Once M is known, R can be estimated by solving a standard quadratic program.

If the matrix M is not known, which is a reference-free version of the problem, the above formulation of the problem can be regarded as a version of non-negative matrix factorization (NNMF) problem. NNMF has been suggested in several applications in biology; notably, the problem of inference of cell type composition from methylation data has been recently formulated as an NNMF problem [9]. In order to optimize the model, the authors use an alternative optimization procedure in which M or R are optimized while the other is kept fixed. However, as demonstrated by the authors [9], their version of NNMF results in the inference of a linear combination of the cell proportions R . Put differently, more than one component of the NNMF is required for explaining each cell type in the data. Another recent reference-free method for estimating cell composition in methylation data, ReFACTor [8], performs a feature selection step followed by a principal components analysis (PCA). Similarly as in the NNMF solution, ReFACTor is an unsupervised method and it only finds principal components (PCs) that form linear combinations of the cell proportions rather than directly estimates the cell proportion values [8].

Here, we suggest a more detailed model by adding a prior on R and taking into account potential covariates. Specifically, we assume that

$$R_i^T \sim \text{Dirichlet}(\alpha_1, \dots, \alpha_k) \quad (6)$$

where $\alpha_1, \dots, \alpha_k$ are assumed to be known. In practice, the parameters are estimated from external data in which cell type proportions of the studied tissue are known. Such experimentally obtained cell type proportions were used to test the appropriateness of the Dirichlet prior in describing cell composition distribution (data not shown). We are interested in estimating R . Deriving a maximum likelihood-based solution for this model and repeating the constraints for completeness results in the following optimization problem:

$$\hat{R}, \hat{M}, \{\hat{\beta}_j\}_{j=1}^m = \underset{R, M, \{\beta_j\}_{j=1}^m}{\operatorname{argmin}} \frac{1}{2\sigma^2} \sum_{j=1}^m \sum_{i=1}^n (O_{ji} - M_j \cdot R_i^T - \beta_j X_i^T)^2 - \sum_{i=1}^n \sum_{h=1}^k (\alpha_h - 1) \log(R_{ih}) \quad (7)$$

$$\text{s.t } \forall i \forall h : R_{ih} \geq 0 \quad (8)$$

$$\forall i : \sum_{h=1}^k R_{ih} = 1 \quad (9)$$

$$\forall j \forall h : 0 \leq M_{jh} \leq 1 \quad (10)$$

Our intuition in this model is that since the priors on R are estimated from real data, incorporating them will push the solution of the optimization to return estimates of R which are closer to the true values as opposed to a linear combination of them.

2.2 Algorithm

Our algorithm uses ReFACTor as a starting point. Specifically, we use ReFACTor's PCs (ReFACTor components) in order to estimate R by finding an appropriate linear transformation of the ReFACTor components. In principle, both ReFACTor and NNMF could be used as the starting point for our method. However we found that ReFACTor captures a larger portion of the cell composition variance compared with the NNMF solution (see Results).

Applying ReFACTor on our input matrix O we get a list of t sites that are most informative with respect to the cell composition in O . Let $\tilde{O} \in \mathbb{R}^{t \times n}$ be a truncated version of O containing only the t sites selected by ReFACTor. We apply PCA on \tilde{O} to get $L \in \mathbb{R}^{t \times d}$, $P \in \mathbb{R}^{n \times d}$, the loadings and scores of the first d ReFACTor components. Then, we reformulate the original optimization problem in terms of linear transformations of L and P as follows:

$$\hat{A}, \hat{V}, \hat{B} = \underset{A, V, B}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\tilde{O} - LAV^T P^T - LBX^T\|_F^2 - \sum_{i=1}^n \sum_{h=1}^k (\alpha_h - 1) \log \left(\sum_{l=1}^d P_{il} V_{lh} \right) \quad (11)$$

$$\text{s.t. } \forall i \forall h : \sum_{l=1}^d P_{il} V_{lh} \geq 0 \quad (12)$$

$$\forall i : \sum_{h=1}^k \sum_{l=1}^d P_{il} V_{lh} = 1 \quad (13)$$

$$\forall j \forall k : 0 \leq \sum_{l=1}^d L_{jl} A_{lh} \leq 1 \quad (14)$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm, $A \in \mathbb{R}^{d \times k}$ is a transformation matrix such that $\tilde{M} = LA$ (\tilde{M} being a truncated version of M with the t sites selected by ReFACTor), $V \in \mathbb{R}^{d \times k}$ is a transformation matrix such that $R = PV$ and $B \in \mathbb{R}^{d \times p}$ is a transformation matrix such that LB corresponds to the effects of each covariate on the methylation levels in each site. The constraints in (12) and in (13) correspond to the constraints in (8) and in (9), and the constraints in (14) correspond to the constraints in (10).

Given \hat{V} , we simply return $\hat{R} = P\hat{V}$ as the estimated cell proportions. Note that in the new formulation we are now required to learn only $d(2k + p)$ parameters - d, k and p being small constants - a dramatically decreased number of parameters compared with the original problem which requires $nk + m(k + p)$ parameters. By taking that approach, we make an assumption that \tilde{O} consists of a low rank structure that captures the cell composition using d orthogonal vectors. While a natural value for d would be $d = k$, d is not bounded to be k . Particularly, in cases where substantial additional cell composition signal is expected to be captured by latter ReFACTor components (i.e. components beyond the first k), we would expect to benefit from increasing d . Clearly, overly increasing d is expected to result in overfitting and thus a decrease in performance. Finally, taking into account covariates with potentially dominant effects in the data should alleviate the risk of introducing noise into \hat{R} in case of mixed low rank structure of cell composition signal and other unwanted variation in the data. We note, however, that similarly to the case of correlated explaining variables in regression, considering covariates that are expected to be correlated with the cell type composition may result in underestimation of A, V and therefore to a decrease in the quality of \hat{R} .

2.3 Imputing Cell Counts Using a Subset of Samples with Measured Cell Counts

In practice, we observe that each of BayesCCE's components corresponds to a linear transformation of one cell type rather than to an estimate of that cell

type in absolute terms. That is, it still lacks the right scaling (multiplication by a constant and addition of a constant) for transforming it into cell proportions. Furthermore, we would like the i th BayesCCE component to correspond to the i th cell type described by the prior using the α_i parameter. Empirically, that is not necessarily the case, especially in scenarios where some of the α_i values are similar. In order to address these two caveats, we suggest incorporating measured cell counts for a subset of the samples in the data.

Assume we have n_0 reference samples in the data with known cell counts $R^{(0)}$ and n_1 samples with unknown cell counts $R^{(1)}$ ($n = n_0 + n_1$). This problem can be regarded as an imputation problem, in which we aim at imputing cell counts for samples with unknown cell counts. We can find \hat{M} by solving the problem in (7) under the constraints in (10) for the n_0 reference samples while replacing R with $R^{(0)}$ and keeping it fixed. Then, given \hat{M} , we can now solve the problem in (11), after replacing LA with \hat{M} (i.e. we find only V, B now), under the following constraints

$$\forall(1 \leq i \leq n_0) \forall h : \sum_{l=1}^d P_{il}^{(0)} V_{lh} = R_{ih}^{(0)} \quad (15)$$

$$\forall(1 \leq i \leq n_1) \forall h : \sum_{l=1}^d P_{il}^{(1)} V_{lh} \geq 0 \quad (16)$$

$$\forall(1 \leq i \leq n_1) : \sum_{h=1}^k \sum_{l=1}^d P_{il}^{(1)} V_{lh} = 1 \quad (17)$$

where $P^{(0)}$ contains n_0 rows corresponding to the reference samples in P , and $P^{(1)}$ contains n_1 rows corresponding to the remaining samples in P . In this case, both problems of estimating M and solving (11) while keeping \hat{M} fixed are convex - the first problem takes the form of a standard quadratic problem and the latter results in an optimization problem of the sum of two convex terms under linear constraints. Using \hat{M} , estimated from cell counts and corresponding methylation levels of a group of samples, as well as adding the constraints in (15), are expected to direct the inference of R towards a set of components such that each one corresponds to one known cell type with a proper scale.

2.4 Implementation and Practical Issues

We estimate σ^2 in (11) as the mean squared error of predicting \tilde{O} with P and X . The $\alpha_1, \dots, \alpha_k$ Dirichlet parameters of the prior can be estimated from cell proportions using maximum likelihood estimators. In practice, we add a column of ones to both L and P in (11) in order to assure feasibility of the problem - these constant columns are used to compose the mean methylation level per site across all cell types and the mean cell proportion fraction in each cell type across all samples. In addition, we slightly relax some of the constraints in the problem to avoid problems due to numeric instability and inconsistent noise issues. First, we do not impose the equality constraints in (13) and in (17) but rather allow a

small deviation from equality (5%). In addition, the inequality constraints in (12) and in (16) are changed to require the cell proportions to be greater than $\epsilon > 0$, as a result of the logarithm term in the objective ($\epsilon = 0.0001$). Finally, given cell counts for a subset of the samples, we allow a small deviation from the equality constraints in (15) due to expected inaccuracies of cell counts measurements (1%).

We performed all the experiments in this paper using a Matlab implementation of BayesCCE. Specifically, we solved the optimization problems in BayesCCE using the *fmincon* function with the default interior-point algorithm, and we used the *fastfit* [17] Matlab package for calculating maximum likelihood estimates of the Dirichlet priors. All executions of BayesCCE required several minutes on a 64-bit Mac OS X computer with 3.1 GHz and 16 GB of RAM. Corresponding code is available at: <https://github.com/cozygene/bayescce>.

2.5 Evaluation of Performance

The fraction of cell composition variation (R^2) captured by each of the reference-free methods, ReFACTor and NMF, was computed for each cell type using a linear predictor fitted with the first k components provided by the method. In order to evaluate the performance of BayesCCE, for each component i we calculated its correlation with the i th cell type, and reported the mean absolute correlation (MAC) across the k estimated cell types. Empirically, we observed that in the case of $k = 6$ with no known cell counts for a subset of samples, the i th BayesCCE component did not necessarily correspond to the i th cell type. Put differently, the labels of the k cell types had to be permuted before calculating the MAC. In this case we considered the permutation of the labels which resulted with the highest MAC as the correct permutation. In the rest of the cases, we did not apply such permutation (all the experiments using $k = 3$ and all the experiments using $k = 6$ with known cell counts for a subset of the samples). For evaluating ReFACTor and NMF, reference-free methods which do not attribute their components to specific cell types in any scenario, we considered the permutation leading to the highest MAC in all experiments when compared with BayesCCE. In addition, we considered the mean absolute error (MAE) as an additional quality measurement. When calculating absolute errors for the ReFACTor components, we scaled each component to be in the range $[0, 1]$. Finally, in experiments where cell counts were assumed to be known for a subset of the samples, MAC and MAE were calculated using only the samples for which cell counts were assumed to be unknown.

2.6 Implementation of ReFACTor and NMF

The ReFACTor components were calculated for each data set using the parameters $k = 6$ and $t = 500$ and according to the implementation of ReFACTor described at <http://glint-epigenetics.readthedocs.io>, while accounting for known covariates in each data set. More specifically, in the Liu et al. data [18] we accounted for age, sex, smoking status and batch information, and in the

Hannum et al. data [19] we accounted for age, sex, ethnicity and batch information. We used the first six ReFACTor components ($d = 6$) for simulated data in order to accommodate with the number of simulated cell types, and the first ten components ($d = 10$) for real data, as real data are typically more complex and are therefore more likely to contain substantial signal in latter components. The NNMF components were computed for each data set using the RefFreeE-WAS R package from the subset of 10,000 most variable sites in the data set, as performed in the NNMF paper by the authors [9].

2.7 Implementation of the Reference-Based Algorithm

We implemented the reference-based algorithm according to Houseman et al. [5], using 300 highly informative methylation sites defined in a recent study [20] and using reference data collected from sorted blood cells [10].

2.8 Data Sets

We evaluated the performance of BayesCCE using three data sets collected with the Illumina 450K DNA methylation array. All three data sets are publicly available and preprocessed versions of the data were downloaded from the Gene Expression Omnibus (GEO) database. The first data set (accession GSE42861) was studied in a recent association study of DNA methylation with rheumatoid arthritis (RA) by Liu et al. ($n = 686$) [18]. The second data set (accession GSE40279) was originally used in a study of aging rates by Hannum et al. ($n = 656$) [19]. In addition, we used a reference data set of sorted cell types collected in six individuals from whole blood tissue (accession GSE35069) [10]. The latter was used for generating simulated data sets and for estimating the cell type specific mean levels in the implementation of the reference-based algorithm. We excluded from each data set sites coming from the sex chromosomes, as well as polymorphic and cross-reactive sites, as was previously reported [21]. Two samples in the Hannum et al. data were detected as outliers by PCA and were therefore excluded. When running BayesCCE on the data sets by Liu et al. and Hannum et al. we considered known batch information in the analysis.

2.9 Data Simulation

We simulated data following a model that was previously described in details elsewhere [8]. Briefly, we used methylation levels from sorted blood cells [10] and, assuming normality, estimated maximum likelihood parameters for each site in each cell type. Cell type specific DNA methylation data were then generated for each simulated individual from normal distributions with the estimated parameters, conditional on the range $[0,1]$, for six cell types and for each site. Cell proportions for each individual were generated using a Dirichlet distribution. The parameters for the Dirichlet were fitted using the cell proportions estimated for the individuals in the Liu et al. [18] and Hannum et al. [19] data sets using

the reference-based method [5]. Finally, observed DNA methylation levels were composed from the cell type specific methylation levels and cell proportions for each individual, and a random normal noise was added to every data entry to simulate technical noise ($\sigma = 0.01$).

3 Results

3.1 Benchmarking Existing Reference-Free Methods

We first demonstrate that existing reference-free methods can estimate components that are correlated with the tissue composition in methylation data collected from heterogeneous sources. For the experiments in this paper, we used the whole-blood data set by Liu et al. [18] ($n = 686$) and the whole-blood data set by Hannum et al. [19] ($n = 654$; see Methods). In addition, we simulated data based on reference data set of methylation levels from sorted leukocytes cells [10] (see Methods). While cell proportions were known for each sample in the simulated data, cell counts were not available for the two real data sets. We therefore estimated the cell type composition of six major blood cell types (granulocytes, monocytes and four subtypes of lymphocytes) using a reference-based method [5], which was shown to reasonably estimate leukocyte cell proportions from whole blood methylation data collected from adult individuals [16, 20, 22]. Due to the absence of large publicly available data with measured cell counts, these estimates were considered as the ground truth for evaluating the performance of the different methods.

We considered two reference-free methods, ReFACTor [8] and NNMF [9], both allowing to generate components that were shown to capture cell type composition information in methylation. We evaluated the first six components of ReFACTor and the six components provided by NNMF - six being the number of estimated cell types composing the ground truth. We found both methods to capture a large portion of the cell composition in all data sets; particularly, we observed that ReFACTor performed considerably better than NNMF in all data sets (Fig. 1a–c). Yet, in spite of the fact that both ReFACTor and NNMF capture a large portion of the cell composition variance, each component provided by these methods is a linear combination of the cell types in the data rather than an estimate of the proportions of a single cell type. As a result, as we show in the following experiments, both methods, in general, perform poorly when their components are considered as estimates of cell proportions.

3.2 Evaluation of BayesCCE on Real and Simulated Data

We evaluated BayesCCE under various scenarios. The results of the experiments described hereafter are summarize in Fig. 1d–h. In the first and most common scenario, we assume that no appropriate reference methylation data of sorted cells exist for the studied tissue, but we do have information about the distribution of the cell composition in the studied tissue. Such information can be

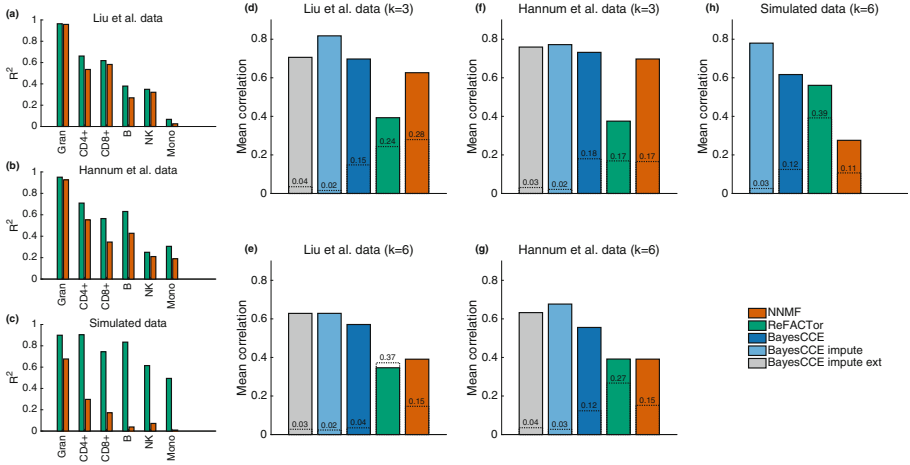


Fig. 1. Capturing cell type composition in real and simulated data. (a)–(c) The fraction of variance explained (R^2) by the first six components of NNMF and ReFACToR in each one of the six cell types in the Liu et al. data ($n = 686$), the Hannum et al. data ($n = 654$) and in simulated data ($n = 650$; average over 10 simulations). (d)–(h) Considering a single component for estimating each cell type in real data and in simulated data (average over 10 simulations), using existing reference-free methods (NNMF and ReFACToR) and BayesCCE, as well as using BayesCCE with known cell counts for 5% of the samples in the data (BayesCCE impute) and BayesCCE with cell counts and methylation for a group of samples from external data (BayesCCE impute ext). The bar plots present the mean absolute correlation of the components with the cell types in case of three assumed cell types ($k = 3$) and in case of six assumed cell types ($k = 6$). Dashed line on each bar plot indicates the mean absolute error of the estimates across all cell types. For more details about the evaluation of performance see Methods.

inferred from cell counts collected in previous studies of the same tissue (without the need for any additional genomic data). This information can be then used by BayesCCE for tuning the prior required for the model (see Methods). In order to demonstrate this, we used cell counts collected from 35 healthy adults in a recent study [23]. These cell counts measured levels of lymphocytes, monocytes and three subtypes of granulocytes. Since our ground truth, compiled using the reference-based method, contained only the total granulocyte levels, we collapsed the three subtypes of granulocytes into a total measurement of granulocytes.

We applied BayesCCE on the real data sets under the assumption that three cell types compose the data ($k = 3$). Since each component of BayesCCE is expected to correspond to a linear transformation of one cell type, we report absolute linear correlations (see Methods). BayesCCE provided excellent estimates of the levels of granulocytes and lymphocytes in both data sets ($r = 0.96$ and $r = 0.98$ in the Liu et al. data, and $r = 0.94$ and $r = 0.98$ in the Hannum et al. data; see Fig. 2). In contrast, we observed poor estimates of the monocyte

levels ($r = 0.14$ in the Liu et al. data and $r = 0.26$ in the Hannum et al. data). We note that poor performance in capturing some cell type may be partially derived by inaccuracies introduced by the reference-based estimates which are used as the ground truth in our experiments. For example, several recent studies consisted of samples for which both methylation levels and cell count measurements were available, demonstrated that while the reference-based estimates of the overall lymphocyte and granulocyte levels were found to be highly correlated with the true levels, the accuracy of the estimates of monocytes was found to be substantially lower [8, 16, 24]. Such inaccuracies in estimating a specific cell type by the reference-based approach may be the result of utilizing inappropriate reference. More specifically, cell types with highly variable methylation patterns across different populations may not be well represented for the target population by existing reference (coming from a specific population). Another possible driver for low quality estimates is having cell types with methylation profiles that do not distinct them well enough from other cell types in the tissue, or failing to select a set of informative features that mark some of the cell types.

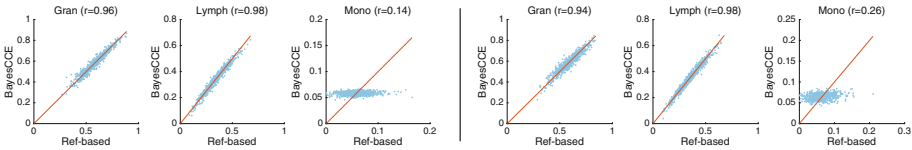


Fig. 2. BayesCCE captures cell type composition under the assumption of three cell types in the data ($k = 3$): granulocytes, lymphocytes and monocytes. Each BayesCCE component was linearly transformed to match its corresponding cell type in scale. Left: the results for the Liu et al. data. Right: the results for the Hannum et al. data.

As a second validation of our method, we used the reference-based estimates of the six cell types for learning the prior. For each one of the two real data sets, we used the cell proportion estimates of the other data set for learning the prior. We then applied BayesCCE on each data set under the assumption of six cell types ($k = 6$) and measured the correlation with the reference-based estimates. The mean absolute correlation across the six cell types was found to be 0.57 in the Liu et al. data and 0.56 in the Hannum et al. data (Fig. 3). In addition to the real data analysis, we further conducted a similar experiment on simulated data ($n = 650$). In this case, we estimated the prior from a group of 50 samples that were generated from the true distribution. We applied BayesCCE on ten different simulated data sets, and found the mean absolute correlation across all cell types and across all the simulated data sets to be 0.62. As expected, applying BayesCCE on increased sample size resulted in an improved performance (Supplementary Fig. S1).

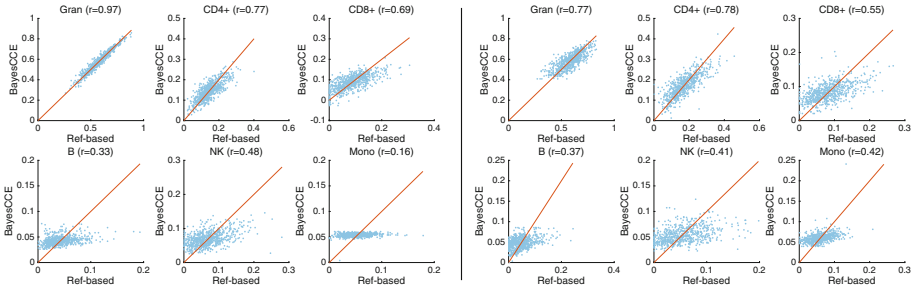


Fig. 3. BayesCCE captures cell type composition under the assumption of six cell types in the data ($k = 6$): granulocytes, four subtypes of lymphocytes (CD4+, CD8+, B cells and NK cells) and monocytes. Each BayesCCE component was linearly transformed to match its corresponding cell type in scale. Left: the results for the Liu et al. data. Right: the results for the Hannum et al. data.

3.3 Evaluation of Cell Count Imputation

Next, we considered the scenario in which cell counts are known for a small subset of the samples in the data. This problem can be viewed as an imputation problem of the missing cell count values (see Methods). We repeated the previous experiments ($k = 3$ and $k = 6$), only this time we used the values of the estimated cell counts for randomly selected 5% of the samples in each data set. As opposed with the previous experiments, in which each one of BayesCCE’s components formed a linear transformation of one of the cell types, here we get that the BayesCCE components form absolute estimates of the cell proportions (i.e. low absolute error). In addition, we observed up to 22% improvement in the mean correlation values compared with our previous experiments (Supplementary Figs. S2 and S3). We further tested this approach on simulated data ($n = 650$), while assuming known cell counts for 5% of the samples in the data, and found the mean correlation across different cell types and across ten different simulated data set to be 0.78. Applying this approach with an increased number of samples for which cell counts are known, reveals that the cell count estimates can be improved using a relatively small subset of a couple of dozens of samples with known cell counts (Supplementary Fig. S4).

In the absence of cell counts for a subset of the individuals in the data, external data with samples for which both methylation levels and cell counts are available can be added to the analysis. Again, we repeated the previous experiments ($k = 3$ and $k = 6$), only this time for each data set we added randomly selected 5% of the samples from the other data set, and used both their methylation levels and estimated cell counts in the analysis. Unlike in the previous experiments, here we potentially introduce new batch effects into the analysis, as in each experiment the original sample is combined with external data. We therefore accounted for the new batch information by adding it as a new covariate into BayesCCE. We observed up to 14% improvement in the mean correlation values compared with our previous experiments not taking any cell

counts into account (Supplementary Figs. S5 and S6), showing that incorporating external samples with both methylation and cell counts can be a practical and useful way for estimating cell counts.

4 Discussion

We introduce BayesCCE, a Bayesian method that estimates cell type composition from heterogeneous methylation data using a prior on the cell composition distribution. In contrast to previous methods, using BayesCCE we can generate components such that each component corresponds to a linear transformation of a single cell type. These components can allow researchers to perform downstream analysis that is not possible using existing reference-free methods.

Our approach is based on finding a suitable linear transformation of the components found by ReFACTor [8]. Thus, it is limited by the quality of the ReFACTor components, and particularly BayesCCE will provide the exact same result as ReFACTor if used for correcting for potential cell type composition confounder in methylation data. We therefore suggest to use ReFACTor for correction and BayesCCE for cases in which a study of individual cell types is performed. We note that several supervised and unsupervised deconvolution methods have been suggested for estimating cell composition from gene expression [25–29]. However, these were refined for gene expression data and, to the best of our knowledge, none of these methods takes into account prior knowledge about the cell composition distribution as in BayesCCE. It remains of interest to investigate whether BayesCCE can be adapted for estimating cell composition from gene expression without the need for purified expression profiles.

The parameters of the prior required for BayesCCE can be estimated by utilizing previous studies that collected cell counts from the tissue of interest. Since no other genomic information is required, obtaining such data is relatively easy for many tissues, such as brain [30], heart [31] and adipose tissue [32]. Particularly, such data should be substantially easier to obtain compared to reference data from sorted cells for the corresponding tissues. Ideally, one would want to use cell counts coming from the same population as the target population in the study, especially when the cell composition distribution of the studied tissue may vary substantially across different populations. While this may be a potential limitation of BayesCCE in cases where cell counts from the target population are not available, our results using priors estimated from three different data sets empirically show that priors estimated from a different population than the target population can still provide good estimates.

Since no large data with measured cell counts are currently publicly available, we used a supervised method [5] for obtaining cell type proportion estimates, which were used as the ground truth in our experiments. Even though the method used for obtaining these estimates was shown to reasonably estimate leukocyte cell proportions from whole blood methylation data in several independent studies [16, 20, 22], these estimates may have introduced biases into the analysis. Particularly, in the presence of systematic biases, the estimates could

have affected the estimated priors, which in turn could have affected the results. However, we believe that our results on several independent data sets, including simulated data, and the use of priors estimated from several sources, including real cell counts, provide a compelling evidence for the utility of BayesCCE.

Finally, we demonstrate that imputation of the cell counts can be highly accurate even when cell counts are available for only a relatively small number of individuals. Moreover, in the general setting of BayesCCE, each component is correlated to one cell type, and the identity of that cell type may not be known, while in the case of imputation BayesCCE is able to reconstruct the cell counts up to a small absolute error (i.e. each component corresponds to a known cell type and is scaled to form cell proportion estimates of that cell type). We therefore recommend that in future studies either the cell counts be measured for at least a couple of dozens of the samples or external data of samples with measured cell counts be utilized in the analysis.

Acknowledgments. We would like to thank Lana Martin for feedback on the manuscript. This research was partially supported by the Edmond J. Safra Center for Bioinformatics at Tel Aviv University. E.H., E.R., L.S. and R.S. were supported in part by the Israel Science Foundation (Grant 1425/13), E.H., L.S. and R.S. by the United States Israel Binational Science Foundation grant 2012304. E.R. and L.S. were supported by Len Blavatnik and the Blavatnik Research Foundation. R.S. was supported by the Colton Family Foundation. E.E. was supported by National Science Foundation grants 1065276, 1302448, 1320589 and 1331176, and National Institutes of Health grants R01-GM083198, R01-ES021801, R01-MH101782, R01-ES022282 and U54EB020403.

Appendix

The supplementary materials can be found at: <https://github.com/cozygene/BayesCCE/blob/master/BayesCCE-SI.pdf>.

References

1. Koch, M.W., Metz, L.M., Kovalchuk, O.: Epigenetic changes in patients with multiple sclerosis. *Nat. Rev. Neurol.* **9**(1), 35–43 (2013)
2. Ikegami, T., Bundo, M., Sunaga, F., Asai, T., Nishimura, F., Yoshikawa, A., Kawamura, Y., Hibino, H., Tochigi, M., Kakiuchi, C., et al.: DNA methylation analysis of BDNF gene promoters in peripheral blood cells of schizophrenia patients. *Neurosci. Res.* **77**(4), 208–214 (2013)
3. Toperoff, G., Aran, D., Kark, J.D., Rosenberg, M., Dubnikov, T., Nissan, B., Wainstein, J., Friedlander, Y., Levy-Lahad, E., Glaser, B., et al.: Genome-wide survey reveals predisposing diabetes type 2-related DNA methylation variations in human peripheral blood. *Hum. Mol. Genet.* **21**(2), 371–383 (2012)
4. Jaffe, A.E., Irizarry, R.A.: Accounting for cellular heterogeneity is critical in epigenome-wide association studies. *Genome Biol.* **15**(2), R31 (2014)

5. Houseman, E.A., Accomando, W.P., Koestler, D.C., Christensen, B.C., Marsit, C.J., Nelson, H.H., Wiencke, J.K., Kelsey, K.T.: DNA methylation arrays as surrogate measures of cell mixture distribution. *BMC Bioinform.* **13**(1), 86 (2012)
6. Houseman, E.A., Molitor, J., Marsit, C.J.: Reference-free cell mixture adjustments in analysis of DNA methylation data. *Bioinformatics* **30**(10), 1431–1439 (2014)
7. Zou, J., Lippert, C., Heckerman, D., Aryee, M., Listgarten, J.: Epigenome-wide association studies without the need for cell-type composition. *Nat. Methods* **11**(3), 309–311 (2014)
8. Rahmani, E., Zaitlen, N., Baran, Y., Eng, C., Hu, D., Galanter, J., Oh, S., Burchard, E.G., Eskin, E., Zou, J., et al.: Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies. *Nat. Methods* **13**(5), 443–445 (2016)
9. Houseman, E.A., Kile, M.L., Christiani, D.C., Ince, T.A., Kelsey, K.T., Marsit, C.J.: Reference-free deconvolution of DNA methylation data and mediation by cell composition effects. *BMC Bioinform.* **17**(1), 259 (2016)
10. Reinius, L.E., Acevedo, N., Joerink, M., Pershagen, G., Dahlén, S.E., Greco, D., Söderhäll, C., Scheynius, A., Kere, J.: Differential DNA methylation in purified human blood cells: implications for cell lineage and studies on disease susceptibility. *PLoS ONE* **7**(7), e41361 (2012)
11. Teschendorff, A.E., Gao, Y., Jones, A., Ruebner, M., Beckmann, M.W., Wachter, D.L., Fasching, P.A., Widschwendter, M.: DNA methylation outliers in normal breast tissue identify field defects that are enriched in cancer. *Nat. Commun.* **7**, 10478 (2016)
12. Quintivano, J., Aryee, M.J., Kaminsky, Z.A.: A cell epigenotype specific model for the correction of brain cellular heterogeneity bias and its application to age, brain region and major depression. *Epigenetics* **8**(3), 290–302 (2013)
13. Horvath, S.: DNA methylation age of human tissues and cell types. *Genome Biol.* **14**(10), R115 (2013)
14. Singmann, P., Shem-Tov, D., Wahl, S., Grallert, H., Fiorito, G., Shin, S.Y., Schramm, K., Wolf, P., Kunze, S., Baran, Y., et al.: Characterization of whole-genome autosomal differences of DNA methylation between men and women. *Epigenet. Chromatin* **8**(1), 1–13 (2015)
15. Yousefi, P., Huen, K., Davé, V., Barcellos, L., Eskenazi, B., Holland, N.: Sex differences in DNA methylation assessed by 450 K BeadChip in newborns. *BMC Genomics* **16**(1), 1 (2015)
16. Yousefi, P., Huen, K., Quach, H., Motwani, G., Hubbard, A., Eskenazi, B., Holland, N.: Estimation of blood cellular heterogeneity in newborns and children for epigenome-wide association studies. *Environ. Mol. Mutagen.* **56**(9), 751–758 (2015)
17. Minka, T.: Estimating a Dirichlet distribution (2000)
18. Liu, Y., Aryee, M.J., Padyukov, L., Fallin, M.D., Hesselberg, E., Runarsson, A., Reinius, L., Acevedo, N., Taub, M., Ronninger, M., et al.: Epigenome-wide association data implicate DNA methylation as an intermediary of genetic risk in Rheumatoid Arthritis. *Nat. Biotechnol.* **31**(2), 142–147 (2013)
19. Hannum, G., Guinney, J., Zhao, L., Zhang, L., Hughes, G., Sada, S., Klotzle, B., Bibikova, M., Fan, J.B., Gao, Y., et al.: Genome-wide methylation profiles reveal quantitative views of human aging rates. *Mol. Cell* **49**(2), 359–367 (2013)
20. Koestler, D.C., Jones, M.J., Usset, J., Christensen, B.C., Butler, R.A., Kobor, M.S., Wiencke, J.K., Kelsey, K.T.: Improving cell mixture deconvolution by identifying optimal DNA methylation libraries (IDOL). *BMC Bioinform.* **17**(1), 1 (2016)

21. Chen, Y.A., Lemire, M., Choufani, S., Butcher, D.T., Grafodatskaya, D., Zanke, B.W., Gallinger, S., Hudson, T.J., Weksberg, R.: Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray. *Epigenetics* **8**(2), 203–209 (2013)
22. Koestler, D.C., Christensen, B.C., Karagas, M.R., Marsit, C.J., Langevin, S.M., Kelsey, K.T., Wiencke, J.K., Houseman, E.A.: Blood-based profiles of DNA methylation predict the underlying distribution of cell types: a validation analysis. *Epigenetics* **8**(8), 816–826 (2013)
23. Chomczynski, P., Wilfinger, W.W., Eghbalnia, H.R., Kennedy, A., Rymaszewski, M., Mackey, K.: Inter-individual differences in RNA levels in human peripheral blood. *PLoS ONE* **11**(2), e0148260 (2016)
24. Cardenas, A., Allard, C., Doyon, M., Houseman, E.A., Bakulski, K.M., Perron, P., Bouchard, L., Hivert, M.F.: Validation of a DNA methylation reference panel for the estimation of nucleated cells types in cord blood. *Epigenetics* **11**, 773–779 (2016)
25. Lu, P., Nakorchevskiy, A., Marcotte, E.M.: Expression deconvolution: a reinterpretation of DNA microarray data reveals dynamic changes in cell populations. *Proc. Natl. Acad. Sci.* **100**(18), 10370–10375 (2003)
26. Abbas, A.R., Wolslegel, K., Seshasayee, D., Modrusan, Z., Clark, H.F.: Deconvolution of blood microarray data identifies cellular activation patterns in systemic lupus erythematosus. *PLoS ONE* **4**(7), e6098 (2009)
27. Kuhn, A., Thu, D., Waldvogel, H.J., Faull, R.L., Luthi-Carter, R.: Population-specific expression analysis (PSEA) reveals molecular changes in diseased brain. *Nat. Methods* **8**(11), 945–947 (2011)
28. Zuckerman, N.S., Noam, Y., Goldsmith, A.J., Lee, P.P.: A self-directed method for cell-type identification and separation of gene expression microarrays. *PLoS Comput. Biol.* **9**(8), e1003189 (2013)
29. Steuerman, Y., Gat-Viks, I.: Exploiting gene-expression deconvolution to probe the genetics of the immune system. *PLoS Comput. Biol.* **12**(4), e1004856 (2016)
30. Azevedo, F.A., Andrade-Moraes, C.H., Curado, M.R., Oliveira-Pinto, A.V., Guimarães, D.M., Szczupak, D., Gomes, B.V., Alho, A.T., Polichiso, L., Tampellini, E., et al.: Automatic isotropic fractionation for large-scale quantitative cell analysis of nervous tissue. *J. Neurosci. Methods* **212**(1), 72–78 (2013)
31. Pinto, A.R., Ilinykh, A., Ivey, M.J., Kuwabara, J.T., D’Antoni, M.L., Debuque, R., Chandran, A., Wang, L., Arora, K., Rosenthal, N.A., et al.: Revisiting cardiac cellular composition. *Circ. Res.* **118**(3), 400–409 (2016)
32. Divoux, A., Tordjman, J., Lacasa, D., Veyrie, N., Hugol, D., Aissat, A., Basdevant, A., Guerre-Millo, M., Poitou, C., Zucker, J.D., et al.: Fibrosis in human adipose tissue: composition, distribution, and link with lipid metabolism and fat mass loss. *Diabetes* **59**(11), 2817–2825 (2010)

Towards Recovering Allele-Specific Cancer Genome Graphs

Ashok Rajaraman and Jian Ma^(✉)

Computational Biology Department, School of Computer Science,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
jianma@cs.cmu.edu

Abstract. Integrated analysis of structural variants (SVs) and copy number alterations (CNAs) in aneuploid cancer genomes is key to understanding the tumor genome complexity. A recently developed new algorithm Weaver can estimate, for the first time, allele-specific copy number of SVs and their interconnectivity in aneuploid cancer genomes. However, one major limitation is that not all SVs identified by Weaver are phased. In this paper, we develop a general convex programming framework that predicts the interconnectivity of unphased SVs with possibly noisy allele-specific copy number estimations as input. We demonstrated through applications to both simulated data and the HeLa whole-genome sequencing data that our method is robust to the noise in the input copy numbers and can predict SV phasings with high specificity. We found that our method can make consistent predictions with Weaver even if a large proportion of the input variants are unphased. We also applied our method to TCGA ovarian cancer whole-genome sequencing samples to phase unphased SVs obtained by Weaver. Our work provides an important new algorithmic framework for recovering more complete allele-specific cancer genome graphs.

1 Introduction

A significant proportion of cancer genomes are aneuploid and have undergone somatic copy number alterations (CNAs) and even whole-genome duplications (WGD) [2, 7, 20]. Structural variations (SVs) that involve complex somatic rearrangements can further modify aneuploid cancer genomes. It has been shown that aneuploid cancer genomes typically have a higher rate of CNAs and SVs that happen together [20]. Therefore, it is important to analyze CNAs and SVs in an integrated manner in aneuploid cancer genomes in order to obtain a more complete view of the tumor genome complexity, which would in turn help understand the somatic evolutionary history of cancer genomes [8]. In the past few years, many computational tools have been developed to infer SVs and CNAs individually [3, 18, 19], but none gives us a completely integrated view of how CNAs and SVs interact, nor do they provide an allele-specific context to SVs.

We previously developed a new algorithm Weaver [13], which can simultaneously analyze SVs and allele-specific copy number of the genome (ASCNG) in

the context of aneuploid cancer genome. Specifically, Weaver is able to identify allele-specific copy number of SVs (ASCNS) as well as the inter-connectivity of them (i.e., phasing). Weaver uses a Markov Random Field (MRF), where ASCNS and SV phasing configuration, together with ASCNG, are hidden states in the nodes in the graph and the observations include sequencing coverage and read linkage between SVs. The results from [13] demonstrated that Weaver can be successfully applied to cancer cell lines (MCF-7 and HeLa) as well as TCGA patient samples to generate base-pair resolution ASCNS and ASCNG with high accuracy.

However, one major limitation of Weaver is that it is not guaranteed to output a phasing for all SVs. In some tumor samples we have tested, as few as 60% of the detected SVs may be phased by using the paired-end reads from the whole-genome sequencing sample together with the known SNP phasing information from the 1000 Genomes Project. It is therefore important to develop additional approaches as a step further to predict the interconnectivity of allele-specific SVs and phase them into a more complete haplotype structure. Our motivation in this work is that we may be able to utilize the copy number information gathered to further predict phasing for the remaining allele-specific breakpoints caused by SVs. Such an approach could serve two useful purposes: (1) the predicted phasing structure would provide a more complete context for interpreting cancer-specific functional genomic data (such as [1]); and (2) the predicted phasing structure may offer insights into incorporating data from other technologies (such as physical maps [9], PacBio [1, 6], or 10X Genomics [22]), if available, to further solve somatic genome architecture at the haplotype level in aneuploid cancer genomes.

The goal of this work is to develop a new algorithm to fully leverage the output from Weaver to further improve SV phasing. Given copy number predictions from some source, e.g., from sources such as [1] or from Weaver, and a large number of putative unphased SVs, we wish to use the new algorithm to further predict SV phasing accurately. Here, we develop a convex optimization framework which minimizes a flow-like objective function while phasing the set of unphased SVs. We implement an integer linear program derived from this framework and test it on both simulations and real data to demonstrate that our method is not sensitive to false positives, and robust to copy number noise in the input. We aim to extend this method to account for long read data in the future, and support the Weaver framework in obtaining a more complete description of a cancer genome.

2 Background

As we previously mentioned, one major limitation of Weaver is that it does not always phase all SVs. For example, out of 36 TCGA ovarian cancer patient samples we analyzed using Weaver in this paper, the average fraction of unphased SVs is a little over 30% and many as 53% of the detected SVs (where the total number varies from 17 to 527) may be unphased in these samples. This may be

due to low read support for the SV or due to balanced copy numbers for the bordering region alleles. Currently, Weaver also does not predict the phasing of copy number neutral events, such as inversions. On the other hand, we know that Weaver can produce accurate estimation of ASCNG in aneuploid tumor genomes [13]. Our goal here is to improve upon, or predict the phasing of SVs, given the allele-specific copy numbers (ASCNs). Note that in this work we do not seek information from germline SNPs to help phasing as we assume that we have already exhausted the information from SNPs and the read linkage provided by the paired-end reads from the sequencing data.

2.1 Problem Setting

Our input is a set of genomic regions and SVs. A genomic region is specified by a pair of loci (i.e., coordinates) on a chromosome in the reference genome, called the *extremities* of that genomic region. We assume that the set of all genomic regions is disjoint, i.e., there are no two regions such that one or both extremities of one region lies between the extremities of the other. An SV is specified by a set of genomic loci, called *breakpoints*. These breakpoints need not lie on the same chromosome. Since Weaver infers pairs of breakpoints as SVs, we use the terms ‘pair of breakpoints’ and SVs interchangeably. In the input, each genomic locus is associated with at most one breakpoint: this is called the *infinite sites* assumption [12, 14]. This assumption comes as a natural consequence of the hypothesis that the probability of a single genomic site being subjected to an SV breakpoint is usually low. Formally, breakpoints may be assumed to occur randomly across the genome, following an unobserved probability distribution. If the number of genomic loci is very large, this distribution may be approximated by a continuous density function. Under this assumption, a genomic locus will almost surely not be used as a breakpoint more than once, even if some regions are more likely to host breakpoints than others. However, this assumption may be invalid if the breakpoints are resolved at a very low resolution. Therefore, in the context of providing a general framework, the infinite-sites assumption can be ignored as a condition which is probably too strong in real data.

The normal human genome is diploid, where each genomic region on a chromosome can be associated with two *alleles*. Specifically, it is possible to differentiate between chromosome regions that arise from the paternal copy of the chromosome, and those that come from the maternal copy of the chromosome. In a non-cancerous genome, for most genomic regions, we expect exactly one copy of the chromosome region from each allele. In a tumor genome, on the other hand, due to somatic CNAs and SVs, as well as aneuploidy, we frequently find multiple copies of allelic regions, or regions from a specific allele which are lost (*loss of heterozygosity*). It is now acknowledged that CNAs and whole-genome duplications are prevalent in various types of cancer genomes [20]. Therefore, it is important to accurately characterize the *allele-specific copy number* of a genomic region in the tumor genome. Furthermore, SVs and CNAs in cancer genomes could lead to rearranged haplotype structure and a different interconnectivity of the SVs from that seen in the non-cancerous genome. Thus, if we

can identify the connections among different SVs at the haplotype level (i.e., providing a *phasing*), we can specify the haplotype that the breakpoints lie in, resolving the complexity of aneuploid cancer genome to a more refined resolution. We can also associate an ASCN to SVs, based on how many times the phased variant is found in the tumor genome. ASCN of SVs (ASCNS) is also estimated through Weaver [13].

Figure 1A shows an example of how a cancer genome may compare to a normal genome. The figure shows two copies of the same chromosome, with blocks representing regions, black edges representing connections between regions which were adjacent in the normal genome, and colored edges representing connections between regions which are detected in the cancer genome. Red edges are adjacencies representing SVs that are phased. Green edges represent a possible phased configuration of an unphased SV. The numbers depict the number of copies of each region/SV. The goal is to predict a phasing of the SVs represented by the green edges, which translates to setting the ASCN of the other possibilities to 0, that minimizes an objective function which corresponds to ‘discordant’ ASCN information in the input data.

2.2 Method Overview

Using the regions and SVs, and their corresponding ASCNs as predicted by Weaver as input, we can construct an allele-specific cancer genome graph. In such a graph, we represent different alleles of a region by different vertices. Specifically, we represent each region extremity by two vertices, each representing a different allele with the corresponding ASCN. Two vertices are then adjacent to each other if the corresponding allele-specific regions occurred next to each other in the tumor genome, i.e., they either represent a putative SV with allele-specific breakpoints, or the adjacency has never been broken in the cancer genome. In the resulting graph, there may be breakpoints adjacent to more than one SV edge, representing the possible phasings of an SV. Thus, the graph is incomplete, in the sense that there is a set of edges in the graph which are either not representative of the actual adjacency information in the tumor sample. If all the SVs were phased, and all ASCNs were predicted correctly, the copy number of an allele-specific region should be equal to the cumulative degree (sum of weights on edges) at each extremity.

Our aim is to use the ASCNs predicted by Weaver to phase unphased SVs. To achieve this, we designed an optimization problem which attempts to balance the inflow and outflow through a region, while conforming to a set of constraints which specify the structure of the problem.

3 Method

3.1 Preliminaries

We now introduce the notion of an *incomplete allele-specific cancer genome graph* (see example in Fig. 1B), a representation of the input where vertices and edges encode region extremities and adjacencies between them, respectively.

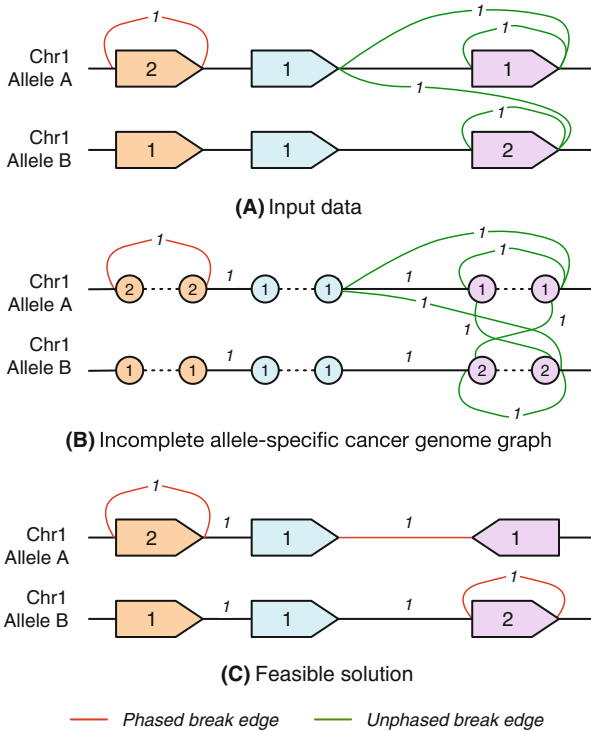


Fig. 1. An example of an incomplete allele-specific cancer genome graph, showing two alleles of the same chromosome. (A) shows the problem setting, where two copies of the same chromosome are given. Genomic regions are given as blocks, adjacencies between blocks present in the reference genome are shown in black, and SVs are given by colored edges. Phased SVs, with the allele of both breakpoints known, are in red, and those which are unphased are in green. ASCNs for both genomic regions and SVs are shown as numbers near the regions/edges. If an SV is unphased, there are many possibilities for the correct phasing. (B) shows the translation into an *incomplete allele-specific cancer genome graph*, with regions being replaced by 2 vertices each, and copy numbers assigned to all SVs and possible non-cancer edges. This is the instance on which the objective (3) is defined. A correct solution will modify the copy numbers of the edges to minimize (3), selecting one of the many possible SVs. (C) shows a feasible solution to the toy problem presented. Edges which are assigned copy number 0 in this solution are not shown, and SVs which are phased are now given in red.

Definition 1. The set of region extremities V is a set of objects, each of which has the following characteristics.

1. Each $v \in V$ is associated to a chromosome and a position on the chromosome.
2. There is a symmetric relation on the set V which associates each $v \in V$ with a corresponding extremity called its mate in V . We denote the mate of an extremity v by \bar{v} .
3. v has an associated allele, such that \bar{v} is also associated with the same allele.

4. Every $v \in V$ is associated to another region extremity $\gamma(v) \in V$, which is called its variant. $\gamma(v)$ is associated with the same chromosome and position as v , but must be associated to a different allele. The relation γ is symmetric (i.e., $\gamma(\gamma(v)) = v$), and $\gamma(\bar{v}) = \overline{\gamma(v)}$ for every $v \in V$.

Since the regions are all associated with allele-specific copy numbers, we define a multiplicity function on the set of region extremities as follows.

Definition 2. The multiplicity function $\mu: V \rightarrow \mathbb{N}$ is a non-negative integer function on a set of region extremities with the following conditions.

1. For every $v \in V$, $\mu(v) = \mu(\bar{v})$.
2. If allele labels are not known, region extremities are labelled as either major or minor, such that, if extremity v is a major allele, $\mu(v) \geq \mu(\gamma(v))$.

We can now define the main object that we study in our problem.

Definition 3. An incomplete allele-specific cancer genome graph $G = (V, E)$ is an undirected graph, where the set of vertices V is a set of region extremities, and the set of edges E is defined as follows.

1. There is an edge between all region extremities which are putatively adjacent in the normal genome. We call these non-break edges. All other edges will be called break edges.
2. For every SV that is phased, we add an edge between the vertices corresponding to the region extremities which form the breakpoints of the SV. We call these phased edges.
3. For every SV in which exactly one breakpoint is unphased, we add edges from the vertex corresponding to the phased region extremity to both possible alleles of the unphased region extremity.
4. For every SV in which both breakpoints are unphased, we add edges from the vertices corresponding to both alleles of one region extremity to those corresponding to both alleles of the other extremity.

Edges that are not phased are called unphased edges. E_v denotes the set of edges adjacent to a vertex v .

A non-negative integer function $\nu: E \rightarrow \mathbb{N}$ is defined on the set of edges, which we will call the edge multiplicity function.

Note that an incomplete allele-specific cancer genome graph may be disconnected. ν may be defined as a partial function ν' . In this case we extend ν , such that $\nu(e) = \nu'(e)$, if ν' is defined on $e = \{u, v\} \in E$, and $\nu(e) = \min\{\mu(u), \mu(v)\}$ otherwise. The functions μ and ν represent the expected number of copies of the region/SV as observed in the tumor sample, respectively. They are also important when deciding on a traversal of the graph. Such a traversal would correspond to a possible representation of the cancer genome, with each linear or cyclic walk corresponding to a linear or circular chromosomal segment formed after genomic alterations. Figure 1B shows an example of an incomplete allele-specific cancer genome graph.

Given an incomplete allele-specific cancer genome graph $G = (V, E)$, with multiplicity and edge multiplicity functions μ and ν , respectively, the *imbalance* of a region r , associated with the extremities v, \bar{v} , is the following quantity.

$$\left| \sum_{e' \in E_v} \nu(e') - \sum_{e' \in E_{\bar{v}}} \nu(e') \right|.$$

A non-telomeric region is said to be balanced if it has imbalance 0. This is the same as saying that the number of regions adjacent to one end of an allele-specific region r in the cancer genome is equal to the number of regions adjacent to the other end. Clearly, this condition does not hold for telomeric regions. An incomplete allele-specific cancer genome graph in which all regions are balanced with respect to functions μ and ν is said to be (μ, ν) -*resolved*. If μ and ν are clear from the context, we just say the graph is *resolved*.

3.2 The Convex Program

Assume we are given an incomplete allele-specific cancer genome graph $G = (V, E)$, and let $\mu: V \rightarrow \mathbb{N}$ and $\nu: E \rightarrow \mathbb{N}$ be the multiplicity and edge multiplicity functions on V and E , respectively. Our goal is to edit the functions μ and ν to μ' and ν' , respectively, so that the total imbalance over all regions is minimized.

Let x_v be a variable associated with every vertex $v \in V$, and let y_e be a variable associated with every edge. We define $\delta(v)$ for a vertex v , and $\Delta(v, \bar{v})$ as the following variables.

$$\delta(v) = \mu(v) + x_v - \sum_{e \in E_v} (\nu(e) + y_e), \tag{1}$$

$$\Delta(v, \bar{v}) = |\delta(v) - \delta(\bar{v})|. \tag{2}$$

More generally, we will define $\delta(v)$ to be within an ϵ threshold of the term on the right hand side in (1), where $0 < \epsilon < 1$. Note that $\Delta(v, \bar{v})$ is analogous to the imbalance of the region defined by v, \bar{v} , and only differs by the new variables introduced. Using these definitions, we can now define the following problem.

Problem 1 (Total allele specification problem). Given an incomplete allele-specific cancer genome graph $G = (V, E)$, find an optimal integer solution to the following convex program.

$$\min_{\{x_v\}, \{y_e\}} \sum_{\{v, \bar{v}\}} w_{v, \bar{v}} \Delta(v, \bar{v}) + \sum_v \lambda_v |x_v| + \sum_e \lambda_e |y_e| \tag{3}$$

subject to:

$$x_v - x_{\bar{v}} = 0, x_v \geq -\mu(v), \quad \delta(v) \geq 0 \quad \forall v \in V, \tag{4}$$

$$y_e \geq -\nu(e) \quad \forall e \in E, \tag{5}$$

$$(\nu(e) + y_e) \cdot (\nu(e') + y_{e'}) = 0 \quad \forall e, e' \in E_v, e \neq e' \tag{6}$$

being break edges, $\forall v \in V$,

where we optimize over all variables x_v and y_e for all vertices and edges, respectively, the sum in (3) is over all distinct sets $\{v, \bar{v}\}$ of extremity and mate extremity in V , $w_{v, \bar{v}}$ is a real, non-negative weight, and λ_v, λ_e are real positive parameters used for regularization.

This is a feasible, bounded integer program, interpreted in the following section. A solution to this problem returns a ‘smoothing’ of the copy numbers, and an ASCN for every possible phasing of an unphased SV.

3.3 Interpreting the Objective and Constraints

The objective function (3) is the sum of the imbalance over all regions defined in the incomplete allele-specific cancer genome graph, along with two regularization terms. We interpret the total imbalance as follows. Assume that a region defined by vertices v, \bar{v} has copy number k and is not telomeric in any chromosome in the cancer genome. Clearly, the number of adjacencies next to the region at extremity v should be equal to the number of adjacencies next to the region at extremity \bar{v} , and should be equal to k . If so, the difference between the sum of the copy number of the edges from v and the sum of the edges from \bar{v} should be 0. This is the expected imbalance of the region, which we are trying to minimize.

By definition (1), $\delta(v)$ itself is a constraint which states that every region is only adjacent to as many other regions as its copy number. However, it allows the copy number of the region to be greater than the sum of the copy number of the edges adjacent to it at a single end. The variables x_v and y_e are the amounts by which we must modify the copy numbers in order to find a minimum total imbalance solution to the integer program described. However, an ASCN cannot be negative, ensured by Constraints (4) and (5). The extra terms in the objective are regularization terms to make sure that the copy numbers are not changed significantly from the original assignment.

Constraint (6) is the only non-linear constraint specified in the problem. It states that, assuming there are two break edges adjacent to the same extremity of a region, then the copy number of at least one of these two edges must be 0. In other words, it enforces the very strong infinite sites assumptions. This assumption only remains mostly valid when the breakpoints are resolved to the nucleotide level. We shall later discuss how to interpret solutions for a system in which we discard this constraint.

To understand how the program aids in phasing breakpoints, we note that in case an SV is not phased, then we add all possible phasings of the SV edges to the input graph. However, this causes an imbalance in the concerned regions on which the breakpoints lie. In order to reduce/remove this imbalance, the method will modify the copy numbers. It will set the copy numbers of all unsupported phasings of an SVs to 0 while solving the problem, thus finding an assignment of phase to the unphased breakpoints. A solution for the toy example presented is shown in Fig. 1C.

It is instructive to compare the presented formulation against the integer linear program (ILP) proposed by Oesper et al. [17] to infer the structure of

cancer genomes, and to the flow framework presented by Dzamba et al. [5] (which itself was based on prior work by [15, 16]). While their methods are not designed to resolve SVs at an allele-specific level, it uses the same basic principles for inferring SVs, and presents the ILP as a maximum likelihood problem. The constraints of these ILPs include the flow condition, i.e., the total in-flow should be equal to the total out-flow through a node. In comparison, the objective (3) in our formulation is presented as a combinatorial problem in which we seek to minimize the total number of flow-like constraints violated, anticipating the problem of missing edges or incorrectly estimated copy numbers. A theoretical basis for similar frameworks is also given in [21].

3.4 Removing Non-linear Constraints

Recall that Constraint (6) is included to ensure that every vertex has maximum degree 2. If we remove it, we would allow the possibility of multiple break edges adjacent to a vertex, violating the infinite sites assumption, and the resulting solution may lead to finding breakpoints which are used more than once. This is a practical consideration that allows for breakpoints that are not resolved at a nucleotide level. Since we are dealing with possibly noisy data in order to resolve breakpoints, we will assume that we do not have nucleotide-level resolution of breakpoints, and discard the infinite-sites assumption as a very strong constraint on the data used. The absence of this constraint simplifies the problem to an ILP, which is easier to solve in practice, though still theoretically NP-hard.

If, however, we wish to only analyze SVs which have a single unambiguous phasing, we can discard all SVs for which more than one putative phasing is predicted. This guarantees that none of the predicted phased variants is a false positive. However, as we shall see on real data, the relative number of SVs for which more than one possible phasing is predicted is generally small compared to those successfully phased.

3.5 Implementation

We implemented the ILP in Python using the CVXPY package [4] and used GUROBI [10] as the preferred solver. The program obtains an exact integer solution to the linear program (3). While integer linear programs are known to be NP-hard [11], they can be efficiently solved in practice for instances with thousands of variables using sparse matrix data structures. This is important, since the number of variables can be anywhere between 5,000 and 50,000, depending on the input data. This is because the number of vertices in the graph may vary from a few thousand to tens of thousands, and number of edges usually scales linearly with the vertices. We require auxiliary variables in the ILP to represent some constraints.

4 Results

4.1 Evaluation by Simulations

In order to evaluate the method, we simulated 25 datasets having region sizes sampled from the distribution of regions produced by Weaver on the HeLa whole-genome sequencing data in Adey et al. [1, 13]. We simulated a set of SVs on this set of regions such that the number of regions that are copy number altered is relatively close (~ 250 regions) to that produced by Weaver.

In the set of simulations created, we introduced copy number errors and deleted the phasing of the breakpoints in order to create the input for the method. We created two such subsets. In the first subset of simulations, we discarded the phasing of SVs with probability 0.3, which is the expected fraction of unphased SVs in the Weaver output, and perturbed the copy numbers with the same probability. On average, each simulation in this subset has about 102 unphased SVs. However, the number of SVs used is higher than those which are analyzed by Weaver in its final step. In the second subset, we used the same parameters but discarded SVs with probability 0.3, so that the number of SVs is similar to those analyzed by Weaver. This simulated the cases where the set of SVs cannot account for all the CNAs observed in the data. On average, each simulation in this subset has about 69 unphased SVs.

We used static parameters of ℓ , 100ℓ , and 0.01 as the cost, region regularization, and edge regularization parameters, respectively, where ℓ is the length of a region. The main reasoning behind the choice of the parameters is that we generally have more confidence in the ASCN of a region, since we expect a large region to also support enough read alignments to be able to confidently predict its copy number.

Under these parameters, we present the phasing results in Fig. 2. If we have the entire set of possible SVs, over 50% of the ~ 100 unphased SVs are correctly phased, while the number of incorrectly phased SVs is quite low ($< 10\%$). However, a large number of SVs may remain unphased, i.e., more than 1 possible phasing is predicted for them. If we do not detect $\sim 30\%$ of the SVs that occurred, the noise added causes the quality of the phasing to drop significantly, and the mean percentage of correctly phased SVs drops to about 40%. The number of incorrectly phased SVs remains under 10%, which means the method does not create many false positives and is quite specific, but in the absence of the non-linearity, the number of unphased SVs remains high.

Figure 3 shows how the number of incorrectly phased SVs, i.e., false positives, in the two simulation datasets varies as the two regularization parameters are changed. We note that the number of incorrectly phased SVs is always quite low, at $\leq 15\%$ of the number of unphased SVs in the input. Indeed, assuming all SVs are detected, then with reasonable parameter values the number of errors during phasing can be kept to single digits. However, if the regularization coefficient for the edges exceeds that used for regions, the quality of the phasing drops significantly in both data sets. This is expected behavior, considering that the number of errors in ASCNs of regions is significantly lower than that for edges.

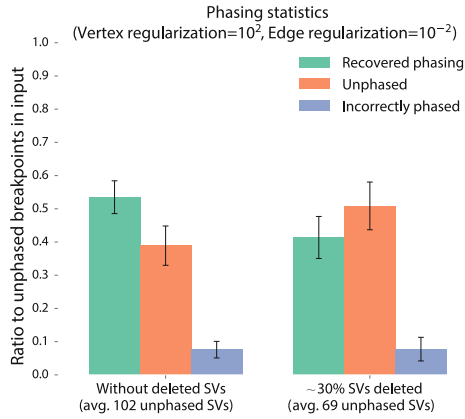


Fig. 2. Phasing results for the two different simulation sets we create. Both sets consist of approximately the same number of genomic regions as the HeLa results produced by Weaver [13]. In the first one, we keep all the simulated SVs, but discard the phasing of breakpoints with probability about 0.30. This leads to ~ 102 unphased SVs, of which we can correctly phase $\sim 52\%$. In the second set, we also delete SVs with probability 0.30, leading to ~ 69 SVs on average. In this case, the missing information from the deleted breakpoints affects the phasing inference, and the number of recovered SVs that are phased correctly is fewer than those that remain unphased. In both cases, however, the number of wrongly phased SVs is only $\sim 8\%$.

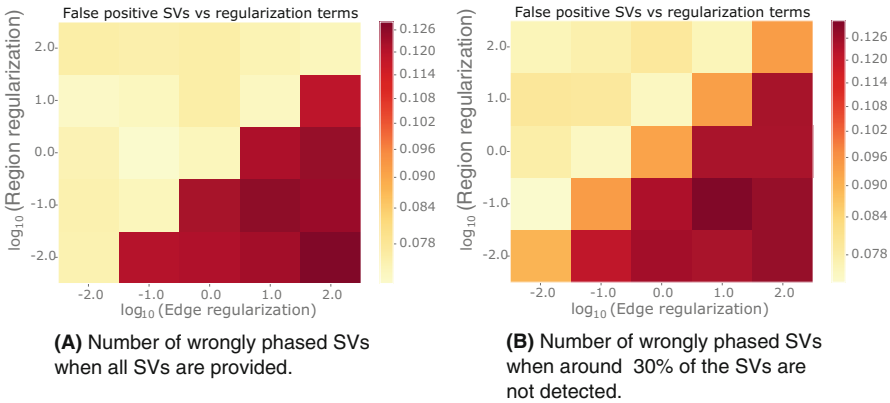


Fig. 3. Variation in the number of incorrectly phased SVs with variation in the two regularization terms. The weighting coefficient is kept constant at 1. The axes are log scaled.

Therefore, if the ASCNS is perturbed and we have to find the phasing, it is more likely to introduce errors by disallowing large corrections in the ASCNS than by disallowing large corrections in the ASCNG.

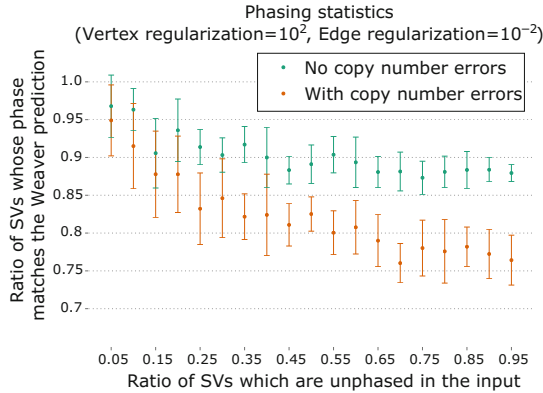


Fig. 4. Phasing results compared to Weaver on the HeLa whole-genome sequencing data. The X-axis represents the fraction of unphased SVs in the input provided to our method, while the Y-axis is the fraction of SVs where the recovered phasing agrees with that obtained by Weaver. The green plot shows the results when the input does not have noisy copy numbers, which should aid phasing. The orange plot shows how the results vary when copy numbers of SVs and regions are perturbed with probability 0.3.

4.2 Application to the HeLa Whole-Genome Sequencing Data

To evaluate the method on real data, we used as input noisy results from Weaver on the HeLa whole-genome sequencing dataset [13] on the HeLa genome [1]. In this input, we examined the results for our method when the number of phased breakpoints is varied. We then verified how our recovered phasing compares with the prediction from Weaver, and how it is affected when noise is introduced in the form of perturbation in the copy numbers of the SVs and the regions.

Figure 4 shows the fraction of SVs where our phasing prediction is similar to that predicted by Weaver versus the fraction of unphased SVs in the input. The results were calculated at parameter values of ℓ , 100ℓ , $.01$ for the cost, region regularization, and edge regularization parameters respectively, where ℓ is the length of the corresponding region. In the results obtained, we found that our method manages to recover a large percentage of discarded phasing information. When the copy numbers of the regions and the SVs are known, we can recover almost 90% or better of the predicted phasing by Weaver, irrespective of the number of SVs that are already phased. Weaver, on the other hand, uses germline SNP data and their haplotype information from the 1000 Genomes Project to predict phasing.

When noise is introduced, the fraction of recovered phasing drops to $\sim 74\%$ when only 5% of the original input is phased, but it rises to $\sim 90\%$ if even 50% of the original input is phased. Therefore, any noise in the copy number data is being offset by prior information about the phasing of some SVs. This information allows the method to balance some of the copy numbers so that the phasing of the remaining SVs can be recovered. In Fig. 5, we provide an example

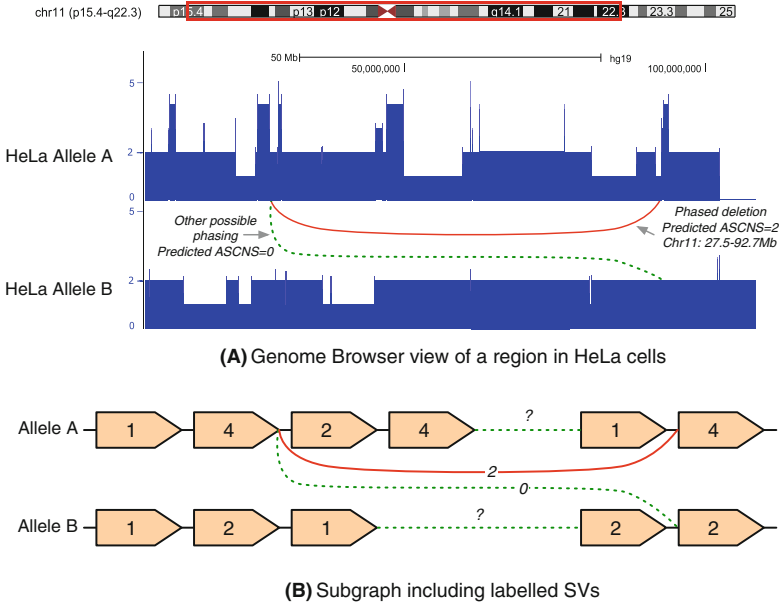


Fig. 5. (A) A Genome Browser view that shows two possible phasings of a given SV (a deletion) in the HeLa genome, where the phasing prediction by Weaver was discarded, in a region with a large number of unphased SVs and CNAs. The number of CNAs and SVs makes it liable to predict the phasing incorrectly. The two tracks represent allele A and allele B frequencies for regions on chromosome 11 as predicted by Weaver. The red edges are possible phasings of an SV detected on the chromosome. On solving (3), one of these possibilities is predicted to have copy number 0 (dotted edge), and the other (solid edge), which is the phasing predicted by Weaver, has copy number 2. Note that other unphased SVs in the figure are not shown for clarity. (B) A schematic representation of the corresponding incomplete allele-specific cancer genome graph near the region shown in (A). Here the regions are represented as oriented blocks, with the number on each block referring to their ASCNG. The green connections represent a large set of regions excluded in the schematic. These regions are also subject to unrepresented SVs that may have confound the inference. Despite this, the inference agrees with Weaver’s prediction, which was erased from the input.

of an SV in a region with CNAs in the HeLa data, where the predicted phase is consistent with the Weaver prediction. Note that the prediction is unaffected by other unphased SVs (not shown in the figure) in a region with a large number of CNAs.

4.3 Application to Ovarian Cancer Data from TCGA

In Li et al. [13], Weaver was used to infer allele-specific copy numbers and phasing information for TCGA ovarian cancer samples. The number of SVs detected in these samples varied from under 50 to over 500. However, on average, 23% of

the SVs detected in a sample were unphased, and some of the detected unphased variants were inferred to have copy number 0. Some samples had over 50% of the detected SVs unphased at either one or both breakpoints. We attempted to resolve these variants using the convex programming framework introduced in this paper, and classified the SVs into 3 classes: (1) variants that were phased using the framework and were estimated to have non-zero copy number; (2) variants that remained unphased; and (3) variants whose copy number is predicted to be 0 by both Weaver and the ILP.

In these results (Table 1), on average, about 60% of the unphased SVs in the samples were estimated to have a copy number of 0, matching the estimation through Weaver. We classify these as undetermined variants, since we cannot distinguish the classes they might fall into, nor the phasing. However, of the other 40% of the SVs which are predicted to have non-zero copy number, almost 60% are phased on average, with the number of ambiguous phasings never rising above single digits. Thus, for a large number of SVs with predicted copy number >0 , we are able to obtain a single, unambiguous phasing.

5 Discussion and Conclusion

The main contribution of this paper is a combinatorial framework within which we can examine allele-specific rearrangements in cancer genomes. An advantage of our method is that given ASCN information and putative SVs from any source, we can infer phased SVs efficiently. We showed that this framework, as implemented here, has high specificity. As a proof-of-principle, we demonstrated the performance of our method on real data. Even in the presence of external noise in the form of errors in the input ASCNs, our results agreed closely with the original predicted results from Weaver that used many other types of input data (germline SNP and read linkage from paired-end reads) to obtain SV phasing information.

The method proposed in this paper can also be extended for more sophisticated inference. We will explore adding further non-linear constraints which can capture information gathered from long-range sequencing and alignment data, such as PacBio and 10X Genomics data. In such a framework, we should be able to incorporate both long and short read data into a comprehensive assembly-like problem formulation and obtain a description of the cancer genome through the cancer genome graph. In addition, a theoretical analysis of the method, in the sense of accuracy of rounded solutions to relaxations of problem, can also be pursued. Assuming we can bound the error in our prediction of value of the objective function, we believe the main drawback in the current iteration of the method, which is the number of false negatives, can be overcome. We also do not specifically consider the problem of subclones in this work. Indeed, if multiple clones are present, then the incomplete allele-specific cancer genome graph obtained will represent SVs from different subclones, and the problem of decomposing the graph into individual clones needs to be addressed. Finally, from a practical point of view, the ILP framework is fast and efficient for most cases,

Table 1. The results of the ILP on 36 ovarian cancer samples from TCGA. The first column gives the sample ID. The second column shows the number of the SVs identified by Weaver, while the third column shows the number of SVs which are left unphased. The fourth column shows the number of SVs which are phased by the ILP and are predicted to have more than 1 copy. The fifth column is the number of SVs whose phase is left ambiguous by the ILP, though Weaver predicts that they have non-zero copies. The sixth and final column shows the number of SVs that are predicted to have copy number 0 by both Weaver and the ILP. These are classified as undetermined SVs.

Sample name	SVs detected by Weaver	Unphased SVs	Newly phased SVs	Remaining unphased SVs	Undetermined SVs
TCGA-04-1331	80	24	10	3	11
TCGA-04-1347	174	25	11	5	9
TCGA-04-1349	62	7	3	1	3
TCGA-04-1367	59	24	4	2	18
TCGA-04-1514	173	73	14	7	52
TCGA-09-1666	222	105	21	8	76
TCGA-09-2045	105	14	4	2	8
TCGA-09-2050	120	21	3	6	12
TCGA-10-0934	128	7	0	2	5
TCGA-10-0937	100	17	6	2	9
TCGA-10-0938	276	89	18	7	64
TCGA-13-0725	21	8	2	2	4
TCGA-13-0751	120	19	6	5	8
TCGA-13-0906	94	22	1	6	15
TCGA-13-1477	17	6	2	1	3
TCGA-13-1487	275	29	5	5	19
TCGA-13-1491	101	19	4	1	14
TCGA-23-1110	145	32	7	5	20
TCGA-24-0982	32	8	2	4	2
TCGA-24-1419	51	10	4	1	5
TCGA-24-1466	527	39	16	9	14
TCGA-24-1544	288	152	13	7	132
TCGA-24-1548	73	9	2	2	5
TCGA-24-1557	265	37	8	4	25
TCGA-24-1558	189	26	8	3	15
TCGA-24-1562	92	11	4	1	6
TCGA-24-1614	180	80	11	9	60
TCGA-24-2024	67	19	1	3	15
TCGA-24-2290	168	55	10	4	41
TCGA-25-1632	165	11	2	2	7
TCGA-25-1634	118	28	8	8	12
TCGA-25-2391	88	24	2	3	19
TCGA-25-2400	213	54	29	6	19
TCGA-36-1570	161	29	7	6	16
TCGA-36-1574	114	21	9	3	9
TCGA-61-2000	460	90	13	3	74

but there is no guarantee of a polynomial time solution. For certain input programs, for example, the algorithm may take exponential time, and it is hard to quantify which cases these are. In the context of the large graphs being handled, this could be a potential problem. A more thorough analysis, a more efficient implementation, and a study of the structure of edge cases would improve the utility of the method.

Acknowledgments. The authors would like to thank anonymous reviewers for suggestions that improved the paper. The authors would also like to thank the TCGA Research Network for making the data publicly available. This work is supported in part by National Institutes of Health Grants CA182360, HG007352, and DK107965 (to J.M.), and National Science Foundation Grants 1054309 and 1262575 (to J.M.).

References

1. Adey, A., Burton, J.N., Kitzman, J.O., Hiatt, J.B., Lewis, A.P., Martin, B.K., Qiu, R., Lee, C., Shendure, J.: The haplotype-resolved genome and epigenome of the aneuploid HeLa cancer cell line. *Nature* **500**(7461), 207–211 (2013)
2. Beroukhi, R., Mermel, C.H., Porter, D., Wei, G., Raychaudhuri, S., Donovan, J., Barretina, J., Boehm, J.S., Dobson, J., Urashima, M., et al.: The landscape of somatic copy-number alteration across human cancers. *Nature* **463**(7283), 899–905 (2010)
3. Carter, S.L., Cibulskis, K., Helman, E., McKenna, A., Shen, H., Zack, T., Laird, P.W., Onofrio, R.C., Winckler, W., Weir, B.A., et al.: Absolute quantification of somatic DNA alterations in human cancer. *Nat. Biotechnol.* **30**(5), 413–421 (2012)
4. Diamond, S., Boyd, S.: CVXPY: a python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.* **17**(83), 1–5 (2016)
5. Dzamba, M., Ramani, A.K., Buczkowicz, P., Jiang, Y., Yu, M., Hawkins, C., Brudno, M.: Identification of complex genomic rearrangements in cancers using CouGaR. *Genome Res.* **27**(1), 107–117 (2017)
6. Eid, J., Fehr, A., Gray, J., Luong, K., Lyle, J., Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., et al.: Real-time DNA sequencing from single polymerase molecules. *Science* **323**(5910), 133–138 (2009)
7. Gordon, D.J., Resio, B., Pellman, D.: Causes and consequences of aneuploidy in cancer. *Nat. Rev. Genet.* **13**(3), 189–203 (2012)
8. Greenman, C.D., Pleasance, E.D., Newman, S., Yang, F., Fu, B., Nik-Zainal, S., Jones, D., Lau, K.W., Carter, N., Edwards, P.A., et al.: Estimation of rearrangement phylogeny for cancer genomes. *Genome Res.* **22**(2), 346–361 (2012)
9. Gupta, A., Place, M., Goldstein, S., Sarkar, D., Zhou, S., Potamouisis, K., Kim, J., Flanagan, C., Li, Y., Newton, M.A., et al.: Single-molecule analysis reveals widespread structural variation in multiple myeloma. *Proc. Nat. Acad. Sci.* **112**(25), 7689–7694 (2015)
10. Gurobi Optimization Inc.: Gurobi optimizer reference manual (2015)
11. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) *Complexity of Computer Computations*, pp. 85–103. Springer, New York (1972)
12. Kimura, M.: The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics* **61**(4), 893 (1969)

13. Li, Y., Zhou, S., Schwartz, D.C., Ma, J.: Allele-specific quantification of structural variations in cancer genomes. *Cell Syst.* **3**(1), 21–34 (2016)
14. Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Miller, W., Haussler, D.: The infinite sites model of genome evolution. *Proc. Nat. Acad. Sci.* **105**(38), 14254–14261 (2008)
15. Medvedev, P., Fiume, M., Dzamba, M., Smith, T., Brudno, M.: Detecting copy number variation with mated short reads. *Genome Res.* **20**(11), 1613–1622 (2010)
16. Medvedev, P., Stanciu, M., Brudno, M.: Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods* **6**, S13–S20 (2009)
17. Oesper, L., Ritz, A., Aerni, S.J., Drebin, R., Raphael, B.J.: Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinform.* **13**(6), S10 (2012)
18. Van Loo, P., Nordgard, S.H., Lingjærde, O.C., Russnes, H.G., Rye, I.H., Sun, W., Weigman, V.J., Marynen, P., Zetterberg, A., Naume, B., et al.: Allele-specific copy number analysis of tumors. *Proc. Nat. Acad. Sci.* **107**(39), 16910–16915 (2010)
19. Wang, J., Mullighan, C.G., Easton, J., Roberts, S., Heatley, S.L., Ma, J., Rusch, M.C., Chen, K., Harris, C.C., Ding, L., et al.: Crest maps somatic structural variation in cancer genomes with base-pair resolution. *Nat. Methods* **8**(8), 652–654 (2011)
20. Zack, T.I., Schumacher, S.E., Carter, S.L., Cherniack, A.D., Saksena, G., Tabak, B., Lawrence, M.S., Zhang, C.Z., Wala, J., Mermel, C.H., et al.: Pan-cancer patterns of somatic copy number alteration. *Nat. Genet.* **45**(10), 1134–1140 (2013)
21. Zerbino, D.R., Ballinger, T., Paten, B., Hickey, G., Haussler, D.: Representing and decomposing genomic structural variants as balanced integer flows on sequence graphs. *BMC Bioinform.* **17**(1), 400 (2016)
22. Zheng, G.X., Lau, B.T., Schnall-Levin, M., Jarosz, M., Bell, J.M., Hindson, C.M., Kyriazopoulou-Panagiotopoulou, S., Masquelier, D.A., Merrill, L., Terry, J.M., et al.: Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. *Nat. Biotechnol.* **34**(3), 303–311 (2016)

Using Stochastic Approximation Techniques to Efficiently Construct Confidence Intervals for Heritability

Regev Schweiger¹(✉), Eyal Fisher², Elior Rahmani¹, Liat Shenhav², Saharon Rosset², and Eran Halperin^{3,4}

¹ Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel
schweiger@post.tau.ac.il

² School of Mathematical Sciences, Department of Statistics,
Tel Aviv University, Tel Aviv, Israel

³ Department of Computer Science, University of California, Los Angeles, CA, USA

⁴ Department of Anesthesiology and Perioperative Medicine,
University of California, Los Angeles, CA, USA

Abstract. Estimation of heritability is an important task in genetics. The use of linear mixed models (LMMs) to determine narrow-sense SNP-heritability and related quantities has received much recent attention, due of its ability to account for variants with small effect sizes. Typically, heritability estimation under LMMs uses the restricted maximum likelihood (REML) approach. The common way to report the uncertainty in REML estimation uses standard errors (SE), which rely on asymptotic properties. However, these assumptions are often violated because of the bounded parameter space, statistical dependencies, and limited sample size, leading to biased estimates and inflated or deflated confidence intervals. In addition, for larger datasets (e.g., tens of thousands of individuals), the construction of SEs itself may require considerable time, as it requires expensive matrix inversions and multiplications.

Here, we present FIESTA (Fast confidence Intervals using Stochastic Approximation), a method for constructing accurate confidence intervals (CIs). FIESTA is based on parametric bootstrap sampling, and therefore avoids unjustified assumptions on the distribution of the heritability estimator. FIESTA uses stochastic approximation techniques, which accelerate the construction of CIs by several orders of magnitude, compared to previous approaches as well as to the analytical approximation used by SEs. FIESTA builds accurate CIs rapidly, e.g., requiring only several seconds for datasets of tens of thousands of individuals, making FIESTA a very fast solution to the problem of building accurate CIs for heritability for all dataset sizes.

1 Introduction

Heritability, or the proportion of phenotypic variation that is explained by genetic variation, is an important population parameter in human genetics, in evolution, in plant and animal breeding, and more. Estimating the heritability

has been traditionally performed using related individuals such as in twin studies or pedigree designs [1–3]. More recently, genetic variation has been estimated using genetic marker information, and in particular in genome-wide association studies (GWAS) [4, 5], which have identified thousands of genetic variants that are associated with dozens of common diseases. However, genome-wide significant associations were generally found to explain only a small proportion of the heritability of complex diseases.

To cope with this challenge, linear mixed model (LMM) approaches [6–13] have been applied to estimate the heritability explained by common SNPs (the narrow-sense SNP-heritability, to which we refer as heritability, and denote by h^2) from cohorts of unrelated individuals, such as those found in GWAS [14]. Estimation under the LMM is usually performed using restricted maximum likelihood (REML) estimation, and is implemented in some widely used tools, like the GCTA software package [15]. LMMs utilize all variants from a GWAS, and not just the variants that are statistically significant, and therefore is able to account for variants with small effect sizes.

As in any statistical analysis, the process of estimating the heritability suffers from statistical uncertainty. Typically, confidence intervals (CIs) are reported alongside with point estimates to quantify this uncertainty. Usually, such CIs are constructed from standard errors (SEs), which make the assumption that the estimators asymptotically follow a normal distribution. However, it has been shown [13, 16–20] that such CIs can be highly inaccurate. This is because estimators do not necessarily obey the conditions required for them to asymptotically follow the normal distribution. Additionally, these CIs may spread beyond the natural boundaries of their parameters, e.g., including negative values for heritability. As a result, these CIs are often inaccurate, difficult to interpret, or lead to erroneous conclusions.

To handle these issues, previous approaches have taken several directions. Non-standard asymptotic theory for boundary and near-boundary maximum likelihood estimates has been developed (e.g., [21–23]), and it has been suggested to replace the asymptotic normality assumption with the asymptotics developed for the non-standard boundary case [24]. Visscher and Goddard [25] derived an analytical expression for the asymptotic variance of the heritability estimator in a range of pedigree- and marker-based experimental designs. Unfortunately, these conditions typically do not hold for genomic datasets, mainly due to the limited sample size, making either of these approximations ineffective [20]. Other approaches include hierarchical bootstrapping schemes, e.g., [26]; extending the REML estimation method with Bayesian priors, e.g., [27, 28]; using alternative statistics as a basis for building CIs [17, 29, 30]; or using Bayesian posterior distribution of the heritability value [31].

An alternative approach is the parametric bootstrap test inversion technique, which constructs CIs via sampling phenotypes, performing heritability estimation on the sampled phenotypes, estimating the distribution of the heritability estimator and using these estimates as a basis for CI construction [32]. The main advantage of using a parametric bootstrap approach is that it does not require

any assumptions on the distribution of the heritability estimator or of Bayesian priors. As a naïve implementation of this approach would be computationally prohibitive, the ALBI method [20] utilizes a highly accurate approximation that allows an efficient construction of accurate CIs. However, ALBI still requires a preprocessing step. Newer datasets (e.g. the UK Biobank [33]) may contain tens or hundreds of thousands of individuals, for which this step may require hours of computation time. In addition, the need for a preprocessing step can be an obstacle in the adoption of a better CI construction method.

In this paper, we introduce FIESTA (Fast confidence Intervals using Stochastic Approximation), which dramatically reduces the running time of CI construction by several orders of magnitude, e.g., to mere seconds for dataset with tens of thousands of individuals, compared to hours or days. The key ingredient of our approach is a CI construction algorithm from the field of stochastic approximation (for a review, see [34]). Originating in the work of Robbins and Monro [35], stochastic approximation algorithms are recursive update rules that can be used, among other things, to solve optimization problems or function inversion problems when the collected data is subject to noise. It has been shown [36] that stochastic approximation can be used to construct CIs for general families of parametric distributions, given the ability to randomly sample from them, and this is the approach we employ here. We validate FIESTA on two real datasets, the Northern Finland Birth Cohort (NFBC) dataset [37] and the Wellcome Trust Case Control Consortium 2 (WTCCC2) [38] dataset.

In addition to the significant speedup in time, FIESTA requires no preprocessing step beyond calculating the eigendecomposition of the kinship matrix, which is usually already performed as a part of heritability estimation. Finally, we show that FIESTA is even significantly faster than the analytical SE formulation. In summary, FIESTA can effectively be used extremely easily to rapidly generate accurate CIs for REML heritability estimates. FIESTA is available as part of the ALBI toolkit at <https://github.com/cozygene/albi>.

2 Results

2.1 A Faster Method for Calculating CIs for Heritability

CIs constructed from standard errors, which are based on the assumption of a normal distribution for the heritability estimators, were previously shown to be inaccurate [13, 16–20]. In this paper, we introduce FIESTA, a method that generates accurate CIs for h^2 , the true heritability value, given \hat{h}^2 , the restricted maximum likelihood (REML) estimator for h^2 (see Methods). FIESTA uses the principle of test inversion to construct accurate CIs, using a stochastic approximation method that directly estimates the CI boundaries. We review FIESTA below; for a full description, see Methods.

The methodology of test inversion can be described as follows. The estimator \hat{h}^2 is a function of the phenotype, which is a random variable whose distribution depends on h^2 , assuming a fixed kinship matrix. Therefore, \hat{h}^2 is distributed

differently for every value of h^2 . For each true value of h^2 , we select a subset of possible \hat{h}^2 values that has a sampling probability of $1-\alpha$, where \hat{h}^2 is distributed under the assumption of a true heritability value h^2 . We define this subset to be the acceptance region for that value of h^2 . The CI accompanying an estimate \hat{h}^2 is the interval containing all values of h^2 whose acceptance region includes \hat{h}^2 , namely, for which \hat{h}^2 does not imply the rejection of the null hypothesis that the true heritability value is h^2 , with a significance level of α .

It remains to define suitable acceptance regions. In the Methods section, we review our scheme for defining acceptance regions. A basic ingredient of our construction of acceptance regions is inverting certain quantile functions of the distribution of \hat{h}^2 , as a function of h^2 . For example, finding the inverse of a value H^2 of the 95%-quantile function is finding a heritability value h^2 for which $\Pr_{h^2}(\hat{h}^2 \leq H^2) = 0.95$, i.e., the probability to get an heritability estimate of H^2 or below is precisely 95%, when \hat{h}^2 is distributed with the heritability value h^2 .

Instead of carrying out this task by a full parametric bootstrap estimate of the distribution of the estimator, we employ a technique from the field of stochastic approximation to achieve the same results with a fraction of the computational cost. The modified Robbins-Monro procedure [39], described in the Methods section, is an iterative method that finds the inverse of the quantile function of a one-parameter distribution. It operates by iteratively (1) drawing a sample with a true heritability value equal to our current guess for the required inverse value, (2) comparing its estimated heritability to H^2 ; (3) updating our current guess accordingly, by moving in the right direction, with a step size that decreases with the number of iterations. An additional speedup is acquired by using a fast method to calculate the derivative of likelihood of the sample, and using the derivative to compare its estimated heritability to H^2 , instead of performing the full likelihood maximization.

We applied FIESTA to construct 95% CIs for the NFBC dataset [37] and the WTCCC2 dataset [38], as seen in Fig. 1. We then turned to verify the accuracy of these CIs, which can be measured as follows. Draw multiple phenotype vectors from the distribution assumed by the LMM with parameters that correspond to a true heritability value h^2 . From each such phenotype, construct a CI for its estimated heritability with a confidence level of, e.g., 95%. If the constructed CIs are accurate, then they should cover the true underlying h^2 95% of the time. Then, check the percentage of times in which the CI covered h^2 , as a function of h^2 . We measured the accuracy of FIESTA, with CIs designed to have a coverage of 95%. The results are shown in Fig. 2, demonstrating that FIESTA accurately achieves the desired confidence levels.

2.2 Benchmarks

We compared the speed of the stochastic approximation approach, implemented in FIESTA, with that of using the parametric bootstrap for estimating the distribution of heritability estimator. The latter was tested either as implemented naively by using either GCTA [15] and pylmm [40], or by using ALBI [20]. Both the stochastic approximation and parametric bootstrap approaches require the

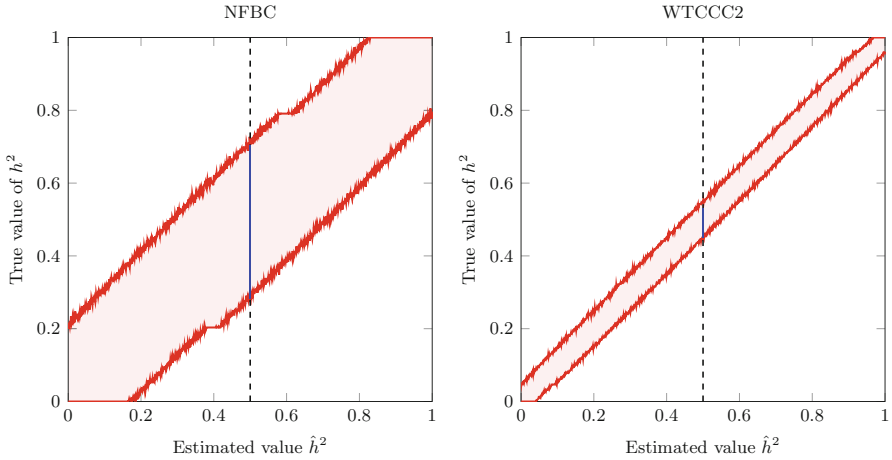


Fig. 1. 95% CIs for the NFBC and WTCCC2 datasets. Accurate 95% CIs constructed for the NFBC dataset [37] (left) and the WTCCC2 [38] dataset (right) by FIESTA. For each \hat{h}^2 on a fine grid of 1000 values (x axis), we constructed a CI, whose boundaries are shown (y axis). For example, for $\hat{h}^2 = 0.5$ (denoted by a dashed line), the CI for NFBC is $[0.282, 0.705]$ (denoted by a full line).

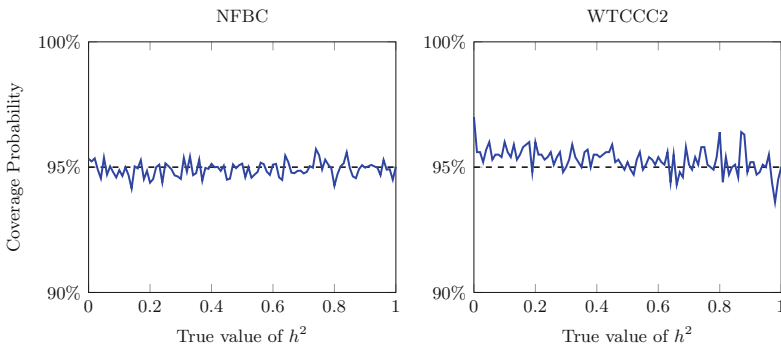


Fig. 2. Accuracy of CIs for the NFBC and WTCCC2 datasets. The coverage probabilities of the FIESTA CIs. The coverage probabilities are shown for CIs designed to have coverage probabilities of 95%. The CIs achieve accurate coverage.

calculation of the eigendecomposition of the kinship matrix. As this is already often a part of the heritability estimation algorithm, its calculation time is separated in the benchmarks. In the Discussion, we discuss how this step could be avoided altogether.

One difference between the two approaches is that the bootstrap approach performs a lengthy preprocessing step that estimates many distributions. Once these distributions are estimated, constructing a CI is very rapid. In contrast, the stochastic approximation approach does not perform a preprocessing step, but performs a non-trivial calculation per CI.

The construction of a single CI with FIESTA consists of calculating six to eight values using the modified Robbins-Monro procedure (see Methods). The first four values depend only on the kinship matrix, but not on the heritability estimate for which we construct a CI, so they need to be calculated only once per kinship matrix, and can then be shared between several CIs. Each modified Robbins-Monro run has the complexity of $O(nT)$, where n is the number of individuals in the sample and T is the number of iterations (in the order of 1,000; see Methods). Therefore, in total, the time complexity to construct K CIs with FIESTA grows linearly with K, T and n .

We also compared FIESTA to the performance of the analytical SE approach. While often inaccurate, analytical SEs are often the go-to method by many practitioners: First, their calculation is conceptually easy to understand, since a closed-form formula exists for the SEs (see Appendix); second, using a closed-form expression is often perceived as faster than more involved algorithmic procedures. However, this is not the case for heritability estimation, as SEs are calculated using variants of the Fisher information matrix (e.g., the AI matrix, as in GCTA [15]), whose calculation requires matrix-by-vector multiplications, which are $O(n^2)$. In contrast, FIESTA is linear in n , giving it an advantage at larger datasets in particular.

We performed a benchmark to evaluate FIESTA, using the NFBC and WTCCC2 datasets. We estimated the distributions of \hat{h}^2 for $h^2 = 0, 0.01, \dots, 1$, with GCTA [15] and pylmm [40], both of which perform full estimation, using 1,000 random bootstrap samples. For the same task, we also used ALBI [20], at a grid resolution of 0.001. The accuracy of CIs constructed according to the full estimation approach, as implemented in ALBI, are shown in the Appendix. As explained above, the time of construction of CIs given these distributions is negligible relative to the time required for the estimation of the distributions. We also constructed analytical SEs for both datasets using the AI method (see Appendix). These times are reported in Table 1.

As a comparison, we used FIESTA to construct varying number of CIs, using 1,000 iterations in the modified Robbins-Monro procedure (see Methods). In Table 1, it can be seen that FIESTA is significantly faster, particularly when few CIs are needed. We also note that FIESTA is currently implemented in the Python language, using the numpy package; a significant additional speedup can be obtained by migrating to a compiled language, e.g., C++.

We then continued to investigate the stability of CI construction and its dependency on the number of iterations. We ran FIESTA 100 times to construct CI for the NFBC and WTCCC2 datasets using 200, 500, 1,000 or 2,000 iterations. We measured the variance in the constructed CI endpoints (Table 2). As expected, the variance decreases with the number of iterations. In addition, we measured the mean and variance of the coverage of CIs under a grid of true heritability values. Here, also, we observed that variance of coverage decreases with the number of iterations. We note that 500 iterations are sufficient for reasonably accurate CIs for these datasets, and that the coverage of even 200 iterations is only slightly biased downwards.

Table 1. Benchmarks. Running times of FIESTA, compared with previous methods (see Results for more details). Running times are reported for the NFBC (2,520 individuals) and WTCCC2 (13,950 individuals) datasets.

Algorithm	Software	Time for NFBC	Time for WTCCC2
Eigen-decomp	GCTA	50 s	2 h
Full bootstrap	GCTA	> 30 days	> 30 days
Full bootstrap	pylmm	3.8 h	> 8 days
Full bootstrap	ALBI	5.35 min	2.5 h
Analytical SEs	n/a	~3.1 sec \times # of CIs, e.g.:	~6.2 min \times # of CIs, e.g.:
		1 CI, ~3 s	1 CI, ~6 min
		5 CIs, ~15 s	5 CIs, ~31 min
		10 CIs, ~31 s	10 CIs, ~1 h
		50 CIs, ~2.6 min	50 CIs, ~5 h
Stochastic approximation	FIESTA	~1.8 sec + 0.6 sec \times # of CIs, e.g.:	~6 sec + 2.8 sec \times # of CIs, e.g.:
		1 CI, ~3 s	1 CI, ~9 s
		5 CIs, ~6 s	5 CIs, ~20 s
		10 CIs, ~8 s	10 CIs, ~34 s
		50 CIs, ~33 s	50 CIs, ~2.4 min

Table 2. Stability of CI construction. 95% CIs for the NFBC and WTCCC2 datasets were constructed 100 times, with either 200, 500, 1,000 or 2,000 iterations. CIs were constructed for $\hat{h}^2 = 0, 0.001, \dots, 1$. In order to assess the variance of the construction process, the mean empirical standard error (SE) of the lower and upper endpoints is reported, where the mean was calculated over all non-constant endpoints, across all \hat{h}^2 values. In addition, the CI coverage for $h^2 = 0, 0.01, \dots, 1$ was calculated as in Fig. 2. The average mean and SE across all 100 runs, calculated across all h^2 , is reported.

Dataset	NFBC				WTCCC2			
	200	500	1,000	2,000	200	500	1,000	2,000
CI lower point SE	0.0201	0.0132	0.0094	0.0067	0.0050	0.0032	0.0023	0.0016
CI upper point SE	0.0206	0.0133	0.0096	0.0070	0.0050	0.0031	0.0023	0.0016
Mean coverage	94.20%	94.71%	94.87%	94.95%	94.720%	95.217%	95.323%	95.373%
SE of coverage	0.45%	0.34%	0.30%	0.28%	0.781%	0.575%	0.486%	0.442%

3 Methods

For clarity of presentation, we begin by defining the heritability under the LMM, and briefly reviewing stochastic approximation and its relevance to finding CIs. Finally, we introduce FIESTA, our improved method for faster construction of CIs for heritability.

3.1 The Linear Mixed Model and REML

We consider the following standard linear mixed model (see [41] for a detailed review). Let n be the number of individuals and m is the number of SNPs. Let \mathbf{y} be a $n \times 1$ vector of phenotype measurements for each individual. Let \mathbf{X} be a $n \times p$ matrix of p covariates (possibly including an intercept vector $\mathbf{1}_n$ as a first column, as well as other covariates such as sex, age, etc.). Let \mathbf{Z} be the $n \times m$ standardized genotype matrix, i.e., columns have zero mean and unit variance. Let $\boldsymbol{\beta}$ be a $p \times 1$ vector of fixed effects, \mathbf{s} a $m \times 1$ vector of random effects, and \mathbf{e} a $n \times 1$ vector of errors. Then, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{s} + \mathbf{e}$. We assume \mathbf{s} and \mathbf{e} are statistically independent and are distributed normally as $\mathbf{s} \sim \mathcal{N}(\mathbf{0}_m, \frac{1}{m}\sigma_g^2\mathbf{I}_m)$, $\mathbf{e} \sim \mathcal{N}(\mathbf{0}_n, \sigma_e^2\mathbf{I}_n)$. The fixed effects $\boldsymbol{\beta}$ and the coefficients σ_g^2 and σ_e^2 are the parameters of the model.

Define $\mathbf{K} = \frac{1}{m}\mathbf{Z}\mathbf{Z}^T$. Typically, \mathbf{K} is commonly called the kinship matrix, or the genetic relationship matrix. Under these conditions, it follows [14] that:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma_g^2\mathbf{K} + \sigma_e^2\mathbf{I}_n). \quad (1)$$

The narrow-sense heritability due to genotyped common SNPs is defined as the proportion of total variance explained by genetic factors [6]:

$$h^2 = \frac{\sigma_g^2}{\sigma_g^2 + \sigma_e^2}. \quad (2)$$

Defining $\sigma_p^2 = \sigma_g^2 + \sigma_e^2$, Eq. 1 becomes: $\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma_p^2\mathbf{V}_{h^2})$, where $\mathbf{V}_{h^2} = h^2\mathbf{K} + (1 - h^2)\mathbf{I}_n$.

The most common way of estimating h^2 is restricted maximum likelihood (REML) estimation. REML consists of maximizing the likelihood function associated with the projection of the phenotype onto the subspace orthogonal to that of the fixed effects of the model [42]. In [20], it is shown that the distribution of \hat{h}^2 depends only on h^2 , and is invariant under changes to σ_p^2 and $\boldsymbol{\beta}$. We may therefore limit our study to the \hat{h}^2 estimator alone, in the special case of fixed $\sigma_p^2 = 1$ and $\boldsymbol{\beta} = \mathbf{0}_p$, which substantially simplifies the problem; namely, we may focus on properties of the distribution $\mathcal{N}(\mathbf{0}_n, \mathbf{V}_{h^2})$ instead of the more general $\mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma_p^2\mathbf{V}_{h^2})$.

3.2 Confidence Intervals for h^2

We wish to build confidence intervals with a coverage probability of $1 - \alpha$ (e.g., 95%). The full derivation is developed in [20], and is reviewed in the Appendix; we cite the results here.

Let $c_\beta(h^2)$ be the β -th quantile function of \hat{h}^2 , when the true heritability is h^2 ; i.e. $\Pr_{h^2}(\hat{h}^2 \leq c_\beta(h^2)) = \beta$. Define s and t to be the values for which $\Pr_{h^2=s}(\hat{h}^2 = 0) = \alpha/2$ and $\Pr_{h^2=t}(\hat{h}^2 = 1) = \alpha/2$. In addition, let

$s^* = c_{1-\alpha}(0), t^* = c_\alpha(1)$. Then the lower and upper CI boundaries for an estimate H^2 are given, respectively, by

$$l_{H^2} = \begin{cases} 0 & \text{if } H^2 \leq s^* \\ c_{1-\alpha}^{-1}(H^2) & \text{if } c_{1-\alpha}^{-1}(H^2) < s \\ s & \text{if } s \in [c_{1-\alpha/2}^{-1}(H^2), c_{1-\alpha}^{-1}(H^2)] \\ c_{1-\alpha/2}^{-1}(H^2) & \text{if } s < c_{1-\alpha/2}^{-1}(H^2) \end{cases} \quad (3)$$

and

$$u_{H^2} = \begin{cases} c_{1-\alpha/2}^{-1}(H^2) & \text{if } c_{\alpha/2}^{-1}(H^2) < t \\ t & \text{if } t \in [c_\alpha^{-1}(H^2), c_{\alpha/2}^{-1}(H^2)] \\ c_\alpha^{-1}(H^2) & \text{if } t < c_\alpha^{-1}(H^2) \\ 1 & \text{if } t^* \leq H^2. \end{cases} \quad (4)$$

3.3 Using Stochastic Approximation to Calculate CIs

Robbins-Monro. Stochastic approximation methods are a family of iterative stochastic optimization algorithms that attempt to find zeroes, inverses or extrema of functions which cannot be computed directly, but only estimated via noisy observations. The classical Robbins-Monro algorithm presents a methodology for solving a function inversion problem, where the function is the expected value of a parametrized family of distributions. Namely, a function $g(\theta)$ is given, for which we want to find an inverse, i.e., a value θ for which $g(\theta) = C$, for some constant C . However, the function g is not directly available to us, but rather we are only able to obtain noisy observations from it. The Robbins-Monro procedure is a modification of Newton’s method, where the step sizes are instead an appropriately decreasing sequence. Starting with an initial guess, θ_0 , at iteration n we obtain a noisy sample y_n from a distribution whose mean is $g(\theta_n)$, and update our estimate with

$$\theta_{n+1} = \theta_n - \gamma_n \cdot (y_n - C) \quad (5)$$

where $\gamma_n = 1/n$. The Robbins-Monro procedure is shown to converge to the correct solution when: (i) the random variables defining our sampling process at each $g(\theta)$ are uniformly bounded; (ii) $g(\theta)$ is nondecreasing; and (iii) $g'(\bar{\theta})$ exists and is positive [35].

Using Robbins-Monro to calculate CIs. Garthwaite and Buckland [36] have used the Robbins-Monro process for finding the endpoints of CIs, as we will now describe. We discuss the case of one-sided CIs, but the application to two-sided CIs is immediate.

Suppose that $[0, u_{\hat{\theta}})$ is the one-sided $1 - \alpha$ CI for θ , when data \mathbf{y} has been observed, with an estimate $\hat{\theta} = \hat{\theta}(\mathbf{y})$. Then, the correct endpoint satisfies

$$\Pr_{\theta=u_{\hat{\theta}}}(\hat{\theta} \leq \hat{\theta}(\mathbf{y})) = \alpha \quad (6)$$

If we define $g(\theta) = \Pr_{\theta}(\hat{\theta} \geq \hat{\theta}(\mathbf{y}))$ (to make it nondecreasing), then finding $u_{\hat{\theta}}$ is equivalent to finding the inverse of g at $1 - \alpha$. However, under these settings, we do not have direct access to g . Rather, we sample a binary random variable Y_{θ} , indicating that a sample \mathbf{y}_{θ} randomly drawn from $g(\theta)$ has an estimate $\hat{\theta}(\mathbf{y}_{\theta})$ larger than $\hat{\theta}(\mathbf{y})$. By definition, $\Pr_{\theta}(Y_{\theta}) = \Pr_{\theta}(\hat{\theta}(\mathbf{y}_{\theta}) \geq \hat{\theta}(\mathbf{y})) = g(\theta)$, so the random sample Y_{θ} has a mean of $g(\theta)$. Effectively, this formulation allows us to use the Robbins-Monro procedure to invert the quantile function as a function of θ . Full asymptotic efficiency can be achieved by multiplying the step size γ_n by some constant c .

In detail, denote by y_n a random sample from the random variable Y_{θ_n} . The update rule is $\theta_{n+1} = \theta_n - c\gamma_n \cdot (y_n - (1 - \alpha))$, or explicitly:

$$\theta_{n+1} = \begin{cases} \theta_n - \frac{c\alpha}{n} & \text{if } y_n = 1 \\ \theta_n + \frac{c(1-\alpha)}{n} & \text{if } y_n = 0 \end{cases} \quad (7)$$

The procedure is shown to be fully asymptotic efficient if $c = 1/g'(u_{\theta})$. However, as neither g nor u_{θ} are known in advance, c is estimated adaptively, using the current estimate θ_n in place of u_{θ} , and assuming a parametric form for g [36].

The modified Robbins-Monro procedure. As mentioned above, if the optimal step size constant is known, this procedure is fully asymptotic efficient. However it was empirically shown to work poorly for extreme quantiles. Joseph [39] suggested a modification of this procedure, which is tuned to obtain optimal convergence speed. It uses the following update form:

$$\theta_{n+1} = \theta_n - a_n(y_n - C_n). \quad (8)$$

Joseph allows the use of a different target value, C_n , in each iteration, instead of the required constant, C . The step sizes a_n and target values C_n are derived explicitly in [39] to be optimal under a Bayesian analysis framework. As in [36], the optimal step size also uses $g'(u_{\theta})$, which is unknown, and a suitable approximation scheme is used. The modified Robbins-Monro procedure achieves significantly faster convergence rates in the case of the estimation of extreme quantiles.

3.4 Using the Modified Robbins-Monro Procedure to Obtain CIs for Heritability

We now describe how to rapidly construct CIs for heritability. As described above, the first step is to find s, t, s^* and t^* . To find s , we employ the modified Robbins-Monro procedure [39], where the parameter of interest is $\theta := h^2$, the function is $g(\theta) := \Pr_{h^2=\theta}(\hat{h}^2 = 0)$ and the inverse value we wish to find corresponds to $C = \alpha/2$. We note that we chose g here to be nonincreasing for the sake of clarity of presentation; to conform with the Robbins-Monro formulation, we would need to redefine $g \rightarrow 1 - g$ and $C \rightarrow 1 - C$. At a single iteration of the modified Robbins-Monro procedure, we have an estimate h_n^2 for s , and we need

to sample from a distribution whose mean is $\Pr_{h_n^2}(\hat{h}^2 = 0)$. To achieve that, we draw a sample from the distribution corresponding to h_n^2 , $\mathcal{N}(\mathbf{0}_n, \mathbf{V}_{h_n^2})$, and check if the maximum likelihood estimate for it is 0 (or above). This procedure can be done quickly in $O(n)$, as we now describe, circumventing the need to perform a full likelihood maximization for the sample.

As detailed above, we make repeated use of the following procedure: (1) Draw a random sample \mathbf{y} from the distribution corresponding to a given heritability value h^2 , $\mathcal{N}(\mathbf{0}_n, \mathbf{V}_{h^2})$; (2) Decide whether its heritability estimate, $\hat{h}^2(\mathbf{y})$, is larger than a given value, H^2 . In [20], it is shown that when $\mathbf{X} = \mathbf{1}_n$, these two steps may equivalently be performed by drawing a vector \mathbf{u} of i.i.d, standard normal variables $\mathbf{u} \sim \mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$, and checking if

$$\sum_{i=1}^n \xi_i^{h^2, H^2} u_i^2 > 0, \tag{9}$$

where

$$\xi_i^{h^2, H^2} = \frac{h^2(d_i - 1) + 1}{H^2(d_i - 1) + 1} \left(\frac{d_i - 1}{H^2(d_i - 1) + 1} - \frac{1}{n - 1} \sum_{j=1}^{n-1} \frac{d_j - 1}{H^2(d_j - 1) + 1} \right) \tag{10}$$

for $i = 1, \dots, n - 1$, and $\xi_n^{h^2, H^2} = 0$, with d_i being the eigenvalues of \mathbf{K} . The sign of the expression in Eq. (9) is equal to the sign of $\frac{\partial \ell_{REML}}{\partial h^2}(H^2)$, the derivative of ℓ_{REML} at the point H^2 . Therefore, assuming the restricted likelihood function is well behaved, a positive derivative indicates that the REML heritability estimate is larger than H^2 . Similar expressions are defined for a general \mathbf{X} in [20]. Once the eigendecomposition of \mathbf{K} is obtained, this procedure may be performed in a time complexity linear in n .

Similarly, for finding s^* , we define the function $g(\theta) := \Pr_{h^2=0}(\hat{h}^2 \leq \theta)$, for which we want to find the inverse of $C = 1 - \alpha$. The procedures for finding t and t^* are similar.

The second step involves calculating the quantities $c_{\alpha/2}^{-1}(H^2), c_{\alpha}^{-1}(H^2), c_{1-\alpha}^{-1}(H^2)$ and $c_{1-\alpha/2}^{-1}(H^2)$ as required. This can again be done by the modified Robbins-Monro procedure, by setting $\theta := h^2, g(\theta) := \Pr_{h^2}(\hat{h}^2 \leq H^2)$, and $C = \alpha/2, \alpha, 1 - \alpha/2$ or $1 - \alpha$. To sample from a distribution whose mean is $\Pr_{h_n^2}(\hat{h}^2 \leq H^2)$, we draw a sample from the distribution corresponding to h_n^2 , and check if the maximum likelihood estimate for it is above H^2 . Again, this procedure can be done quickly in $O(n)$. Once these quantities have been calculated, the CI can be calculated as detailed in Eqs. (3) and (4).

In practice, we used the following choices in the modified Robbins-Monro procedure: (i) We used $T = 1000$ iterations; (ii) we set the prior standard deviation to $\tau = 0.4$, used to derive a_n and C_n via the Bayesian analysis (see [39]); (iii) we used the midpoint between the estimate and relevant boundary (0 or 1, depending on the quantile required) as a starting point; (iv) we adaptively changed the step size constant, following the suggestion of Garthwaite and Buckland,

by approximating the derivative with an expression proportional to the distance from $\hat{\theta}$:

$$g'(u_\theta) \approx k(h_n^2 - H^2), \quad k = \frac{2}{z_\beta \cdot (2\pi)^{-1/2} \cdot e^{-z_\beta^2/2}} \quad (11)$$

where z is the quantile function of the normal distribution, and β is the required quantile.

3.5 The NFBC Dataset

We analyzed 5,236 individuals from the Northern Finland Birth Cohort (NFBC) dataset, which consists of genotypes at 331,476 genotyped SNPs and 10 phenotypes [37]. From each pair of individuals with relatedness of more than 0.025, one was reserved, resulting in 2,520 individuals.

3.6 The WTCCC2 Dataset

We analyzed the Wellcome Trust Case Control Consortium 2 dataset [38]. In the multiple sclerosis (MS) and ulcerative colitis (UC) datasets, we used the same data processing described in [43] to ensure consistency. Briefly, UK controls and cases from both UK and non-UK were used. SNPs were removed with $> 0.5\%$ missing data, $p < 0.01$ for allele frequency difference between two control groups, $p < 0.05$ for deviation from Hardy-Weinberg equilibrium, $p < 0.05$ for differential missingness between cases and controls, or minor allele frequency $< 1\%$. In all analyses, SNPs within 5M base pairs of the human leukocyte antigen (HLA) region were excluded, because they have large effect sizes and highly unusual linkage disequilibrium patterns, which can bias or exaggerate the results. Finally, from each pair of individuals with relatedness of more than 0.025, one was reserved, resulting in 13,950 individuals.

4 Discussion

We have presented FIESTA, an efficient method for constructing accurate CIs using stochastic approximation. We have shown that FIESTA is very fast, while achieving exact coverage due to the fact that it does not rely on any assumptions of the distribution of the estimator. FIESTA is also faster than the analytical approximation used by SEs. Due to its speed, FIESTA can be easily used for datasets with tens or hundreds of thousands of individuals.

FIESTA requires the eigendecomposition of the kinship matrix, whose computational complexity is cubic in the number of individuals. While this is often a preliminary step in heritability estimation, it may be computationally prohibitive for larger datasets. Recent methods for heritability estimation (see [44]) utilize conjugate gradient methods to avoid cubic steps altogether. One direction of extension for FIESTA is devising a procedure to calculate the derivative of the restricted likelihood function using conjugate gradient methods, which are quadratic, but do not require the eigendecomposition.

We note that the confidence intervals constructed by FIESTA are estimated under a set of assumptions, particularly that the data is generated from the linear mixed model as described in the Methods. Deviations from these assumptions could result in inaccurate confidence intervals. Specifically, we observed that when the genotype matrix is of low rank (e.g., in the case where duplicates are introduced), then the confidence intervals calculated by FIESTA may be inaccurate. We therefore recommend removing duplicates and closely related individuals from the data prior to the application of FIESTA.

A common extension of the LMM is that of multiple variance components, where the genome is divided into distinct partitions (e.g., according to functional annotations, or by chromosomes), and the relative genetic contribution of each partition is estimated instead. Another extension is that of multiple traits, where several phenotypes are estimated concurrently, allowing dependencies between them. In principle, the methodology behind FIESTA can be applied to the multiparametric case as well. However, there are several computational and conceptual hurdles that make this application highly nontrivial. First, a major difficulty rises from the fact that it is no longer necessarily possible to jointly diagonalize several kinship matrices. Thus, the computation of the derivatives of the logarithm of the restricted likelihood functions can no longer utilize the eigendecomposition. Second, the inversion of acceptance regions of multiple parameters results in confidence regions of more than one dimension. While these have the required coverage probability, their shape may be difficult to report or to interpret easily (e.g. an ellipsoid). For example, hyper-rectangular confidence regions are often desirable [45], as the marginal CI of each parameter has the same coverage probability as the confidence region. Therefore, multiparametric extensions remain a future direction of research.

Acknowledgements. The authors would like to thank David Steinberg. R.S. is supported by the Colton Family Foundation. This study was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University to R.S. The Northern Finland Birth Cohort data were obtained from dbGaP: phs000276.v2.p1. This study makes use of data generated by the Wellcome Trust Case Control Consortium. A full list of the investigators who contributed to the generation of the data is available from www.wtccc.org.uk. Funding for the project was provided by the Wellcome Trust under award 076113.

Appendix

The supplementary material, including additional figures, are located at <https://github.com/cozygene/albi>.

References

1. Fisher, R.A.: The correlation between relatives on the supposition of mendelian inheritance. *Trans. R. Soc. Edinb.* **52**, 399–433 (1918)
2. Silventoinen, K., Sammalisto, S., Perola, M., Boomsma, D.I., Cornes, B.K., Davis, C., Dunkel, L., De Lange, M., Harris, J.R., Hjelmberg, J.V., et al.: Heritability of adult body height: a comparative study of twin cohorts in eight countries. *Twin Res.* **6**(05), 399–408 (2003)
3. Macgregor, S., Cornes, B.K., Martin, N.G., Visscher, P.M.: Bias, precision and heritability of self-reported and clinically measured height in Australian twins. *Hum. Genet.* **120**(4), 571–580 (2006)
4. Manolio, T.A., Brooks, L.D., Collins, F.S.: A hapmap harvest of insights into the genetics of common disease. *J. Clin. Invest.* **118**(5), 1590 (2008)
5. Welter, D., MacArthur, J., Morales, J., Burdett, T., Hall, P., Junkins, H., Klemm, A., Flicek, P., Manolio, T., Hindorf, L., Parkinson, H.: The NHGRI GWAS catalog, a curated resource of SNP-trait associations. *Nucleic Acids Res.* **42**(Database issue), D1001–D1006 (2014)
6. Visscher, P.M., Hill, W.G., Wray, N.R.: Heritability in the genomics era—concepts and misconceptions. *Nat. Rev. Genet.* **9**(4), 255–266 (2008)
7. Kang, H.M., Zaitlen, N.A., Wade, C.M., Kirby, A., Heckerman, D., Daly, M.J., Eskin, E.: Efficient control of population structure in model organism association mapping. *Genetics* **178**(3), 1709–1723 (2008)
8. Kang, H.M., Sul, J.H., Service, S.K., Zaitlen, N.A., Kong, S.Y.Y., Freimer, N.B., Sabatti, C., Eskin, E.: Variance component model to account for sample structure in genome-wide association studies. *Nat. Genet.* **42**(4), 348–354 (2010)
9. Lippert, C., Listgarten, J., Liu, Y., Kadie, C.M., Davidson, R.I., Heckerman, D.: Fast linear mixed models for genome-wide association studies. *Nat. Methods* **8**(10), 833–835 (2011)
10. Zhou, X., Stephens, M.: Genome-wide efficient mixed-model analysis for association studies. *Nat. Genet.* **44**(7), 821–824 (2012)
11. Vattikuti, S., Guo, J., Chow, C.C.: Heritability and genetic correlations explained by common SNPs for metabolic syndrome traits. *PLoS Genet.* **8**(3), e1002637 (2012)
12. Wright, F.A., Sullivan, P.F., Brooks, A.I., Zou, F., Sun, W., Xia, K., Madar, V., Jansen, R., Chung, W., Zhou, Y.H., Abdellaoui, A., Batista, S., Butler, C., Chen, G., Chen, T.H., D’Ambrosio, D., Gallins, P., Ha, M.J., Hottenga, J.J., Huang, S., Kattenberg, M., Kocher, J., Middeldorp, C.M., Qu, A., Shabalin, A., Tischfield, J., Todd, L., Tzeng, J.Y., van Grootheest, G., Vink, J.M., Wang, Q., Wang, W., Wang, W., Willemsen, G., Smit, J.H., de Geus, E.J., Yin, Z., Penninx, B., Boomsma, D.I.: Heritability and genomics of gene expression in peripheral blood. *Nat. Genet.* **46**(5), 430–437 (2014)
13. Kruijer, W., Boer, M.P., Malosetti, M., Flood, P.J., Engel, B., Kooke, R., Keurentjes, J.J., van Eeuwijk, F.A.: Marker-based estimation of heritability in immortal populations. *Genetics* **199**(2), 379–398 (2015)
14. Yang, J., Benyamin, B., McEvoy, B.P., Gordon, S., Henders, A.K., Nyholt, D.R., Madden, P.A., Heath, A.C., Martin, N.G., Montgomery, G.W., Goddard, M.E., Visscher, P.M.: Common SNPs explain a large proportion of the heritability for human height. *Nat. Genet.* **42**(7), 565–569 (2010)
15. Yang, J., Lee, S.H., Goddard, M.E., Visscher, P.M.: GCTA: a tool for genome-wide complex trait analysis. *Am. J. Hum. Genet.* **88**(1), 76–82 (2011)

16. Lohr, S.L., Divan, M.: Comparison of confidence intervals for variance components with unbalanced data. *J. Stat. Comput. Simul.* **58**(1), 83–97 (1997)
17. Burch, B.D.: Comparing pivotal and REML-based confidence intervals for heritability. *J. Agric. Biol. Environ. Stat.* **12**(4), 470–484 (2007)
18. Burch, B.D.: Assessing the performance of normal-based and REML-based confidence intervals for the intraclass correlation coefficient. *Comput. Stat. Data Anal.* **55**(2), 1018–1028 (2011)
19. Kraemer, K.: Confidence intervals for variance components and functions of variance components in the random effects model under non-normality (2012)
20. Schweiger, R., Kaufman, S., Laaksonen, R., Kleber, M.E., März, W., Eskin, E., Rosset, S., Halperin, E.: Fast and accurate construction of confidence intervals for heritability. *Am. J. Hum. Genet.* **98**(6), 1181–1192 (2016)
21. Chernoff, H.: On the distribution of the likelihood ratio. *Ann. Math. Stat.* 573–578 (1954)
22. Moran, P.A.: Maximum-likelihood estimation in non-standard conditions. In: *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 70, pp. 441–450. Cambridge University Press (1971)
23. Self, S.G., Liang, K.Y.: Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *J. Am. Stat. Assoc.* **82**(398), 605–610 (1987)
24. Stern, S., Welsh, A.: Likelihood inference for small variance components. *Can. J. Stat.* **28**(3), 517–532 (2000)
25. Visscher, P.M., Goddard, M.E.: A general unified framework to assess the sampling variance of heritability estimates using pedigree or marker-based relationships. *Genetics* **199**(1), 223–232 (2015)
26. Thai, H.T., Mentré, F., Holford, N.H.G., Veyrat-Follet, C., Comets, E.: A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. *Pharm. Stat.* **12**(3), 129–140 (2013)
27. Wolfinger, R.D., Kass, R.E.: Nonconjugate Bayesian analysis of variance component models. *Biometrics* **56**(3), 768–774 (2000)
28. Chung, Y., Rabe-hesketh, S., Gelman, A., Dorie, V., Liu, J.: Avoiding boundary estimates in linear mixed models through weakly informative priors. *Berkeley Preprints*, pp. 1–3 (2011)
29. Harville, D.A., Fenech, A.P.: Confidence intervals for a variance ratio, or for heritability, in an unbalanced mixed linear model. *Biometrics* 137–152 (1985)
30. Burch, B.D., Iyer, H.K.: Exact confidence intervals for a variance ratio (or heritability) in a mixed linear model. *Biometrics* 1318–1333 (1997)
31. Furlotte, N.A., Heckerman, D., Lippert, C.: Quantifying the uncertainty in heritability. *J. Hum. Genet.* **59**(5), 269–275 (2014)
32. Carpenter, J., Bithell, J.: Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat. Med.* **19**(9), 1141–1164 (2000)
33. Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., et al.: Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS Med.* **12**(3), e1001779 (2015)
34. Kushner, H., Yin, G.G.: *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35. Springer Science & Business Media, New York (2003)
35. Robbins, H., Monro, S.: A stochastic approximation method. *Ann. Math. Stat.* 400–407 (1951)
36. Garthwaite, P.H., Buckland, S.T.: Generating monte carlo confidence intervals by the robbins-monro process. *Appl. Stat.* 159–171 (1992)

37. Sabatti, C., Service, S.K., Hartikainen, A.L.L., Pouta, A., Ripatti, S., Brodsky, J., Jones, C.G., Zaitlen, N.A., Varilo, T., Kaakinen, M., Sovio, U., Ruukonen, A., Laitinen, J., Jakkula, E., Coin, L., Hoggart, C., Collins, A., Turunen, H., Gabriel, S., Elliot, P., McCarthy, M.L., Daly, M.J., Järvelin, M.R.R., Freimer, N.B., Peltonen, L.: Genome-wide association analysis of metabolic traits in a birth cohort from a founder population. *Nat. Genet.* **41**(1), 35–46 (2009)
38. Sawcer, S., Hellenthal, G., Pirinen, M., Spencer, C.C., Patsopoulos, N.A., Moutsianas, L., Dilthey, A., Su, Z., Freeman, C., Hunt, S.E., et al.: Genetic risk and a primary role for cell-mediated immune mechanisms in multiple sclerosis. *Nature* **476**(7359), 214 (2011)
39. Joseph, V.R.: Efficient Robbins-Monro procedure for binary data. *Biometrika* **91**(2), 461–470 (2004)
40. Furlotte, N.A., Eskin, E.: Efficient multiple trait association and estimation of genetic correlation using the matrix-variate linear mixed-model. *Genetics* **200**(1), 59–68 (2015)
41. Searle, S.R., Casella, G., McCulloch, C.E.: *Variance Components*, vol. 391. Wiley, Hoboken (2009)
42. Patterson, H.D., Thompson, R.: Recovery of inter-block information when block sizes are unequal. *Biometrika* **58**(3), 545–554 (1971)
43. Yang, J., Zaitlen, N.A., Goddard, M.E., Visscher, P.M., Price, A.L.: Advantages and pitfalls in the application of mixed-model association methods. *Nat. Genet.* **46**(2), 100–106 (2014)
44. Loh, P.R., Bhatia, G., Gusev, A., Finucane, H.K., Bulik-Sullivan, B.K., Pollack, S.J., de Candia, T.R., Lee, S.H., Wray, N.R., Kendler, K.S., O'Donovan, M.C., Neale, B.M., Patterson, N., Price, A.L.: Contrasting genetic architectures of schizophrenia and other complex diseases using fast variance-components analysis. *Nat. Genet.* **47**(12), 1385–1392 (2015)
45. Sidak, Z.: Rectangular confidence regions for the means of multivariate normal distributions. *J. Am. Stat. Assoc.* **62**(318), 626–633 (1967)
46. Wasserman, L.: *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, New York (2013)
47. Gilmour, A.R., Thompson, R., Cullis, B.R.: Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics* 1440–1450 (1995)

Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees

Brad Solomon¹ and Carl Kingsford²(✉)

¹ Computational Biology Department, School of Computer Science, Joint Carnegie Mellon University–University of Pittsburgh Ph.D. Program in Computational Biology, Carnegie Mellon University, Pittsburgh, PA, USA

² Computational Biology Department, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, USA
carlk@cs.cmu.edu

Abstract. Enormous databases of short-read RNA-seq sequencing experiments such as the NIH Sequencing Read Archive (SRA) are now available. These databases could answer many questions about the condition-specific expression or population variation, and this resource is only going to grow over time. However, these collections remain difficult to use due to the inability to search for a particular expressed sequence. While some progress has been made on this problem, it is still not feasible to search collections of hundreds of terabytes of short-read sequencing experiments. We introduce an indexing scheme called Split Sequence Bloom Tree (SSBT) to support sequence-based querying of terabyte-scale collections of thousands of short-read sequencing experiments. SSBT is an improvement over the SBT [1] data structure for the same task. We apply SSBT to the problem of finding conditions under which query transcripts are expressed. Our experiments are conducted on a set of 2,652 publicly available RNA-seq experiments contained in the NIH for the breast, blood, and brain tissues. We demonstrate that this SSBT index can be queried for a 1000 nt sequence in under 4 min using a single thread and can be stored in just 39 GB, a five-fold improvement in search and storage costs compared to SBT.

1 Introduction

An enormous amount of DNA and RNA short read sequence data has been published world wide. The NIH Sequence Read Archive (SRA) [2] alone contains almost four petabases of open-access sequence and continues to grow at an accelerating rate. This collection could be a great resource for understanding genetic variation, and condition- and disease-specific gene function in ways the original depositors of the data did not anticipate. For example, a natural use would be to search all public, human RNA-seq short-read files in the SRA (representing individual sequencing runs) for the presence of a particular transcript of interest to understand conditions of expression or develop a manageable subset for further analysis. However, searching the entirety of such a database for a query sequence has not been possible in reasonable computational time.

Some progress has been made toward enabling sequence search on large databases. The NIH SRA does provide a sequence search functionality [3]; however, it requires the selection of a small number of experiments to which to restrict the search. Existing full-text indexing data structures such as Burrows-Wheeler transform [4], FM-index [5], or others [6–8] cannot at present be efficiently built at this scale. Word-based indices [9, 10], such as those used by Internet search engines, are not appropriate for edit-distance-based biological sequence search. The sequence-specific solutions caBLAST and its variants [11–13] require an index of a known genomes, genes, or proteins and so cannot search for novel sequences in unassembled read sets. Further, all of these existing approaches do not handle the additional complication that a match to a query sequence q may span many short reads.

More recently, two methods have been developed that store the kmer content of a set of experiments in a directly searchable index. The Sequence Bloom Tree [1] (SBT) encodes an approximation of each experiment’s kmer content in a single bloom filter and builds a directly searchable binary tree of bloom filters over increasingly large subsets of the data. Queries are processed by looking up each kmer in a query for their presence or absence in the tree and recursing until all matching leaves have been found. It represents the current best method for searching a large database but cannot handle petabase-scale data. The Bloom Filter Trie [14] (BFT) was designed as a direct compression method for a pan-genome and provides an exact index of kmer content which can be queried.

We address the search problem of finding all the experiments that contain enough reads matching a given query sequence q to indicate that q was present in the experiment. A query is an arbitrary sequence such as a transcript, and we find the collection of short-read experiments in which that sequence is present. These estimates themselves could be used to analyze conditions of gene expression or could make downstream analysis more efficient by filtering a large database for the relevant files. The Sequence Bloom Tree was the first data structure to directly address this problem and could search a 5 terabase dataset in under 20 min using a 200 GB index. We modify the base structure of SBT with a new indexing data structure called Split Sequence Bloom Tree (SSBT). SSBT “splits” the bloom filters present in an SBT into two distinct filters which store unique subsets of the base filter as described in Sect. 2.1. In addition to this novel storage strategy, the SSBT method introduces the concept of a ‘non-informative bit’ (Sect. 2.3) and uses a more efficient query algorithm which can prune query indices when universal matches or mismatches are found in a subtree (Sect. 2.5). These novel elements are the basis of the space and time improvements described below.

SSBT also extends a number of important properties found in SBT. Like SBT, SSBT is independent of the eventual queries, so the approach is not limited to searching only for known genes, but can potentially identify arbitrary sequences. SSBTs can be efficiently built, extended, and stored in limited space and do not require retaining the original sequence files. Using SSBT, datasets can be searched using low memory for the existence of arbitrary query sequences. We compared SSBT against BFT and SBT and found that it outperforms both in terms of query time (5 times faster than SBT and 15 times faster than BFT)

and storage cost, at the price of some additional time and temporary storage needed to construct the index.

2 Methods

2.1 Split Sequence Bloom Tree

The main idea behind SSBT is the creation of a hierarchy of compressed bloom filters [15, 16], which is used to efficiently store a set of experiments each consisting of many short reads (Fig. 1). A bloom filter is a probabilistic storage structure which encodes an arbitrary set into a fixed length bit vector using hash functions. As in the SBT, each bloom filter here encodes the set of k -mers (length- k subsequences) present within a subset of the sequencing experiments and is stored using hash functions that convert these kmers to a specific index on the filter. We define each experiments' individual kmer content as the set b_i , with the collection $B = \{b_i \mid 0 \leq i < n\}$ denoting a set of n experiments represented by their kmer content. Throughout, we abuse notation slightly to identify bloom filters with the sets they represent.

An SSBT is a binary tree that stores each b_i in B across a unique path from root to leaf with each leaf mapping to a single experiment. This is a change from

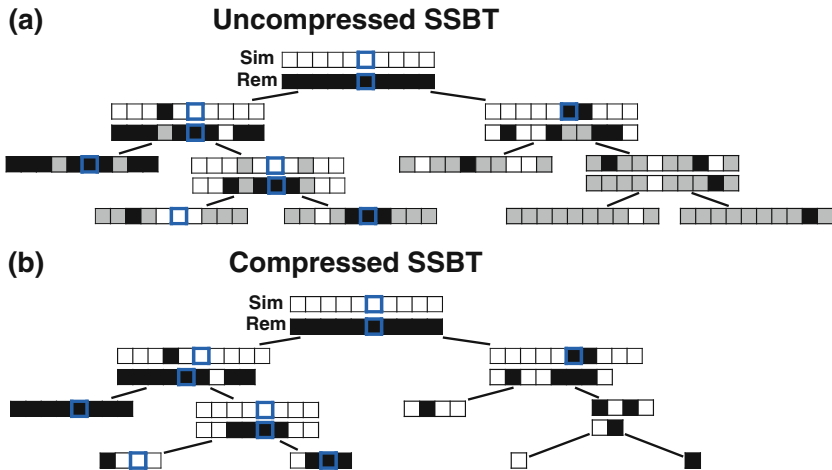


Fig. 1. An example uncompressed and compressed SSBT where black corresponds to a bit value of ‘1’ and white corresponds to a bit value of ‘0’. (a) Grey bits correspond to non-informative bits whose value is known (always zero) given a parent filter. We see that grey bits are cumulative and exist at all index positions below a on ‘1’ in the sim filter or a ‘0’ in the rem filter. When looking up index value 6, each filter is queried until either a sim ‘1’ is found or a rem ‘0’ is found. (b) All non-informative bits have been removed from the uncompressed tree (RRR does not change the bit values and is not represented in the figure). The lookup for index value 6 is adjusted based on the removed non-informative bits.

the SBT, which stores each b_i in B both as a unique leaf and in each node from root to leaf. The root node r of an SSBT contains the total content of each b_i and stores this information using two identically sized bloom filters with a single conserved hash; we define the pair of bloom filters at a node as a single ‘split bloom filter’. We define the first filter as the *similarity filter* $r_{sim} = \bigcap_{i=0}^{n-1} b_i$ as the bloom filter that stores the universally expressed kmers in set B . The second filter is defined as the *remainder filter* $r_{rem} = \bigcup_{i=0}^{n-1} (b_i - r_{sim})$ as the bloom filter which stores the remaining kmers — those in the union of all b_i excluding those found in the similarity vector. By this definition, the bitwise union of r_{sim} and r_{rem} is equivalent to a single fixed length bloom filter which stores all b_i in B . Furthermore, the bitwise intersection of r_{sim} and r_{rem} is the nullset. Different kmers in B may hash to the same position in r_{sim} . As r_{rem} is defined by r_{sim} , additional ‘similarity’ may be found by random chance when hashing kmers. The inverse (identical kmers mapping to different positions) is not possible in a bloom filter.

Given the root r , only the kmers which are stored in r_{rem} are then passed to the children. In this way r ’s immediate children do not store the full set b_i but rather the modified $b'_i = (b_i - r_{sim})$. This sparsification continues from root to leaf with $b''_i = (b'_i - r'_{sim})$ and more generally takes the form of a recurrence relationship $b_i^{(d)'} = b_i^{(d-1)'} - r_{sim}^{(d-1)'}$ for the depth d node on an arbitrary path from root to leaf. A leaf at depth d stores $b_i^{(d)'}$ in its sim filter. The leaf’s rem filter is defined as $b_i^{(d)'}$ or the nullset. We can recover the original set by following the unique path from root to leaf and find $b_i = \bigcup_{j=0}^d r_{sim}^{(j)'}$.

2.2 SSBT Construction and Insertion

A SSBT inherits the same general build process as an SBT and is built by repeated insertion of sequencing experiments followed by the removal of all non-informative bit indices from each filter. Given a (possibly empty) SSBT T , a new sequencing experiment s can be inserted into T by first computing the fixed-length bloom filter b_s of the kmers present in s and then walking from the root along a path to the leaves and inserting s at the bottom of T in the following way. When at node u , if u has children, b_s has to be split between u_{sim} and u_{rem} . This is done through the bit updates defined by Table 1 for each bit index i in $0 \leq i < |b_s|$. These updates ensure that u correctly stores what is still universally similar in u_{sim} and what now exists below u in the tree with u_{rem} and that b_s has been updated to store similar elements at u .

The potentially modified b_s is then compared against each child c of u in order to find the ‘‘most similar’’ child. This greedy insertion strategy attempts to maximize the similarity in a branch in order to produce the smallest SSBT with the most efficient branch pruning. While we tested many similarity metrics for comparing a single filter $b(s)$ with a split filter (c_{sim}, c_{rem}) , the metric used in the experiments reported below first computes the total number of 1-bits shared between the c_{sim} and b_s , and chooses the child with the largest number of such shared 1-bits. If no child has any 1-bits in common with b_s , the metric then selects the child with the smallest Hamming distance between b_s and $c_{rem}(s)$.

Table 1. Bit update table when inserting $b(s)$ into the subtree rooted at u . u 's two immediate children are both updated with the single column c_{sim} . A value of '-' implies that no change needs to be made to that bit. (i) $b(s)$ contains a value already stored in u_{sim} , the value is removed from $b(s)$. (ii) $b(s)$ does not contain a value stored in u_{sim} . Although the bit is no longer similar at u , $b(s)$ has not yet been inserted into a child and all current children of u are universally similar at this location. (v) $b(s)$ contains a value not found in the tree. u_{rem} is updated to reflect that the value now exists. (iii, iv, vi) No changes need to be made.

	Before			After			
	u_{sim}	u_{rem}	$b(s)$	u_{sim}	u_{rem}	$b(s)$	c_{sim}
(i)	1	0	1	-	-	0	-
(ii)	1	0	0	0	1	-	1
(iii)	0	1	1	-	-	-	-
(iv)	0	1	0	-	-	-	-
(v)	0	0	1	-	1	-	-
(vi)	0	0	0	-	-	-	-

Once the most similar child has been found, this child then becomes the new current node, and the process is repeated with a new subtree. If u has no children, then u represents a sequencing experiment s' and only contains one vector u_{sim} . In this case, a new node v is created as a child of u 's parent with u and b_s as v 's children. As both $u = u_{sim}$ and b_s are leaves, v_{sim} is the bit-intersection of u and b_s while v_{rem} is the bit-union of $u - v_{sim}$ and $b_s - v_{sim}$. Finally, we remove the elements in this new parent node from both children by replacing u with $u - v_{sim}$ and b_s with $b_s - v_{sim}$. This yields an uncompressed SSBT containing all previous nodes and two new nodes v and b_s .

2.3 Bit Non-informativity in SSBT

Given the definition of SSBT's split filters described above, for an arbitrary node u the only values allowed at a particular index i are $(u_{sim}[i], u_{rem}[i]) = (1, 0), (0, 1), (0, 0)$. However, every index is represented using a bit from either filter, even when one filter's value clearly defines the other. We address this inefficiency by removing these "non-informative bits" from the tree. Here we define a non-informative bit as a bit index i present in node u whose value can be determined by examining the bloom filters present in the set of parent filters above u . We describe the cases of non-informativity in SSBT below and provide an example in Fig. 1:

1. For a bit index i , if $u_{rem}[i] = 0$, then for every descendant c of u , $c_{sim}[i] = c_{rem}[i] = 0$ and i is non-informative below u . This is a direct result of the definition of u_{rem} as the union of all children below it.
2. For a bit index i , if $u_{sim}[i] = 1$ then $u_{rem}[i] = 0$ and $u_{rem}[i]$ is non-informative. This is a formalization of the fact that the rem filter only contains the elements which are not stored in the sim filter.

3. For a bit index i , if $u_{sim}[i] = 1$ then for every descendent c of u , $c_{sim}[i] = c_{rem}[i] = 0$ and i is non-informative below u . This is the logical extension of applying (1) to (2).

Using these cases, we can remove all non-informative bits from u_{rem} given u_{sim} and we can remove all non-informative bits from u 's immediate children using both u_{sim} and u_{rem} . As bits are only ever removed, for a node u and its child c , $|c_{rem}| \leq |c_{sim}| \leq |u_{rem}| \leq |u_{sim}|$. These removed bits lead to size reductions for all subsequent filters. SSBTs are most efficient when there is a large amount of uniformity in the experiments being stored at u either in terms of uniform expression of kmers or uniform absence of any kmer hashing to a particular bit.

2.4 SSBT Compression

Given an uncompressed SSBT T with bloom filters of fixed length m and conserved hash function, we build a compressed SSBT T' by removing all non-informative bits (defined in Sect. 2.3) from T and compressing the variable length filters using RRR-compression [17]. We handle the ‘removal’ of a bit in node u by constructing an intermediate node u' of size equal to the informative bits remaining and copying only these informative bits from u to u' . The total number of informative bits in u_{sim} and u_{rem} , as well as their exact indices, can be determined using two filters: u 's immediate parent's rem filter, $p(u)_{rem}$, and u_{sim} . We define $rank_x(u)[j]$ to be the number of bits set to x in the bit vector u from index $0 \leq i < j$. We compute this value in $O(\log \log d)$ time, with d defined by the size of the vector, $b = \lceil \log(d) \rceil$, by including a 64 bit overhead per 512 bit superblock in our raw bit vectors. This data structure was first proposed as part of the 64-bit rank9 function [18], and we used the implementation found in the C++ package sdsl [19]. For vectors of size 2^{256} bits or less, this can be considered a near-constant time method.

Given $p(u)_{rem}$, the only informative bits in u_{sim} are those i for which $p(u)_{rem}[i] = 1$ and $|u'_{sim}| = m - rank_0(p(u)_{rem})[m] = rank_1(p(u)_{rem})[m]$. Likewise given u_{sim} , the only informative bits in u_{rem} are those i in which both $u_{sim}[i] = 0$ and $p(u)_{rem}[i] = 1$ and $|u'_{rem}| = m - rank_0(p(u)_{rem})[m] - rank_1(u_{sim})[m]$. At each i , the values in $p(u)_{rem}[i]$ and $u_{sim}[i]$ determine if i is informative. If i is informative, it is copied over to the next open position in u'_{sim} and/or u'_{rem} . Subsequently, u' is compressed via the RRR [17] compression scheme, which allows querying a bit without decompression and incurs only a $O(\log m)$ factor increase in access time (where m is the size of the bloom filter with non-informative bits removed). This process operates for every node in the tree, starting with the root node T which has a full length T'_{sim} as it has no parent. See Fig. 1 for an example of the compression step.

2.5 SSBT Querying

Given a query sequence q and a compressed Split Sequence Bloom Tree T , the sequencing experiments that contain q can be found by breaking q into its constituent set of kmers K_q and then flowing these kmers over T starting from the

root. In an SSBT, these kmers are organized into a set of unique kmers and immediately converted into a vector of filter indices $V_q = Hash(K_q)$ using the conserved hash functions defined by T 's root's sim filter. At the root node u , we query first u_{sim} for each index in V_q . Matches in u_{sim} are recorded as 'universal hits' since they are found in intersection of all experiments rooted beneath u . The count of all universal hits represents a lower bound on the number of matching kmers in all experiments rooted below u . Indices that are universal hits do not have to be queried further and are removed from the set — the sum of these hits records their presence at all children to u . The remaining indices that were not found in u_{sim} are then queried in u_{rem} . As $|u_{sim}| \geq |u_{rem}|$, this query is accomplished by adjusting all indices $V_i \in V_q$ to account for the non-informative bits removed. We transition from u_{sim} to u_{rem} by subtracting the number of 1-bits which occurred before V_i in u_{sim} . This is simply the $rank_1(u_{sim}, V_i)$, a property of a bit vector which can be computed in constant time using RRR-compressed vectors.

Each modified index can then be queried in u_{rem} . Indices that map to 0-bits do not have to be queried further as they do not exist in any child to u . Indices that map to 1-bits are potential hits which belong to at least one child below u but not all. By adding the number of potential hits in u_{rem} with the number of universal hits found in u_{sim} , an upper bound on the number of matching kmers is determined for each query. If, for a user-specified cutoff $\theta \in [0, 1]$, this count is less than $\theta|V_q|$, then the query cannot exist in this subtree and the subtree is not searched further (it is pruned). If there exist $\theta|V_q|$ or more universal matches, every child beneath u is a query hit and the tree also does not have to be searched further. Only in the case where the count is greater than $\theta|V_q|$ but not enough universal matches have been found do we have to proceed to u 's children. To transition each index from node u to child node c , each index has to be further adjusted by the number of 0-bits in u_{rem} . Once again, this can be calculated in constant time using $rank_0(u_{rem}, V_i)$. By repeating this process down through the tree, SSBT efficiently prunes branches that cannot contain the query, prunes queries which are known to exist in all children, and maintains a consistent hash function across a variable length set of compressed filters. After searching or pruning every branch, the set of leaves which contain the query are then returned. An example of this query process can be found on Fig. 1.

Using this logic, not all query indices are searched at each node in the tree. All indices are initially searched in an 'active' state but may be pruned to 'inactive' if a universal match or mismatch is found. To prevent having to store a unique query set for every node in the tree, we stored V_q only once outside of the SSBT structure and developed a reversible means of activating or deactivating an index, as well as reversing changes made to the index value when descending the tree. Given a vector of indices V_q , we define a single integer — the *tail-index* — to be the position along the V_q vector that contains the last 'active' query index. This tail-index is initialized to the final value in V_q and queries which are 'deactivated' simply swap positions with the tail-index and the tail-index is decremented by one. In such a way, we store the full set of indices but only query those indices up to and including the tail-index. By storing the tail-index present in each node

(a cost of a single integer per node), we can restore all queries which were active at that node. Because the tail-index defines both the pivot between ‘active’ and ‘inactive’ and the swap position for deactivating an index, the order in the vector records the order of deactivation. Because of this, any index which was ‘active’ for an arbitrary node u with tail-index k will always be among the first k indices in V_q and can be re-activated as the tree traversal pops back up the tree. Thus using the tail-index we can exactly store the unique set of indices that need to be searched at any node using only a single vector and an integer at each node.

‘Active’ indices in V_q also have their values adjusted at each step to match the change in size between u_{sim} and u_{rem} and between u and u ’s children. Given an index position i_{rem} that maps to u_{rem} , we can reconstruct the index position i_{sim} that maps to u_{sim} by looking up the i_{rem} -th informative bit in u_{sim} . As $u_{sim} = 1$ defines non-informativity, we simply find the i_{rem} -th 0 in u_{sim} . This can be done using the $select_x(u)[j]$ operation, which we define to be the index position of the j -th bit set to x in the bit vector u . For an RRR-compressed vector, $select_0(u_{sim})[i_{rem}]$ can be computed in $O(\log m)$ time for a length m vector. Likewise given an index position j_{sim} mapping to an arbitrary child node of u , c_{sim} , we can recover the index position i_{rem} mapping to u_{rem} by finding the j_{sim} -th informative bit in u_{rem} . As $u_{rem} = 0$ defines non-informativity, we simply find the j_{sim} -th 1 in u_{rem} . For an RRR-compressed vector, this can be computed in $O(\log m)$ time for a length m vector using the $select_1(u_{rem})[j_{sim}]$ operation. Thus, using just the rank and select operations implicit to an RRR-compressed vector, we can recover any index position at any node given a position along the SSBT and the SSBT split bloom filters themselves.

2.6 Accuracy

SSBT builds off of the base bloom filters used in SBT and encodes the same information found in the leaves of an SBT. The innovations introduced here improve upon the efficiency of that encoding and provide additional information to facilitate rapid search but an SSBT will always yield an identical set of results as SBT. As it has been shown that kmer similarity is highly correlated to the quality of the alignments between sequences [20–23], and SBT has previously determined the accuracy of this metric [1], we did not investigate the accuracy of SSBT, which is the same as SBT, further here. In fact, one can think of SSBT as a lossless compression and reorganization scheme on SBTs that operates before RRR-compression.

3 Results

3.1 Data and Hardware

All data used in this publication was constructed from the set or a subset of 2652 human, RNA-seq short-read sequencing runs from the NIH SRA [2]. At the time of download, these files consisted of the entire set of publicly available, human RNA-seq runs from blood, brain, and breast tissues stored at the SRA

(as determined by keywords in their metadata and excluding files sequenced using the SOLID technology). This dataset has a total size of 5 TB of raw SRA data or 2.7 TB of gzipped fasta sequences. The 50 files selected for the comparison with BFT were chosen at random from this set and have a total size of 49 GB of gzipped fasta sequences. Kmers counts were computed using the Jellyfish 2.0 library for SBT and SSBT and KMC 2.0 for BFT. Jellyfish counts were constructed using the SBT ‘count’ command using an expected kmer set size of $2e9$ and a single hash function. All times in these experiments were obtained on a shared computer with 16 Intel Xeon 2.60 GHz cores using a single thread. BFT was run using default options with a compression constant of 50. SBT and SSBT use a kmer size of 20 while BFT was built using a kmer size of 18 as it only allows kmer lengths that are multiples of 9.

3.2 Evaluation on Build Time and Storage Cost

We compared the construction costs associated between SBT version 0.3.5, SSBT version 0.1, and BFT version 0.8.1 by measuring their respective build time, maximum memory cost during construction, and storage cost of the resulting 100 experiment index. SBT and SSBT’s RAM loads are controlled by setting the maximum number of filters allowed to be loaded simultaneously. We manually adjusted these values to be roughly equivalent to BFT by setting SBT’s maximum in-memory filter load to 100 nodes, with a measured peak RAM of 24.2 GB, and SSBT was limited to 30 nodes, with a measured peak RAM of 16.2 GB. This is in line with our expectations that an uncompressed SSBT internal node is roughly twice as large as an SBT node and requires a more complicated build process. In addition, as BFT uses a different kmer counting tool (KMC2 vs. Jellyfish), the build time records only the time required to construct an index from a pre-computed set of kmer counts. These results, which are summarized in Tables 2 and 3, indicate that BFT takes 11 times longer to build than the combined time to build and compress an SBT and yields an index over three times larger. Similarly SSBT builds and compresses 3.8 times faster than BFT and yields a directly searchable index with less than 1/13th the total storage cost. We note that this is not a strictly apples-to-apples comparison as BFT exactly encodes the kmer set for each experiment while SBT and SSBT are approximate indices with a high false positive for any one kmer.

We also performed a large-scale comparison between SBT and SSBT using the full 2652 experiment set. Both SBT and SSBT were run with a maximum of 100 tree nodes in memory, with a peak memory of roughly 24 GB and 48 GB respectively. The results from this analysis are summarized in Table 4 and show that SSBT is still roughly three times slower to build and compress than the SBT. However we note that the large-scale SSBT is much less efficient to build while much more efficient to compress when compared to the small-scale test. We hypothesize that the more complex construction scheme, as well as the storage costs associated with storing two bloom filters in each internal node, scales poorly with database size and negatively impacts the build time. Meanwhile, the improvements in compression time is primarily a consequence of the non-informative bits. Specifically, as these non-informative bits do not need to be compressed and are cumulative,

Table 2. Build and compression peak RAM loads and on-disk storage costs for SBT, SSBT, and BFT constructed from a 50 experiment set. As BFT’s built-in compression tool is a core part of its build process, we report only a single value for RAM and final size for BFT.

Data index	BFT	SBT	SSBT
Build peak RAM (GB)	21.2	21.5	15.6
Compress peak RAM (GB)	-	24.2	16.2
Uncompressed size (GB)	-	24	35
Compressed size (GB)	13	3.9	0.94

Table 3. Build and compression times for SBT, SSBT, and BFT constructed from a 50 experiment set. As SBT and SSBT were designed to be queried from a compressed state, we compare the time to build and compress against BFT’s time to build.

Data index	BFT	SBT	SSBT
Build time (Min)	137	6	19
Compression time (Min)	-	6.5	17
Total time (Min)	137	12.5	36

it becomes increasingly efficient to compress filters as one descends the tree. This has a much more significant effect at large scales where the accumulation of non-informative bits at low depths effects a greater number of nodes. In addition, we predict that there should be more non-informative bits overall given a larger population of experiments organized to maximize similarity in each sub-tree. This is supported by the fact that the average size of a compressed leaf filter is smaller in the large-scale SSBT than the small-scale, implying that more bits were removed on average from each individual experiment.

While the ratio of build times remained roughly the same, the SSBT is demonstrably more efficient to store at large scales, yielding a five-fold improvement in overall size. As the indices only need to be built once (and can be incrementally built from the uncompressed state), the SSBT is a superior choice when there is sufficient hardware support for its larger uncompressed size. We further note that even this size (1853 GB) is significantly smaller than the raw data, that this size includes the bloom filter representation of every experiment, and that the raw data is not needed during the search once the bloom filter is constructed.

3.3 Evaluation of the Query Time

We evaluated the efficiency of queries in SBT, SSBT, and BFT on three sets of 100 queries. To build each query set, we estimated the expression profiles of all 50 experiments used in the small-scale indices using Sailfish [24]. We then randomly sampled transcripts that were expressed at TPM values at or above 100, 500, or 1,000 in at least one of those files to build three query sets, of 100 queries each.

Table 4. Build statistics for SBT and SSBT constructed from a 2652 experiment set. The sizes are the total disk space required to store a bloom tree before or after compression. In SSBT’s case, this compression includes the removal of non-informative bits.

Data index	SBT	Split SBT
Build time	18 Hr	78 Hr
Compression time	17 Hr	19 Hr
Uncompressed size	1295 GB	1853 GB
Compressed size	200 GB	39.7 GB

Each query was run individually for each tool and the file system cache was emptied at the end of each run to ensure that the average time is an accurate representation of query behavior. The results are summarized in Table 5 and show that SSBT is anywhere from 3–15x faster than either method at this scale. Although this is a significant improvement, we suspect that this 50-experiment test underestimates SSBT’s relative performance, due to SSBT’s efficient storage of similar elements and better optimized querying. We further note that this comparison is not strictly fair to BFT as it returns exact kmer content rather than approximate content. In practice, this would yield significantly fewer false positives which may be preferred in certain contexts.

Table 5. Comparison in query timing (and average peak memory) between SBT, SSBT, and BFT indices for 50 experiments.

Index	TPM ≥ 100	TPM ≥ 500	TPM ≥ 1000
BFT	85 Sec (11.5 GB)	84 Sec (11.5 GB)	84 Sec (11.5 GB)
SBT	19 Sec (2.9 GB)	21 Sec (3.1 GB)	22 Sec (3.2 GB)
SSBT	5.8 Sec (0.64 GB)	6.2 Sec (0.65 GB)	6.3 Sec (0.66 GB)

A larger-scale comparison was performed using the full 2652-experiment indices with SBT and SSBT. The query sets used in this analysis were randomly selected to exist in at least one of 100 randomly sampled experiments out of the full dataset with three TPM-specific sets constructed as before. Each query was run individually and the results are summarized in Tables 6 and 7 and show that SSBT is over five times faster than SBT regardless of the TPM value or cutoff threshold used in either index.

Table 6. Comparison in query timing between SBT and SSBT for 2652 experiments.

Index	TPM ≥ 100	TPM ≥ 500	TPM ≥ 1000
SBT	19.7 Min	20.7 Min	20 Min
SSBT	3.7 Min	3.8 Min	3.6 Min

Table 7. Comparison of query times using different thresholds θ for SBT and SSBT using queries found at $\text{TPM} \geq 100$. ‘RAM SSBT’ describes a hardware-accelerated search enabled locally based on the SSBT’s smaller index size. A similar improvement in speed would be possible on SBT given the necessary hardware.

Query time:	$\theta = 0.7$	$\theta = 0.8$	$\theta = 0.9$
SBT	20 Min	19 Min	17 Min
SSBT	3.7 Min	3.5 Min	3.2 Min
RAM SSBT	31 Sec	29 Sec	26 Sec

Given that SSBT’s speed improvement closely mirrors its size improvement (a five-fold speedup for a five-fold size reduction), we hypothesized that SSBT could be made significantly faster by reducing or eliminating the I/O costs associated with loading and unloading bloom filters. Both SBT and SSBT are likely to benefit from this engineering adjustment, but under our hardware specifications this is only possible for an SSBT, whose directly searchable index uses less than 1% of the size of the original data. This resulted in an additional 7x speedup over regular SSBT and a roughly 39x increase over SBT. We report this result as ‘RAM SSBT’ in Table 7.

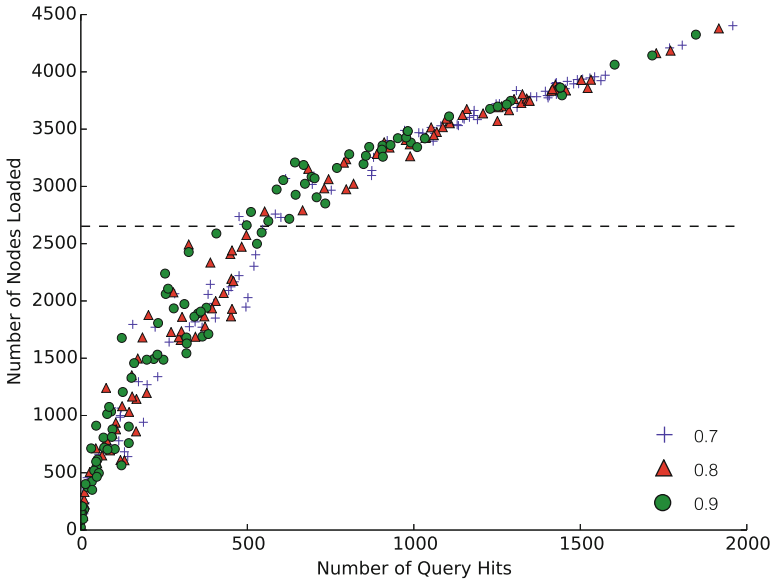


Fig. 2. Number of SSBT nodes that would be loaded if SSBT did not prune queries against the total number of query matches found among 2652 experiments. Blue, green, and red correspond to a kmer matching threshold of 0.7, 0.8, and 0.9 respectively. A naïve approach would search all 2652 leaves as individual bloom filters, represented by the black dashed line.

SSBT’s speed improvement can generally be explained by a reduction in I/O costs resulting from its smaller size but SSBT has another key benefit in the ability to prune queries which are found in every child (“universal query pruning”). This is not relevant for the average query but is a significant improvement in recovery of queries which are expressed in a large fraction of the database. We demonstrate this property by recording the number of SSBT nodes loaded in our TPM 100 set. When universal query pruning is ignored (Fig. 2), queries that are expressed in a majority of the dataset are inefficient to look up, loading many more nodes than the naïve bloom filter search. However, when query pruning is introduced (Fig. 3), significantly fewer queries look at more than 2652 nodes.

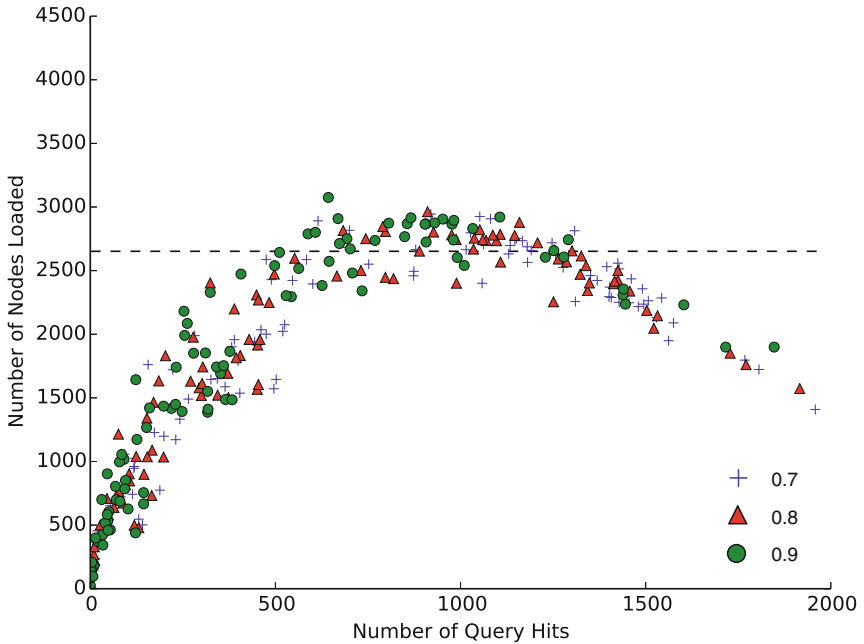


Fig. 3. Number of SSBT nodes loaded against the total number of query matches found among 2652 experiments. Blue, green, and red correspond to a kmer matching threshold of 0.7, 0.8, and 0.9 respectively. A naïve approach would search all 2652 leaves as bloom filters, represented by the black dashed line

4 Conclusion

The Split Sequence Bloom Tree is a novel approach to searching for short read experiments in a large database. It uses a more efficient encoding scheme to generate a compressed but directly searchable index that is at least five times smaller than any existing method. This improvement is significant for all queries

but produces the largest gap over existing techniques when querying transcripts which are found in many experiments. SSBT's improved storage allows 5 TB of sequencing information to be indexed in 40 GB, yielding a 5x increase in speed. Its on-disk memory usage scales more efficiently than any previous tool, and the size of the database which can be stored as a RAM-index is several times larger. For example, a 5 TB dataset could be searched 39x faster using RAM-SSBT but SBT could not be accelerated due to hardware constraints. Although these improvements come at a significant cost in build time and some additional uncompressed storage usage, these operations are typically much more rare than queries. All of the results in this paper were run using a single thread on a single computer. Future work optimizing SSBT for multiple-threaded builds and querying should produce an even more significant improvement in build and query times.

SSBT is open source and available at <http://www.cs.cmu.edu/~ckingsf/software/bloomtree/>.

Acknowledgements. We would like to thank Hao Wang, Natalie Sauerwald, Cong Ma, Tim Wall, Mingfu Sho, and especially Guillaume Marçais, Dan DeBlasio, and Heewook Lee for valuable discussions and comments on the manuscript. This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to Carl Kingsford. It is partially funded by the US National Science Foundation (CCF-1256087, CCF-1319998) and the US National Institutes of Health (R21HG006913, R01HG007104). C.K. received support as an Alfred P. Sloan Research Fellow. B.S. is a predoctoral trainee supported by US National Institutes of Health training grant T32 EB009403 as part of the Howard Hughes Medical Institute (HHMI)-National Institute of Biomedical Imaging and Bioengineering (NIBIB) Interfaces Initiative.

References

1. Solomon, B., Kingsford, C.: Fast search of thousands of short-read sequencing experiments. *Nat. Biotechnol.* **34**, 300–302 (2016)
2. Leinonen, R., Sugawara, H., Shumway, M., The International Nucleotide Sequence Database Collaboration: The sequence read archive. *Nucleic Acids Res.* **39**(Database issue), D19–D21 (2011)
3. Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., Madden, T.L.: BLAST+: architecture and applications. *BMC Bioinform.* **10**(1), 421 (2009)
4. Burrows, M., Wheeler, D.J.: A block sorting lossless data compression algorithm. Technical report 124, Digital Equipment Corporation (1994)
5. Ferragina, P., Manzini, G.: Indexing compressed text. *J. ACM* **52**(4), 552–581 (2005)
6. Grossi, R., Vitter, J.S.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM J. Comput.* **35**(2), 378–407 (2005)
7. Grossi, R., Vitter, J.S., Xu, B.: Wavelet trees: from theory to practice. In: 2011 First International Conference on Data Compression, Communications and Processing (CCP), pp. 210–221. IEEE (2011)

8. Navarro, G., Mäkinen, V.: Compressed full-text indexes. *ACM Comput. Surv.* **39**, 2 (2007)
9. Ziviani, N., Moura, E., Navarro, G., Baeza-Yates, R.: Compression: a key for next-generation text retrieval systems. *IEEE Comput.* **33**, 37–44 (2000)
10. Navarro, G., Moura, E., Neubert, M., Ziviani, N., Baeza-Yates, R.: Adding compression to block addressing inverted indexes. *Inf. Retrieval* **3**, 49–77 (2000)
11. Loh, P.-R., Baym, M., Berger, B.: Compressive genomics. *Nat. Biotechnol.* **30**(7), 627–630 (2012)
12. Daniels, N.M., Gallant, A., Peng, J., Cowen, L.J., Baym, M., Berger, B.: Compressive genomics for protein databases. *Bioinformatics* **29**(13), i283–i290 (2013)
13. Yu, Y.W., Daniels, N.M., Danko, D.C., Berger, B.: Entropy-scaling search of massive biological data. *Cell Syst.* **1**(2), 130–140 (2015)
14. Holley, G., Wittler, R., Stoye, J.: Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms Mol. Biol.* **11**(1), 1 (2016)
15. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7), 422–426 (1970)
16. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: a survey. *Internet Math.* **1**(4), 485–509 (2005)
17. Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2002, Philadelphia, PA, USA*, pp. 233–242. Society for Industrial and Applied Mathematics (2002)
18. Vigna, S.: Broadword implementation of rank/select queries. In: McGeoch, C.C. (ed.) *WEA 2008. LNCS*, vol. 5038, pp. 154–168. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-68552-4_12](https://doi.org/10.1007/978-3-540-68552-4_12)
19. Gog, S., Beller, T., Moffat, A., Petri, M.: From theory to practice: plug and play with succinct data structures. In: Gudmundsson, J., Katajainen, J. (eds.) *SEA 2014. LNCS*, vol. 8504, pp. 326–337. Springer, Cham (2014). doi:[10.1007/978-3-319-07959-2_28](https://doi.org/10.1007/978-3-319-07959-2_28)
20. Rasmussen, K., Stoye, J., Myers, E.: Efficient q-gram filters for finding all ϵ -matches over a given length. *J. Comput. Biol.* **13**(2), 296–308 (2006)
21. Philippe, N., Salson, M., Combes, T., Rivals, E.: CRAC: an integrated approach to the analysis of RNA-seq reads. *Genome Biol.* **14**(3), R30 (2013)
22. Zhang, Q., Pell, J., Canino-Koning, R., Howe, A.C., Brown, C.T.: These are not the k-mers you are looking for: efficient online k-mer counting using a probabilistic data structure. *PLoS ONE* **9**(7), e101271 (2014)
23. Brown, T., Howe, A., Zhang, Q., Pyrkosz, A., Brom, T.: A reference-free algorithm for computational normalization of shotgun sequencing data. [arXiv:1203.4802](https://arxiv.org/abs/1203.4802) [q-bio.GN]
24. Patro, R., Mount, S.M., Kingsford, C.: Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.* **32**, 462–464 (2014)

AllSome Sequence Bloom Trees

Chen Sun¹, Robert S. Harris², Rayan Chikhi³, and Paul Medvedev^{1,4,5}(✉)

¹ Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, USA
pashadag@cse.psu.edu

² Department of Biology, The Pennsylvania State University, University Park, USA

³ CNRS, CRIStAL, University of Lille, Lille, France

⁴ Department of Biochemistry and Molecular Biology,
The Pennsylvania State University, University Park, USA

⁵ Genome Sciences Institute of the Huck, The Pennsylvania State University,
University Park, USA

Abstract. The ubiquity of next generation sequencing has transformed the size and nature of many databases, pushing the boundaries of current indexing and searching methods. One particular example is a database of 2,652 human RNA-seq experiments uploaded to the Sequence Read Archive. Recently, Solomon and Kingsford proposed the Sequence Bloom Tree data structure and demonstrated how it can be used to accurately identify SRA samples that have a transcript of interest potentially expressed. In this paper, we propose an improvement called the AllSome Sequence Bloom Tree. Results show that our new data structure significantly improves performance, reducing the tree construction time by 52.7% and query time by 39–85%, with a price of up to 3x memory consumption during queries. Notably, it can query a batch of 198,074 queries in under 8 h (compared to around two days previously) and a whole set of k -mers from a sequencing experiment (about 27 mil k -mers) in under 11 min.

Keywords: Sequence Bloom Trees · Bloom filters · RNA-seq · Data structures · Algorithms · Bioinformatics

1 Introduction

Data structures for indexing and searching of databases have always been a core contribution of algorithmic bioinformatics to the analysis of biological data and are the building blocks of many popular tools [18]. Traditional databases may include reference genome assemblies, collections of known gene sequences, or reads from a single sequencing experiment. However, the ubiquity of next generation sequencing has transformed the size and nature of many databases. Each

C. Sun and R.S. Harris—Contributed equally to the work.

The full version of this paper is available at [33].

sequencing experiment results in a collection of reads (gigabytes in size), typically deposited into a database such as the Sequence Read Archive (SRA) [15]. There are thousands of experiments deposited into the SRA, creating a database of unprecedented size in genomics (4 petabytes, as of 2016). The SRA enables public access of the database via meta-data queries on the experiments' name, type, organism, etc. However, efficiently querying the raw read sequences of the database has remained out of reach for today's indexing and searching methods, until earlier this year [31].

Given a transcript of interest, an important problem is to identify all publicly available sequenced samples which express it. The SRA contains thousands of human RNA-seq experiments, providing a powerful database to answer this question. One approach is to use tools such as [4, 27, 34] to first identify transcripts present in each of the experiments; however, running these tools on a massive scale is time prohibitive (though cloud-enabled tools like Rail-RNA [26] are making inroads). Moreover, they introduce biases and can easily miss a transcript that is supported by the reads. Another approach is to align the SRA reads to the transcript of interest; however, this approach is infeasible for such large datasets [31].

Recently, Solomon and Kingsford proposed the Sequence Bloom Tree (SBT) data structure and demonstrated how it can accurately identify samples that may have the transcript of interest expressed in the read data [31]. SBT was a breakthrough, allowing to query a set of 214,293 transcripts against a database of 2,652 human RNA-seq experiments in just under 4 days. The SBT is not intended to replace more thorough methods, like alignment, but is intended to be complementary, narrowing down the set of experiments for which a more rigorous investigation is needed.

In this paper, we present the AllSome Sequence Bloom Tree (SBT-ALSO), a time and space improvement on the original SBT (denoted by SBT-SK). It combines three new ideas. The first one is a better construction algorithm based on clustering. The second one is a different representation of the internal nodes of the tree so as to allow earlier pruning and faster exploration of the search space. The final one is building a Bloom filter on the query itself. This allows quick execution of queries that are not just transcripts but are themselves large sequencing experiments.

We evaluate SBT-ALSO on the database of 2,652 human RNA-seq sequencing runs used in [31]. SBT-ALSO reduces tree construction time by 52.7%, when given the Bloom filters of the datasets. It reduces query time by 39–85%, with a price of up to 3x memory consumption. Notably, it can query a batch of 198,074 queries in under 8 h, compared to over two days for SBT-SK. It can also query a whole set of k -mers from a sequencing experiment (about 27 mil k -mers) in under 11 min, compared to more than 23 h by SBT-SK. Our software is open source and freely available via GitHub¹.

¹ SBT-ALSO GitHub repository: <https://github.com/medvedevgroup/bloomtree-allsome>.

2 Related Work

This work falls into the general category of string pattern matching, where we are asked to locate all occurrences of a short pattern in a large text. In many cases, it is useful to pre-process the text to construct an index that will speed up future queries. The k -mer-index, trie, suffix tree, suffix array, BWT, and FM-index are examples of such indices [18]. These form the basis of many read alignment tools such as BWA-mem and Bowtie 2. While many of these approaches are space and time efficient in their intended setting, they can nevertheless be infeasible on terabyte or petabyte scale data. Other approaches based on word-based indices [25,36] and compressive genomics [17,35] do not help for the type of data and queries we consider in this paper.

A Bloom filter (BF) is widely used to improve scalability by determining whether the pattern occurs in the text, without giving its location. It is a space efficient data structure for representing sets that occasionally provides false-positive answers to membership queries [3]. For pattern matching, a BF can be constructed for all the constituent k -mers (strings of length of k) of the text. Then, if a high percentage of a pattern’s constituent k -mers match, the text is a potential match and a full search can be performed. BFs are used in several bioinformatics contexts such as assembly [6,12,22,30], to index and compress whole genome datasets [29], and to compare sequencing experiments against whole genomes [32].

When pattern matching against a database of read collections from sequencing experiments, additional factors need to be considered. First, the reads contain sequencing errors. Second, they only represent short fragments of the underlying DNA and are typically much shorter than the pattern. Third, there are many texts, each of which is its own sequencing experiment. The goal is to identify all texts that match the pattern. A simple way to adapt the BF idea to this case is to simply build a BF for every text and check the pattern separately against every text’s BF. A more sophisticated approach builds a tree to index the collection of BFs [8]. This Bloofi data structure was introduced in the context of distributed data provenance, but it was later adapted to the bioinformatics setting by Solomon and Kingsford in [31].

An orthogonal approach is the Bloom Filter Trie (BFT) [13], which works similarly to a trie on the k -mers in all the texts. Each leaf contains a bitvector describing the texts in which that k -mer appears, and BFs are cleverly used inside the trie to “jump down” ℓ positions at a time, thus speeding up the trie traversal process. The BFT complexity scales up with the number of k -mers in the query, while SBT complexity scales up with the number of datasets. Thus the two approaches suggest orthogonal use cases. In particular, the BFT is very efficient for queries that are single k -mers, significantly outperforming the SBT. An approach that uses BFT to query longer patterns like the ones we consider in this paper is promising but is not yet available.

There is also a body of work about storing and indexing assembled genomes [2,10,16,21,23], which is part of the growing field of pangenomics [7]. However, our work relates to the indexing of unassembled data (i.e. reads) as

opposed to complete genomes. In addition to the topics specifically mentioned above, there are other studies related to scaling up indexing methods [9, 20], though the list here is in no way complete.

3 Technical Background

Terminology: Let x and y be two bitvectors of the same length. The bitwise AND (i.e. intersection) between x and y is written as $x \cap y$, and the bitwise OR (i.e. union) is $x \cup y$. A bitvector can be viewed as a set of positions set to 1, and this notation is consistent with the notion of set union and intersection. The set difference of x and y is written as $x \setminus y$ and can be defined as $x \setminus y = x$ AND (NOT y). A *Bloom filter* (BF) is a bitvector of length b , together with p hash functions, h_1, \dots, h_p , where b and p are parameters. Each hash function maps a k -mer to an integer between 0 and $b - 1$. The empty set is represented as an array of zeros. To add a k -mer x to the set, we set the position $h_i(x)$ to 1, for all i . To check if a k -mer x is in the set, we check that the position $h_i(x)$ is 1, for all i . In this paper, we assume that the number of hash functions is 1. Next, consider a rooted binary tree. The parent of a non-root node u is denoted as $parent(u)$, and the set of all the leaves of the subtree rooted at a node v is denoted by $leaves(v)$. Let $lchild(u)$ and $rchild(u)$ refer to the left and right children of a non-leaf node u , respectively.

SBT: Let Q be a non-empty set of k -mers, and let B be a k -mer BF. Given $0 \leq \theta \leq 1$, we say that Q θ -matches B if $|\{x \in Q : x \text{ exists in } B\}|/|Q| \geq \theta$. That is, the percentage of k -mers in Q that are also in B (including false positive hits) is at least θ . Solomon and Kingsford consider the following problem. We are given a database $D = \{D_1, \dots, D_n\}$, where each D_i is a BF of size b . The query is a k -mer set Q , and the result of the query should be the set $\{i : Q \theta\text{-matches } D_i\}$. The goal is to build a data structure that can construct an index on D to support multiple future queries.

We make a distinction between the abstract data type that Solomon and Kingsford propose for the problem and their implementation of it. We call the first SBT, and the second SBT-SK (note that in [31] no distinction is made and SBT refers to both). A rooted binary tree is called a *Sequence Bloom Tree* (SBT) of a database D if there is a bijection between the leaf nodes and the elements of D . Define $B_{\cup}(u)$ for a leaf node u as its associated database element and $B_{\cup}(u)$ for an internal node as $\bigcup_{i \in leaves(u)} B_{\cup}(i)$. Note that $B_{\cup}(u)$ of an internal node u can be equivalently defined as $B_{\cup}(lchild(u)) \cup B_{\cup}(rchild(u))$. Each node u then represents the set of database entries corresponding to the descendant leaves of u . Additionally, the SBT provides an interface to construct the tree from a database, to query a k -mer set against the database, and to insert/delete a BF into/from the database. An example of an SBT is shown in Fig. 2.

SBT-SK: We call the implementation of the SBT interface provided in [31] as SBT-SK. In SBT-SK, each node u is stored as a compressed version of $B_{\cup}(u)$. The compression is done using RRR [28] implemented in SDSL [11], which allows to efficiently test whether a bit is set to 1 without decompressing the bitvector. To *insert* a BF B into a SBT T , SBT-SK does the following. If T is empty, it just adds B as the root. Otherwise, let r be the root. If r is a leaf, then add a new root r' that is the parent of B and r and set $B_{\cup}(r') = B_{\cup}(r) \cup B$. Otherwise, take the child v of r that has the smallest Hamming distance to B , recursively insert B in the subtree rooted at v , and update $B_{\cup}(r)$ to be $B_{\cup}(r) \cup B$. Note that because RRR compressed bitvectors do not support bitwise operations, each bitvector must be first decompressed before bitwise operations are performed and then recompressed if any changes are made. The running time of an insertion is proportional to the depth of the SBT. To *construct* the SBT for a database, SBT-SK starts with an empty tree and inserts each element of the database one-by-one. Construction can take time proportional to nd , where d is the depth of the constructed SBT. The left panel of Fig. 1 provides an example of the construction algorithm. To *query* the database for a k -mer set Q , SBT-SK first checks if Q θ -matches the root.

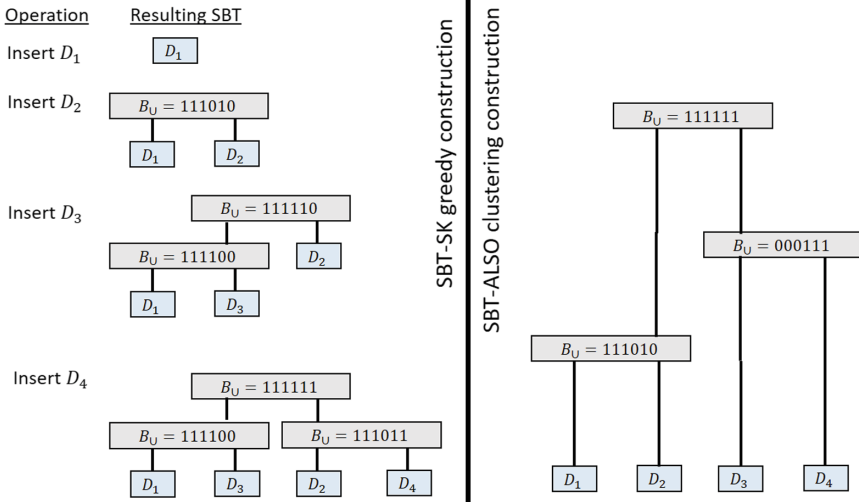


Fig. 1. Example of the SBT-SK and SBT-ALSO construction algorithms for the database $D = \{D_1 = 111000, D_2 = 111010, D_3 = 000100, D_4 = 000011\}$. Leaves shown in blue, internal nodes in gray. In this example, the dataset can be partitioned into two types: 000xxx and 111xxx, based on the first 3 bits. In the SBT-SK construction, after the first two experiments are inserted (both of type 111xxx), they are destined to be in the two different sides of the tree (regardless of future insertions). Any future 111xxx type query will have to examine all the nodes. The SBT-ALSO construction, on the other hand, groups together the experiments so that future 000xxx type or 111xxx type queries will have to examine only about half the nodes of the tree.

If yes, then it recursively queries the children of the root. When the query hits a leaf node, it returns the leaf if Q θ -matches it.

Since SBT is designed to work on very large databases, its implementation should avoid loading the database into memory. In SBT-SK, each $B_{\cup}(u)$ is stored on disk and only loaded into memory when u is being θ -matched by a query. When there are multiple queries to be performed, SBT-SK will *batch* them together so that the θ -matching of multiple queries to the same node will be performed simultaneously. Hence, each node needs to be loaded into memory only once per batch. We implement the same strategy in SBT-ALSO.

4 Methods

We propose the AllSome SBT as an alternative implementation of the SBT abstract data type. In this section, we describe the construction and query algorithms. Insertion and deletion algorithms are the same as in SBT-SK, though some special care is needed. For completeness, they are described in the full version [33].

4.1 ALLSOME Node Representation and Regular Query Algorithm

Define the intersection of leaves in the subtree rooted at a node u as $B_{\cap}(u) = \bigcap_{i \in \text{leaves}(u)} B_{\cup}(i)$. Intuitively, we can partition the 1 bits of $B_{\cup}(u)$ into three sets: $B_{\text{all}}(u)$, $B_{\text{some}}(u)$, and $B_{\cap}(\text{parent}(u))$. $B_{\text{all}}(u)$ are the bits that appear in all of $\text{leaves}(u)$, excluding those in all of $\text{leaves}(\text{parent}(u))$. $B_{\text{some}}(u)$ are the bits in some of $\text{leaves}(u)$ but not in all. Both sets therefore exclude bits present in $B_{\cap}(\text{parent}(u))$. Formally, define

$$\begin{aligned} B_{\text{all}}(u) &= B_{\cap}(u) \setminus B_{\cap}(\text{parent}(u)) \\ B_{\text{some}}(u) &= B_{\cup}(u) \setminus B_{\cap}(u) \end{aligned}$$

At the root r , define $B_{\cap}(\text{parent}(r)) = \emptyset$. $B_{\text{all}}(u)$ and $B_{\text{some}}(u)$ are stored using two bitvectors of size b compressed with RRR. $B_{\cup}(u)$ and $B_{\cap}(u)$ are not explicitly stored. We refer to this representation of the nodes using B_{all} and B_{some} as the ALLSOME representation (Fig. 2 gives an example).

When we receive a query k -mer set Q , we hash each k -mer to determine the list of BF bits corresponding to Q . These are a multi-set of position indices (between 0 and $b - 1$), stored as an array. We call these the list of *unresolved* bit positions. We also maintain two counters: the number of bit positions that have been determined to be 1 (*present*), and the number of bit positions determined to be 0 (*absent*). These counters are both initially 0. The query comparison then proceeds in a recursive manner. When comparing Q against a node u , each unresolved bit position that is 1 in $B_{\text{all}}(u)$ is removed from the unresolved list and the present counter is incremented. Each unresolved bit position that is 0 in $B_{\text{some}}(u)$ is removed from the unresolved list and the absent counter is incremented. If the present counter is at least $\theta|Q|$, we add $\text{leaves}(u)$ to the

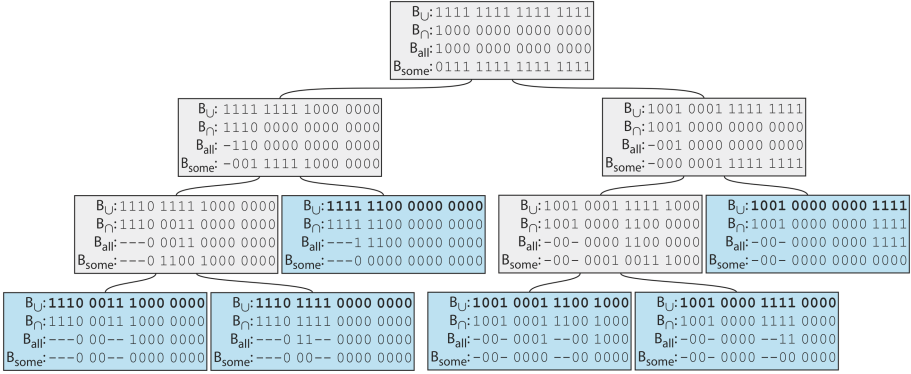


Fig. 2. Example SBT on $D = \{1110001110000000, 1110111100000000, 1111110000000000, 1001000111001000, 1001000011110000, 1001000000001111\}$. Leaves shown in blue, internal nodes in gray. In SBT-ALSO, only B_{all} and B_{some} are explicitly stored, while in SBT-SK, only B_U is stored. Bits present in B_{all} at one node are shown as hyphens (‘-’) in the B_{all} and B_{some} of its descendants, but in the actual SBT-ALSO data structure they are zeros.

list of θ -matches and terminate the search of u ’s subtree. If the absent counter exceeds $(1 - \theta)|Q|$, we realize that Q will not θ -match any of the leaves in the subtree rooted at u and terminate the search of u ’s subtree. If neither of these holds, we recursively pass the two counters and the list of unresolved bits down to its children. When we reach a leaf, the unresolved list will become empty because B_{some} is empty at a leaf, and the algorithm will necessarily terminate.

The idea behind the ALLSOME representation is that in a database of biologically associated samples, there are many k -mers that are shared between many datasets. In the SBT-SK representation, a query must continue checking for the presence of these k -mers at every node that it encounters. By storing at u all the bits that are present in all the leaves of its subtree, we can count those bits as resolved much earlier in the query process – limiting the amount of bit look-ups performed. Moreover, we will often prune the search space earlier and decrease the number of bitvectors that need to be loaded from disk. A query that matches all the leaves of a subtree can often be resolved after just examining the root of that subtree. In the extreme case, the number of nodes examined in a search may be less than the number of database entries that are matched.

A second important point is that the size of the uncompressed bitvectors at each node is now twice as large as before. Because query time has a large I/O component, this has potential negative effects. Fortunately, we observe that the compressed size of these bitvectors is roughly proportional to the number of 1s that are contained. By defining the ALLSOME representation as we do, the number of 1s in total in $B_{all}(u)$ and $B_{some}(u)$ is no more than the number of ones in $B_U(u)$. Moreover, because we exclude $B_\Gamma(u)$ from all of u ’s descendants, the number of 1s is less.

4.2 Construction Algorithm

Except for large queries or large batches of queries, the running time of the query algorithm is dominated by the I/O of loading bitvectors into memory [31]. If the number of leaves that the query θ -matches is localized within the same part of the SBT, then fewer internal nodes have to be explored and, hence, fewer bitvectors have to be loaded into memory. The SBT-SK construction algorithm is greedy and sensitive to the order in which the entries are inserted into the tree, which can lead to trees with poor localization (see example in Fig. 1).

To improve the localization property of the tree, we propose a non-greedy construction method based on agglomerative hierarchical clustering [14]. Every D_i is initially its own SBT, with its B_{\cup} loaded into memory. At every step, two SBTs are chosen and joined together to form a new SBT. The new SBT has a root node r with the left and right subtrees corresponding to the two SBTs being joined. $B_{\cup}(r)$ is computed as $B_{\cup}(lchild(r)) \cup B_{\cup}(rchild(r))$. To choose the pair of SBTs to be joined, we choose the two SBTs that have the smallest Hamming distance between the B_{\cup} of their roots. The right panel of Fig. 1 shows how our construction algorithm works.

Since each B_{\cup} is a large bitvector, computing and maintaining the pairwise distances between all pairs is computationally expensive. Instead, we use the following heuristic. We fix a number $b' \ll b$ (e.g. $b' = 10^5 \ll 10^9 = b$) and then extract b' bits from each D_i , starting from a fixed but arbitrary offset. We then run the above clustering algorithm on this smaller database of extracted bitvectors.

The resulting topology is then extracted and used for constructing the B_{all} and B_{some} bitvectors for all the nodes. We process the nodes in a bottom-up fashion. Initially, for all leaves u , we set $B_{all}(u) = B_{\cup}(u)$ and $B_{some}(u) = \emptyset$. For the general case, consider an internal node u whose children ℓ and r have already been processed. All bits that are set in both $B_{all}(\ell)$ and $B_{all}(r)$ go into $B_{all}(u)$:

$$B_{all}(u) = B_{all}(\ell) \cap B_{all}(r)$$

Additionally, the B_{all} bits of ℓ and r must exclude those that are set in the parent $B_{all}(u)$. After computing $B_{all}(u)$, we can unset these bits:

$$B_{all}(v) = B_{all}(v) \setminus B_{all}(u), \text{ where } v \in \{\ell, r\}. \quad (1)$$

Note that this is the only necessary update to the bitvectors of nodes in the subtree rooted in ℓ or r . Next, we must compute $B_{some}(u)$, which is the set of bits that exist in some of u 's children nodes but not all:

$$B_{some}(u) = B_{some}(\ell) \cup B_{some}(r) \cup B_{all}(\ell) \cup B_{all}(r)$$

Note that here we are using the B_{all} after the application of Eq. (1). This completes the necessary updates to the tree for a node u . These updates can be efficiently computed using bitwise operations on uncompressed bitvectors, so we

keep them uncompressed in memory and only compress them when they are written to disk and are no longer needed. The total time for construction is proportional to n and not to nd , as with SBT-SK. For completeness, we provide a more formal algebraic derivation of the update formulas in the full version [33].

4.3 Large Query Algorithm

The “regular query” algorithm (Sect. 4.1) is designed with relatively small queries in mind (e.g. thousands of k -mers from a transcript). However, after performing a new sequencing experiment, it might be desirable to query the database for other similar samples. In such cases, the query would itself be a whole sequencing experiment, containing millions of k -mers. Our experimental results show that neither SBT-SK nor our own regular query algorithm is efficient for these large queries.

While for small queries, the running time is dominated by the I/O of loading bitvectors into memory, for large queries, the time taken to look up the query k -mers in the B_{all} and B_{some} of a node becomes the bottleneck. Let B_Q be the Bloom filter of size b for the k -mers in the query Q . We propose an alternate “large query” algorithm that can be used whenever the number of k -mers in the query exceeds some pre-defined user threshold. This large query algorithm is identical to the regular one except in the way that the unresolved list is maintained and updated. The basic idea is that instead of checking each k -mer in Q one-by-one, we can do bitwise comparisons using B_Q . Assume for the moment that there are no two k -mers in Q that hash to the same position (recall that our BFs have only one hash function). In this case, the list of unresolved bit positions can be represented as the set of 1 positions in B_Q . At a node u , we first increment the *present* counter by the number of ones in $B_Q \cap B_{all}(u)$ and update the unresolved bit positions to be $B'_Q = B_Q \setminus B_{all}(u)$. Then we increment the *absent* counter by the number of ones in $B'_Q \setminus B_{some}(u)$ and update the unresolved bit positions to be $B''_Q = B'_Q \cap B_{some}(u)$. If the counters do not exceed their respective thresholds, then we pass them and the remaining unresolved bits (B''_Q) down to the children.

When there are k -mers that hash to the same bit positions, the above algorithm can still be used as a heuristic. In fact, it can be shown that the hits returned by the above heuristic algorithm are always a subset of the hits that are returned by an exact algorithm, since the heuristic’s counter values are never greater than those of the exact algorithm. But, we can obtain an exact algorithm by modifying the above heuristic to also maintain a list of bit positions that have multiple k -mers hashing to them. An entry of the list is a bit position and the number of k -mers that hash to it. Whenever we make a bitwise comparison involving B_Q , this list is scanned to convert numbers of bits to numbers of k -mers. When the list is small, this exact algorithm should not be significantly slower than the heuristic one.

Unfortunately, computing bitwise operations cannot be efficiently done on RRR compressed bitvectors. To support the large query algorithm, the bitvectors are compressed using the Roaring [5] scheme (abbreviated ROAR). Roaring

bitmaps are compressed using a hybrid technique that allows them to efficiently support set operations on bitvectors (intersection, union, difference, etc.). However, we found that they generally do not compress as well as RRR on our data, leading to longer I/O times. In cases where both small and large queries are common, and query time is more important than disk space, both a ROAR and an RRR compressed tree can be maintained.

5 Results

We implemented SBT-ALSO, building on the SBT-SK code base [1]. Solomon and Kingsford already explored the advantages, disadvantages and accuracy of the SBT approach as a way of finding experiments where the queried transcripts are expressed [31]. Since SBT-ALSO gives identical query results as SBT-SK, we therefore focus our evaluation on its resource utilization. We used the same dataset for evaluation as in [31]. This is the set of 2,652 runs representing the entirety (at the time of [31]) of human RNA-seq runs from blood, brain, and breast tissues at the SRA, excluding those sequenced with SOLID. In [31], each sequencing run was converted to a k -mer Bloom filter ($b = 2 \cdot 10^9$, $k = 20$) by the Jellyfish k -mer-counting software (containing k -mers that occur greater than a file-dependent threshold, typically at least 3 occurrences). We downloaded these BF's from [1] and used them as our database. Per the results of [31], this Bloom filter size leads to a false positive rate of 0.5 for an individual Bloom filter. We performed experiments on an OpenStack instance with 12 vCPUs (Intel Xeon E312xx), 128 GB memory, and 4 TB network-mounted disk storage.

To choose the appropriate number of bits to use for clustering (b'), we randomly sampled 5,000 bitvector pairs from the dataset and computed their pairwise distances. We then computed distances for the same pairs using only b' bits, for various values of b' . The two distance metrics showed a high correlation ($r^2 = 0.9999874$) for $b' = 500,000$.

We then constructed SBT-SK and SBT-ALSO, as well as two other trees to help us separate out the contributions of the clustering algorithm from the ALLSOME representation. These two trees are SBT-SK+CLUST, which uses the B_{\cup} node representation of SBT-SK but the SBT-ALSO clustering construction, and SBT-SK+AS, which uses the greedy construction of SBT-SK but the ALLSOME node representation of SBT-SK.

First, we compared the space and time used to construct SBT-SK and SBT-ALSO (Table 1). SBT-ALSO reduces the tree construction time by 52.7% and resulting disk space by 11.4%. It requires twice as much intermediate space, due to maintaining two uncompressed bitvectors for each node instead of just one.

To study the regular query performance, we downloaded all known transcripts at least k bases long (198,074 of them) from Gencode (ver. 25). We then queried several subsets of transcripts against both trees, and measured the number of nodes examined for each query (Fig. 3) as well as the running time (Table 2). The results of all query experiments in this paper were verified to be equivalent between the tested data structures. SBT-ALSO reduces the runtime by

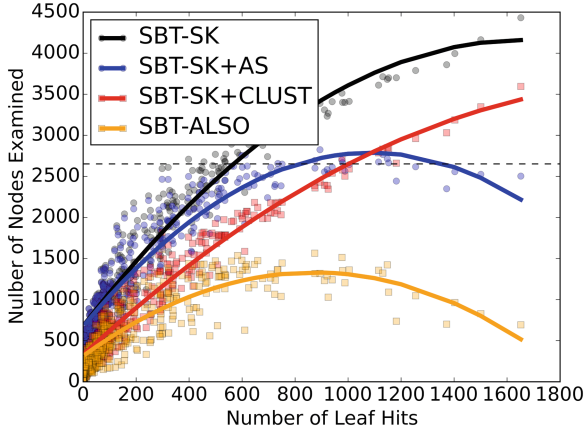


Fig. 3. Number of nodes examined per query for SBT-SK, SBT-ALSO, as well two intermediate SBTs. A set of 1,000 transcripts were chosen at random from Gencode set, and each one queried against the four different trees. A dot represents a query and shows the number of matches in the database (x-axis) compared to the number of nodes that had to be loaded from disk and examined during the search (y-axis). For each tree (color), we interpolated a curve to show the pattern. The dashed horizontal line represents the hypothetical algorithm of simply checking if the query θ -matches against each of the database entries, one-by-one. For θ , we used the default value in the SBT software ($\theta = 0.9$).

39 - 85%, depending on the size of the batch, likely due to the fact that the number of nodes examined per query is reduced by 52.7%, on average. Notably, SBT-ALSO was able to query a very large batch (198,074 queries) in under 8 h, while SBT-SK took over 2 days. SBT-ALSO uses more memory than SBT-SK on larger batches.

To study the performance of the large query algorithm, we selected an arbitrary run from our database (SRR806782) and used Jellyfish [19] to extract all 20-mers that appear at least three times. These 27,546,676 k -mers formed one query. In heuristic mode, the large query algorithm was 22 times faster than the regular one, but only detected 47 hits, which is a subset of the 50 hits by regular algorithm (Table 3). In the exact mode, the large query algorithm recovered all the hits (as expected) and was 18 times faster. Compared to SBT-SK, it was 155 times faster.

The clustering construction, even without the ALLSOME representation, significantly reduces the number of nodes that need to be examined per query (36.5% on average when comparing SBT-SK to SBT-SK+CLUST in Fig. 3). The improvement seems to be uniform regardless of the number of leaf hits. As expected, this leads a significant improvement to the running time (19-32%, Sect. 5).

The ALLSOME representation, without the clustering construction, also gives the benefit of allowing earlier query resolution, but the effect only becomes

Table 1. Construction time and space. Times shown are wall-clock times. A single thread was used. Note the SBT-SK tree that was constructed for the purposes of this table differs from the tree used in [31] and in our other experiments because the insertion order during construction was not the same as in [31] (because it was not described there).

	SBT-SK	SBT-ALSO
Construction of tree topology (i.e. clustering)	N/A	27 m
Construction of internal nodes	56 h 54 m	26 h 3 m
Peak memory usage	726 MB	908 MB
Temporary disk space	1,235 GB	2,469 GB
Final disk space	200 GB	177 GB

Table 2. Query wall-clock run times and maximum memory usage, for batches of different sizes. For the batch of 1,000 queries, we used the same 1,000 queries as in Fig. 3. For the batch of 100 queries, we generated three replicate sets, where each set contains 100 randomly sampled transcripts without replacement from the 1,000 queries set. For the batch of 10 queries, we generated 10 replicate sets by partitioning one of the 100 query sets into 10 sets of 10 queries. For the batch of 1 query, we generated 50 replicate sets by sampling 50 random queries from Gencode set. The shown running times are the averages of these replicates. For θ , we used the default value in the SBT software ($\theta = 0.9$).

# query	SBT-SK	SBT-SK+AS	SBT-SK+CLUST	SBT-ALSO
1	1.2 m/301 MB	2.7 m/301 MB	0.9 m/299 MB	0.5 m/301 MB
10	4 m/305 MB	8.3 m/319 MB	3.3 m/304 MB	2 m/313 MB
100	7.7 m/315 MB	13.7 m/346 MB	6.5 m/317 MB	4.7 m/353 MB
1,000	25.5 m/420 MB	20.8 m/575 MB	17.3 m/418 MB	8.3 m/639 MB
198,074	3082 m/22 GB	1286 m/51 GB	1910 m/23 GB	463 m/63 GB

Table 3. Performance of different trees and query algorithms on a large query. We show the performance of SBT-SK and three query algorithms using SBT-ALSO compressed with ROAR: the regular algorithm, the large exact algorithm, and the large heuristic algorithm. We show the wall-clock run time and maximum RAM usage. We used $\theta = 0.8$ for this experiment. The ROAR compressed tree was 190 GB (7.3% larger than the RRR tree).

	SBT-SK	SBT-ALSO		
	Regular alg	Regular alg	Large exact alg	Large heuristic alg
Query time	1397 m 18 s	195 m 33 s	10 m 35 s	8 m 32 s
Query memory	2.3 GB	4.7 GB	1.3 GB	1.2 GB

pronounced for queries that hit a lot of leaves. For instance, queries that hit more than 800 leaves examined 27.4% less nodes in SBT-SK+AS than in SBT-SK. In the extreme case, there are seven queries out of 1,000 where the number of nodes examined is less than the number of leaf hits, something that is not possible with SBT-SK. However, the benefits of clustering construction and ALLSOME representation are synergistic: the multiplicative effect of their individual contributions (42.3% decrease in number of examined nodes) is less than the observed effect of their combined contributions (52.7%). In terms of the running time performance, the ALLSOME representation incurs the overhead of making two queries per active bit, instead of just one. This is more than compensated by a decrease in the amount of active bits when the tree is clustered well. But, as the SBT-SK+AS column of Sect. 5 shows, the running time can actually deteriorate when the tree is not clustered.

6 Discussion

In this paper we present an alternate implementation of the SBT that provides substantial improvements in query and construction time. We are especially effective for large batches of queries (6 times faster) or for large queries (155 times faster). Solomon and Kingsford make a convincing case that an efficient SBT implementation translates to an efficient and accurate solution to the broader problem of identifying RNA-seq samples that express a transcript of interest. They study the best parameter values of SBT (θ, k, b, p) to achieve accuracy and speed for the broader problem. The focus of this paper is on improving resource performance, and hence we do not revisit these questions, however, a more thorough exploration of the biological questions that the SBT can answer will be important moving forward.

The implications of using the SBT for queries which are themselves sequencing experiments were not explored in SBT-SK or here. The BFT [13], if adapted to multi- k -mer queries with θ -matching, could prove to be powerful in this context. In general, the question of whether the percentage of matching k -mers is a good metric for comparing sequencing experiments is still open, and more investigation into how to best measure similarity is needed (e.g. see [24]). However, our large query algorithm opens the door for efficiently exploring the parameter space of k -mer-based approaches.

Acknowledgements. This work has been supported in part by NSF awards DBI-1356529, CCF-551439057, IIS-1453527, and IIS-1421908 to PM.

References

1. SBT-SK software and data. <http://www.cs.cmu.edu/%7Eeckingsf/software/bloomtree/>. Accessed 01 July 2016
2. Baier, U., Beller, T., Ohlebusch, E.: Graphical pan-genome analysis with compressed suffix trees and the Burrows-Wheeler transform. *Bioinformatics* **32**, 497–504 (2015)

3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7), 422–426 (1970)
4. Bray, N.L., Pimentel, H., Melsted, P., Pachter, L.: Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.* **34**(5), 525–527 (2016)
5. Chambi, S., Lemire, D., Kaser, O., Godin, R.: Better bitmap performance with roaring bitmaps. *Softw. Pract. Exp.* **46**(5), 709–719 (2015)
6. Chikhi, R., Rizk, G.: Space-efficient and exact de Bruijn graph representation based on a bloom filter. *Algorithms Mol. Biol.* **8**(1), 1 (2013)
7. Consortium, C.P.G., et al: Computational pan-genomics: status, promises and challenges. *Brief. Bioinform.* bbw089 (2016)
8. Crainiceanu, A., Lemire, D.: Bloofi: multidimensional bloom filters. *Inf. Syst.* **54**, 311–324 (2015)
9. Dolle, D.D., Liu, Z., Cotten, M.L., Simpson, J.T., Iqbal, Z., Durbin, R., McCarthy, S., Keane, T.: Using reference-free compressed data structures to analyse sequencing reads from thousands of human genomes. *Genome Res.* **27**, 300–309 (2016)
10. Ernst, C., Rahmann, S.: PanCake: a data structure for pangenomes. *Ger. Conf. Bioinform.* **34**, 35–45 (2013)
11. Gog, S., Beller, T., Moffat, A., Petri, M.: From theory to practice: plug and play with succinct data structures. In: Gudmundsson, J., Katajainen, J. (eds.) SEA 2014. LNCS, vol. 8504, pp. 326–337. Springer, Cham (2014). doi:[10.1007/978-3-319-07959-2_28](https://doi.org/10.1007/978-3-319-07959-2_28)
12. Heo, Y., Wu, X.L., Chen, D., Ma, J., Hwu, W.M.: BLESS: bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics* **30**, 1354–1362 (2014)
13. Holley, G., Wittler, R., Stoye, J.: Bloom filter trie – a data structure for pangenome storage. In: Pop, M., Touzet, H. (eds.) WABI 2015. LNCS, vol. 9289, pp. 217–230. Springer, Heidelberg (2015). doi:[10.1007/978-3-662-48221-6_16](https://doi.org/10.1007/978-3-662-48221-6_16)
14. de Hoon, M.J., Imoto, S., Nolan, J., Miyano, S.: Open source clustering software. *Bioinformatics* **20**(9), 1453–1454 (2004)
15. Leinonen, R., Sugawara, H., Shumway, M.: The sequence read archive. *Nucleic Acids Res.* **39**, D19–D21 (2010)
16. Liu, B., Zhu, D., Wang, Y.: deBWT: parallel construction of Burrows-Wheeler Transform for large collection of genomes with de Bruijn-branch encoding. *Bioinformatics* **32**(12), i174–i182 (2016)
17. Loh, P.R., Baym, M., Berger, B.: Compressive genomics. *Nat. Biotechnol.* **30**(7), 627–630 (2012)
18. Mäkinen, V., Belazzougui, D., Cunial, F., Tomescu, A.I.: *Genome-Scale Algorithm Design*. Cambridge University Press, Cambridge (2015)
19. Marçais, G., Kingsford, C.: A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* **27**(6), 764–770 (2011)
20. Marchet, C., Limasset, A., Bittner, L., Peterlongo, P.: A resource-frugal probabilistic dictionary and applications in (meta) genomics (2016). arXiv preprint: [arXiv:1605.08319](https://arxiv.org/abs/1605.08319)
21. Marcus, S., Lee, H., Schatz, M.C.: SplitMEM: a graphical algorithm for pangenome analysis with suffix skips. *Bioinformatics* **30**(24), 3476–3483 (2014)
22. Melsted, P., Pritchard, J.K.: Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinform.* **12**(1), 333 (2011)
23. Minkin, I., Pham, S., Medvedev, P.: TwoPaCo: an efficient algorithm to build the compacted de Bruijn graph from many complete genomes. *Bioinformatics* btw609 (2016)

24. Murray, K.D., Webers, C., Ong, C.S., Borevitz, J.O., Warthmann, N.: kWIP: the k-mer weighted inner product, a de novo estimator of genetic similarity (2016). bioRxiv: 075481
25. Navarro, G., De Moura, E.S., Neubert, M., Ziviani, N., Baeza-Yates, R.: Adding compression to block addressing inverted indexes. *Inf. Retr.* **3**(1), 49–77 (2000)
26. Nellore, A., Collado-Torres, L., Jaffe, A.E., Alquicira-Hernández, J., Wilks, C., Pritt, J., Morton, J., Leek, J.T., Langmead, B.: Rail-RNA: scalable analysis of RNA-seq splicing and coverage. *Bioinformatics* btw575 (2016)
27. Patro, R., Mount, S.M., Kingsford, C.: Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.* **32**(5), 462–464 (2014)
28. Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 233–242. Society for Industrial and Applied Mathematics (2002)
29. Rozov, R., Shamir, R., Halperin, E.: Fast lossless compression via cascading bloom filters. *BMC Bioinform.* **15**(9), 1 (2014)
30. Salikhov, K., Sacomoto, G., Kucherov, G.: Using cascading bloom filters to improve the memory usage for de Bruijn graphs. In: Darling, A., Stoye, J. (eds.) *WABI 2013*. LNCS, vol. 8126, pp. 364–376. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40453-5_28
31. Solomon, B., Kingsford, C.: Fast search of thousands of short-read sequencing experiments. *Nat. Biotechnol.* **34**(3), 300–302 (2016)
32. Stranneheim, H., Källner, M., Allander, T., Andersson, B., Arvestad, L., Lundeberg, J.: Classification of DNA sequences using bloom filters. *Bioinformatics* **26**(13), 1595–1600 (2010)
33. Sun, C., Harris, R.S., Chikhi, R., Medvedev, P.: Allsome sequence bloom trees. bioRxiv (2016). <http://biorxiv.org/content/early/2016/12/02/090464>
34. Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D.R., Pimentel, H., Salzberg, S.L., Rinn, J.L., Pachter, L.: Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nat. Protoc.* **7**(3), 562–578 (2012)
35. Yu, Y.W., Daniels, N.M., Danko, D.C., Berger, B.: Entropy-scaling search of massive biological data. *Cell Syst.* **1**(2), 130–140 (2015)
36. Ziviani, N., de Moura, E.S., Navarro, G., Baeza-Yates, R.: Compression: a key for next-generation text retrieval systems. *IEEE Comput.* **33**(11), 37–44 (2000)

Longitudinal Genotype-Phenotype Association Study via Temporal Structure Auto-learning Predictive Model

Xiaoqian Wang¹, Jingwen Yan^{2,3}, Xiaohui Yao^{2,3}, Sungeun Kim², Kwangsik Nho², Shannon L. Risacher², Andrew J. Saykin², Li Shen², Heng Huang¹(✉), and for the ADNI

¹ Computer Science and Engineering,
University of Texas at Arlington, Arlington, TX 76019, USA
heng@uta.edu

² Radiology and Imaging Sciences,
Indiana University School of Medicine, Indianapolis, IN 46202, USA

³ BioHealth, Indiana University School of Informatics and Computing,
Indianapolis, IN 46202, USA

Abstract. With rapid progress in high-throughput genotyping and neuroimaging, imaging genetics has gained significant attention in the research of complex brain disorders, such as Alzheimer’s Disease (AD). The genotype-phenotype association study using imaging genetic data has the potential to reveal genetic basis and biological mechanism of brain structure and function. AD is a progressive neurodegenerative disease, thus, it is crucial to look into the relations between SNPs and longitudinal variations of neuroimaging phenotypes. Although some machine learning models were newly presented to capture the longitudinal patterns in genotype-phenotype association study, most of them required fixed longitudinal structures of prediction tasks and could not automatically learn the interrelations among longitudinal prediction tasks. To address this challenge, we proposed a novel temporal structure auto-learning model to automatically uncover longitudinal genotype-phenotype interrelations and utilized such interrelated structures to

H. Huang—This work was partially supported by the National Science Foundation [IIS 1302675 to H.H., IIS 1344152 to H.H., DBI 1356628 to H.H., IIS 1619308 to H.H., IIS 1633753 to H.H.] at UTA and [IIS 1622526 to L.S.] at IU; and by the National Institutes of Health [R01 LM011360 to L.S. and A.S., U01 AG024904 to Michael Weiner and A.S., RC2 AG036535 to Michael Weiner and A.S., R01 AG19771 to A.S., P30 AG10133 to A.S., UL1 TR001108 to Anantha Shekhar] and [R01 AG049371 to H.H.] at UTA.

ADNI—Data used in preparation of this article were obtained from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf.

enhance phenotype prediction in the meantime. We conducted longitudinal phenotype prediction experiments on the ADNI cohort including 3,123 SNPs and two types of imaging markers, VBM and FreeSurfer. Empirical results demonstrated advantages of our proposed model over the counterparts. Moreover, available literature was identified for our top selected SNPs, which demonstrated the rationality of our prediction results. An executable program is available online at https://github.com/littleq1991/sparse_lowRank_regression.

Keywords: Alzheimer’s Disease · Genotype-phenotype association prediction · Longitudinal study · Temporal structure auto-learning · Low-rank model

1 Introduction

As the most prevalent and severe type of neurodegenerative disorder, Alzheimer’s Disease (AD) strongly impacts human’s memory, thinking and behavior [1]. This disease is characterized by progressive impairment of memory and other cognitive abilities, triggered by the damage of neurons [2]. AD usually progresses along a temporal continuum, initially from a preclinical stage, subsequently to mild cognitive impairment (MCI) and ultimately deteriorating to AD [3]. According to [4], AD is the 6th leading cause of death in the United States. Every 66 seconds, there is someone in the United States developing AD. Up until 2016, an estimate of 5.4 million individuals in the United States are living with AD, while the number worldwide is about 44 million. To make matters worse, if no breakthrough discovered, the world will see a more striking increase in these numbers in near future.

With all these facts, AD has gained its growing attention in this day and age. Current consensus underscores the need of understanding the genetic causes of AD, with which to achieve the goal of stopping or slowing down disease progression [5]. Recent advances in neuroimaging and microbiology have provided a helping hand for exploring the associations among genes, brain structure and behavior [6]. Meanwhile, rapid developments in high-throughput genotyping have enabled the measurement of hundreds of thousands of, or even more than one million single nucleotide polymorphisms (SNPs) simultaneously [7]. These progresses have facilitated the pullulation of imaging genetics, which holds great promise for better understanding complex neurobiological systems.

In imaging genetics, an emerging strategy to facilitate identification of susceptibility genes for disorders like AD is to evaluate genetic variation using outcome-relevant biomarkers as quantitative traits (QTs). The association studies between genetic variations and imaging measures usually maintain an obvious advantage over case-control studies, as QTs are quantitative measures with the ability of increasing statistical power four to eight fold and decreasing required sample size to a large extent [8]. Numerous works have been reported to identify genetic factors impacting imaging phenotypes of biomedical importance [9–14].

In the genotype-phenotype association study, we can denote our inputs in the matrix format as follows: the SNP matrix $X \in \mathbb{R}^{d \times n}$ (n is the number or samples, d is the number of SNPs) and the imaging phenotype matrices $Y = [Y_1, Y_2, \dots, Y_T] \in \mathbb{R}^{n \times cT}$ (c is the number of QTs, T is the number of time points, and $Y_t \in \mathbb{R}^{n \times c}$ is the phenotype matrix at time t). The goal is to find the weight matrix $W = [W_1, W_2, \dots, W_T] \in \mathbb{R}^{d \times cT}$, which properly reflect the relations between SNPs and QTs and capture a subset of SNPs responsible for phenotype prediction at the same time. If we treat the prediction of one phenotype at one time point as a task, then the association study between genotypes with multiple longitudinal phenotypes can be seen as a multi-task learning problem.

Conventional strategies [15–17], which perform standard regression at all time points, are equivalent to carrying out regression at individual time point separately, thus ignore the longitudinal variations of brain phenotypes. Since AD is a progressive disorder and imaging phenotype is a quantitative reflect of its neurodegenerative status, prediction tasks at various time points can be reasonably assumed related. For a certain QT, its value at different time may be correlated, while distinct QTs at a certain time may also retain some mutual influence. To excavate correlations among longitudinal prediction tasks, several multi-task models were proposed on the basis of sparse learning [18, 19]. The main idea of these models is to exert trace norm on the entire parameter matrix, such that the common subspace globally shared by different prediction tasks can be extracted.

However, longitudinal prediction tasks are interrelated as different groups, not as a whole. Existing methods cannot find the task interrelations properly. It is intractable to discover such task group structure. One straightforward way of capturing such interrelated groups is to conduct clustering analysis and extract the group structure as a preprocessing step. Nevertheless, such a heuristic step is independent to the entire longitudinal learning model, thus the detected group structures are not optimal for the longitudinal learning. To bridge this gap, we propose a novel temporal structure auto-learning low-rank predictive model to simultaneously uncover the interrelations among different prediction tasks and utilize the learned interrelated structures to enhance phenotype prediction.

Notations: In this paper, matrices are all written as uppercase letters while vectors as bold lower case letters. For a matrix $M \in \mathbb{R}^{d \times n}$, its i -th row and j -th column are denoted by \mathbf{m}^i , \mathbf{m}_j respectively, while its ij -th element is written as m_{ij} or $M(i, j)$. For a positive value p , the $\ell_{2,p}$ -norm of M is defined as $\|M\|_{2,p} = (\sum_{i=1}^d (\sum_{j=1}^n m_{ij}^2)^{\frac{p}{2}})^{\frac{1}{p}} = (\sum_{i=1}^d (\|\mathbf{m}^i\|_2)^p)^{\frac{1}{p}}$.

2 Temporal Structure Auto-learning Predictive Model

2.1 Illustration of Our Idea

Our main purpose is to construct a model to simultaneously detect the latent group structure of longitudinal phenotype prediction tasks and study SNP associations across all endophenotypes. As is shown in Fig. 1, the four phenotypes

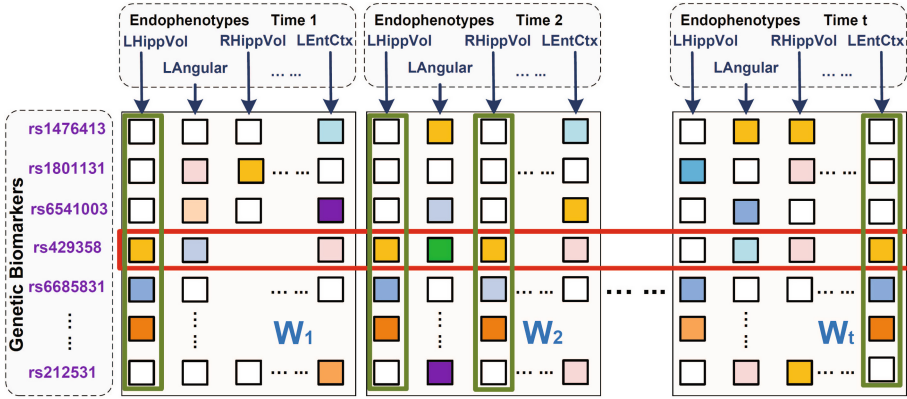


Fig. 1. Illustration of our temporal structure auto-learning regression model. In this figure, parameter matrices at different time points are arranged in the column order. Four tasks marked out by green rectangles obey similar patterns thus have high interrelations while one SNP loci rs429358 enclosed by a red rectangle appears to be correlated with most phenotypes. Our model is meant to uncover the group information among all prediction tasks along the time continuum, *i.e.*, cluster the phenotypes in the same latent subspace (phenotypes marked out by green rectangles) into the same group and meanwhile discover genetic biomarkers responsible for most prediction tasks (SNPs marked out by red rectangles).

marked by the green rectangles have similar distributions and are very likely to be correlated. However, previous models are not capable of capturing such inter-linked structures in different task groups. In our model, we expect to uncover such latent subspaces in different groups via low-rank constraints. Meanwhile, the SNP loci marked by the red rectangle, rs429358, appears to be predominant for most response variables. Correspondingly, we impose a sparsity constraint to pick it out. In consequence, our model should be able to capture the group structure within the prediction tasks and utilize this information to select prominent SNPs across the relevant phenotypes. In the next subsection, we will elaborate how to translate these ideas into the new learning model.

2.2 New Objective Function

To discover the group structure of phenotype prediction tasks, we introduce and optimize a group index matrix set Q . Suppose the tasks come from g groups, then we have $Q = \{Q_1, Q_2, \dots, Q_g\}$, where Q_i is a diagonal matrix and $Q_i \in \{0, 1\}^{cT \times cT}$. For each Q_i , $(Q_i)_{(k,k)} = 1$ means the k -th feature belongs to the i -th group while $(Q_i)_{(k,k)} = 0$ means not. To avoid overlap of subspaces, we maintain the constraint that $\sum_{i=1}^g Q_i = I$.

On the other hand, since SNPs are often correlated and have an overlap in impacting phenotypes, we impose low-rank constraint to uncover the common subspaces shared by prediction tasks. The traditional method to impose

low-rank constraint is minimizing trace norm, which is a convex relaxation of rank minimization problem. However, trace norm is not an ideal approximation of the rank minimization. Here, we use the Schatten p -norm regularization term instead, which approximates the rank minimization better than trace norm [20]. The definition of Schatten p -norm of a matrix $M \in \mathbb{R}^{m \times n}$ is:

$$\|M\|_{S_p} = (Tr((M^T M)^{\frac{p}{2}}))^{\frac{1}{p}} = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p \right)^{\frac{1}{p}}, \tag{1}$$

where σ_i is the i -th singular value of M . Specially, when $p = 1$, the Schatten p -norm of M is exactly the trace norm since $\|M\|_* = \|M\|_{S_1} = Tr((M^T M)^{\frac{1}{2}}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i$. As we recall, the rank of M can be denoted as $rank(M) = \sum_{i=1}^{\min\{m,n\}} \sigma_i^0$, where $0^0 = 0$. Thus, when $0 < p < 1$, Schatten p -norm is a better low-rank regularization than trace norm.

Moreover, since we intend to integrate the SNP selection procedure across multiple learning tasks, here we impose a sparsity constraint. One possible approach is $\ell_{2,1}$ -norm regularization [21], which is popularly utilized owing to its convex property. However, the real data usually don't satisfy the RIP condition, thus the solution of $\ell_{2,1}$ -norm may not be sparse enough. See [22, 23] for details. To solve this problem, in our model, we resort to a stricter sparsity constraint, $\ell_{2,0+}$ -norm, which is defined as follows:

$$\|M\|_{2,q} = \left(\sum_{i=1}^d \left(\sum_{j=1}^n m_{ij}^2 \right)^{\frac{q}{2}} \right)^{\frac{1}{q}} = \left(\sum_{i=1}^d (\|\mathbf{m}^i\|_2^q) \right)^{\frac{1}{q}},$$

where q is a positive value. Similarly to the previous discussion, when $0 < q < 1$, $\ell_{2,0+}$ -norm can achieve a more sparse solution than $\ell_{2,1}$ norm.

All in all, we propose a new temporal structure auto-learning model:

$$\begin{aligned} \min_{W, Q_i|_{i=1}^g} & \|X^T W - Y\|_F^2 + \gamma_1 \sum_{i=1}^g (\|W Q_i\|_{S_p}^p)^k + \gamma_2 \|W\|_{2,0+}, \\ \text{s.t. } & Q_i|_{i=1}^g \in \{0, 1\}^{cT \times cT}, \sum_{i=1}^g Q_i = I. \end{aligned} \tag{2}$$

In Eq. (2), we adopt the k -power of Schatten p -norm to make our model more robust. The use of parameter k will be articulated in Sect. 4. Since it is difficult to solve the proposed new non-convex and non-smooth objective, in the next section we propose a novel alternating optimization method for Problem (2).

3 Optimization Algorithm

In this section, we first introduce an optimization algorithm for general problems with Problem (2) being a special case, and then discuss the detailed optimization steps of Problem (2).

Lemma 1. Let $g_i(x)$ denote a general function over x , where x can be a scalar, vector or matrix, then we can claim:

When $\delta \rightarrow 0$, The optimization problem

$$\min_{x \in \mathcal{C}} f(x) + \sum_i Tr((g_i^T(x)g_i(x))^{\frac{p}{2}})$$

is equivalent to

$$\min_{x \in \mathcal{C}} f(x) + \sum_i Tr(g_i^T(x)g_i(x)D_i), \quad \text{where } D_i = \frac{p}{2}(g_i^T(x)g_i(x) + \delta I)^{\frac{p-2}{2}}. \quad (3)$$

Proof. When $\delta \rightarrow 0$, it's apparent that the optimization problem

$$\min_{x \in \mathcal{C}} f(x) + \sum_i Tr((g_i^T(x)g_i(x) + \delta I)^{\frac{p}{2}}) \quad (4)$$

will reduce to

$$\min_{x \in \mathcal{C}} f(x) + \sum_i Tr((g_i^T(x)g_i(x))^{\frac{p}{2}}). \quad (5)$$

So we turn the non-smooth Problem (5) to the smooth Problem (4) where δ is fairly small.

The Lagrangian function of Problem (4) is:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_i Tr((g_i^T(x)g_i(x) + \delta I)^{\frac{p}{2}}) - r(x, \lambda), \quad (6)$$

where $r(x, \lambda)$ is a Lagrangian term for the domain constraint $x \in \mathcal{C}$. Take derivative w.r.t. x and set it to zero, we have:

$$f'(x) + \sum_i \frac{\partial Tr((g_i^T(x)g_i(x) + \delta I)^{\frac{p}{2}})}{\partial x} - \frac{\partial r(x, \lambda)}{\partial x} = 0. \quad (7)$$

Based on the chain rule [24], Eq. (7) can be rewritten as:

$$f'(x) + \sum_i \frac{tr\left(2\frac{p}{2}(g_i^T(x)g_i(x) + \delta I)^{\frac{p-2}{2}}g_i^T(x)\partial g_i(x)\right)}{\partial x} - \frac{\partial r(x, \lambda)}{\partial x} = 0. \quad (8)$$

According to the Karush-Kuhn-Tucker conditions [25], if we can find a solution x that satisfies Eq. (8), then we usually find a local/global optimal solution to Problem (4). However, it is intractable to directly find the solution x that satisfies Eq. (8). Here we come up with a strategy as follows:

If we define $D_i = \frac{p}{2}(g_i^T(x)g_i(x) + \delta I)^{\frac{p-2}{2}}$ as a given constant, then Eq. (8) can be reduced to

$$f'(x) + \sum_i \frac{tr(2D_i g_i^T(x)\partial g_i(x))}{\partial x} - \frac{\partial r(x, \lambda)}{\partial x} = 0. \quad (9)$$

Based on the chain rule [24], the optimal solution x^* of Eq. (9) is also an optimal solution to the following problem:

$$\min_{x \in \mathcal{C}} f(x) + \sum_i Tr(g_i^T(x)g_i(x)D_i). \tag{10}$$

Based on this observation, we can first guess a solution x , next calculate D_i based on the current solution x , and then update the current solution x by the optimal solution of Problem (10) on the basis of the calculated D_i . We can iteratively perform this procedure until it converges. \square

According to Lemma 1 and the property of Q_i that $Q_i Q_i^T = Q_i$, Problem (2) is equivalent to:

$$\begin{aligned} \min_{W, Q_i |_{i=1}^g} & \|X^T W - Y\|_F^2 + \gamma_1 \sum_{i=1}^g Tr(WQ_i W^T D_i) + \gamma_2 Tr(WW^T B) \\ \text{s.t.} & \quad Q_i |_{i=1}^g \in \{0, 1\}^{cT \times cT}, \sum_{i=1}^g Q_i = I, \end{aligned} \tag{11}$$

where D_i is defined as:

$$D_i = \frac{kp}{2} (\|WQ_i\|_{S_p}^p)^{k-1} (WQ_i W^T + \delta_1 I)^{\frac{p-2}{2}}, \tag{12}$$

and B is defined as a diagonal matrix with the l -th diagonal element to be:

$$b_{ll} = \frac{q}{2} (\mathbf{w}^l (\mathbf{w}^l)^T + \delta_2 I)^{\frac{q-2}{2}}, \tag{13}$$

and δ_1 and δ_2 are two fairly small parameters close to zero.

We can solve Problem (11) via alternating optimization.

The first step is fixing W and solving Q , then Problem (11) becomes:

$$\min_{Q_i |_{i=1}^g} \sum_{i=1}^g Tr((W^T D_i W) Q_i) \quad \text{s.t.} \quad Q_i |_{i=1}^g \in \{0, 1\}^{cT \times cT}, \sum_{i=1}^g Q_i = I.$$

Let $A_i = W^T D_i W$, then the solution of each Q_i is as follows:

$$Q_i(k, k) = \begin{cases} 1, & i = \arg \min_j A_j(k, k) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

The second step is fixing Q and solving W . Problem (11) becomes:

$$\min_W \|X^T W - Y\|_F^2 + \gamma_1 \sum_{i=1}^g Tr(WQ_i W^T D_i) + \gamma_2 Tr(WW^T B),$$

which can be further decoupled for each column of W as follows:

$$\min_{\mathbf{w}_k} \|X^T \mathbf{w}_k - \mathbf{y}_k\|_2^2 + \gamma_1 \sum_{i=1}^g (Q_i(k, k) \mathbf{w}_k^T D_i \mathbf{w}_k) + \gamma_2 \mathbf{w}_k^T B \mathbf{w}_k.$$

Taking derivative *w.r.t.* \mathbf{w}_k in the above problem and set it to zero, we get:

$$\mathbf{w}_k = (XX^T + \gamma_1 \left(\sum_{i=1}^g Q_i(k, k) D_i \right) + \gamma_2 B)^{-1} X \mathbf{y}_k. \tag{15}$$

We can iteratively update D , Q , B and W with the alternative steps mentioned above and the algorithm for Problem (11) is summarized in Algorithm 1.

Algorithm 1. Algorithm to solve problem (11).

Input:

SNP matrix $X \in \mathbb{R}^{d \times n}$, longitudinal phenotype matrix $Y = [Y_1 \ Y_2 \ \dots \ Y_T]$ where $Y_t|_{t=1}^T \in \mathbb{R}^{n \times c}$, parameter $\delta_1 = 10^{-12}$ and $\delta_2 = 10^{-12}$, number of groups g .

Output:

Weight matrix $W = [W_1 \ W_2 \ \dots \ W_T]$ where $W_t|_{t=1}^T \in \mathbb{R}^{d \times c}$ and g different group matrices $Q_i|_{i=1}^g \in \mathbb{R}^{c^T \times c^T}$ which groups the tasks into exactly g subspaces.

Initialize W by the optimal solution to the ridge regression problem:

$$\min_W \|W^T X - Y\|_F^2 + \|W\|_F^2$$

Initialize Q matrices randomly

while not converge **do**

1. Update D according to the definition in Eq. (12).
2. Update Q according to the solution in Eq. (14)
3. Update B according to the definition in Eq. (13).
4. Update W , where the solution to the k -th column of W is displayed in Eq. (15).

end while

Convergence analysis: Our algorithm uses the alternating optimization method to update variables, whose convergence has already been proved in [26]. In our model, variables in each iteration has a closed form solution and can be computed fairly fast. In most cases, our method converges within 10 iterations.

4 Discussion of Parameters

In our model, we introduced several parameters to make it more general and adaptive to various circumstances. Here we analyze the functionality of each parameter in detail.

In Problem (2), parameter p and q are norm parameters proposed for the two regularization terms. For p , it adjusts the stringency of the low-rank constraint. As analyzed in Sect. 3, Schatten p -norm makes a stricter low-rank constraint than trace norm when $0 < p < 1$. The closer p is to 0, the more rigorous low-rank constraint the regularization term $\|M\|_{S_p}^p$ imposes. The rationale is similar for parameter q . The $\ell_{2,0+}$ -norm is a better approximation of $\ell_{2,0}$ -norm than $\ell_{2,1}$ -norm when q lies in the range of $(0, 1)$, thus makes our learned parameter matrix more sparse.

In the low rank regularization term $\|M\|_{S_p}^p$, when p is small, the number of local solutions becomes more thus lead our model (2) to be more sensitive to outliers. Under this condition, we use k -power of this term as $(\|M\|_{S_p}^p)^k$ to make our model robust. According to experimental experience, the value of k can be determined in the range of [2, 3].

The parameters γ_1 and γ_2 are proposed to balance the importance of two regularization terms. Larger γ_1 lead to more attention on the low-rank constraint while larger γ_2 lays more emphasis on the sparse structure. These two parameters can be adjusted to accommodate different cases.

In our empirical section, we didn't make too much effort on tuning these parameters. Instead, in fairness to other comparing methods, we just simply set each parameters to a reasonable value. Though these parameters brought about great challenges in solving our optimization problem, they make our model more flexible and adaptive to different settings and conditions.

5 Experimental Results

In this section, we evaluate the prediction performance of our proposed method using both synthetic and real data. Our goal is to uncover the latent subspace structure of the prediction tasks and meanwhile select a subset of SNPs responsible for their variation.

5.1 Experiments on Synthetic Data

First of all, we utilize synthetic data to illustrate the effectiveness of our model. Our synthetic data is composed of 30 features and 3 groups of tasks from 4 different time points. Each group includes 4 tasks, who are identical to each other up to a scaling. After we generated weight matrix W in this way, we constructed a random X including 10000 samples and get Y according to $Y = X^T W$.

Figure 2(a) shows the original W_o matrix, where weight matrices in different time points are arranged in a column order. According to the construction process of W_o , tasks in W_o should form three low-rank subspaces. For easier visualizing this low-rank structure, we reshuffled tasks in W_o by putting tasks in the same group together and formed Fig. 2(b). Now the low-rank structure within the synthetic data can be easily detected, where every four columns in Fig. 2(b) make a low-rank subspace. We applied our method to this synthetic data and plotted our learned W matrix in Fig. 2(c). To evaluate the structure of W , we did the rearrangement likewise. By comparing Fig. 2(b) and (d), we note that our method has successfully uncover the group structure of the synthetic data and recovered the parameter matrix W in an accurate way.

5.2 Experimental Settings on Real Benchmark Data

In the following we evaluate our method on real benchmark datasets. We compare our method with all the counterparts discussed in the introduction section,

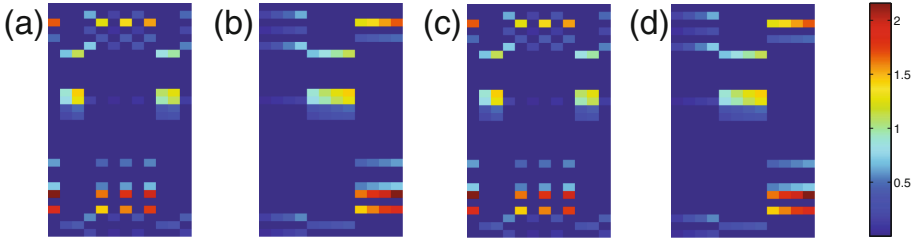


Fig. 2. Visualization of the synthetic parameter matrix W . Columns of W denote 12 prediction tasks from 4 different time points, while rows of W correspond to 30 features. These 12 tasks are equally divided into 3 groups, where tasks in the same group are identical to each other up to a scaling factor. (a) The original weight matrix W_o . (b) Rearrangement of columns in W_o by putting tasks in the same group together, such that the low-rank structure of W_o is more clear. (c) The learned W matrix by our model. (d) Rearrangement of W learned by our model.

which are: multivariate Linear Regression (LR), multivariate Ridge Regression (RR), Multi-Task Trace-norm regression (MTT) [27], Multi-Task $\ell_{2,1}$ -norm Regression (MTL21) [28] and their combination (MTT+L21) [18, 19].

In our pre-experiments, we found the performance of our method to be relatively stable with parameters in the reasonable range (data not shown). For simplicity, we set $\gamma_1 = 1$, $\gamma_2 = 1$, $p = 0.8$, $q = 0.1$, and $k = 2.5$ without tuning. The definition and reasonable range of these parameters has been discussed in Sect. 4.

As the evaluation metric, we reported the root mean square error (RMSE) and correlation coefficient (CorCoe) between the predicted and actual scores in out-of-sample prediction. In our experiment, the RMSE was normalized by the Frobenius norm of the ground truth matrix. Better performance relates with lower RMSE or higher CorCoe value. We utilized the 5-fold cross validation technique and reported the average RMSE and CorCoe on these 5 trials for each method.

5.3 Description of ADNI Data

Data used in this work were obtained from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). One goal of ADNI is to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early AD. For up-to-date information, see www.adni-info.org. The genotype data [29] of all non-Hispanic Caucasian participants from the ADNI Phase 1 cohort were used here. They were genotyped using the Human 610-Quad BeadChip. Among all the SNPs, only SNPs, within the boundary of $\pm 20K$ base pairs of the 153 AD candidate genes listed on the AlzGene database (www.alzgene.org) as of 4/18/2011 [30], were selected after the standard quality control (QC)

and imputation steps. The QC criteria for the SNP data include (1) call rate check per subject and per SNP marker, (2) gender check, (3) sibling pair identification, (4) the Hardy-Weinberg equilibrium test, (5) marker removal by the minor allele frequency and (6) population stratification. As the second pre-processing step, the QC'ed SNPs were imputed using the MaCH software [31] to estimate the missing genotypes. As a result, our analyses included 3,576 SNPs extracted from 153 genes (boundary: $\pm 20KB$) using the ANNOVAR annotation (<http://www.openbioinformatics.org/annovar/>).

Two widely employed automated MRI analysis techniques were used to process and extract imaging phenotypes from scans of ADNI participants as previously described [11]. First, Voxel-Based Morphometry (VBM) [32] was performed to define global gray matter (GM) density maps and extract local GM density values for 90 target regions. Second, automated parcellation via FreeSurfer V4 [33] was conducted to define volumetric and cortical thickness values for 90 regions of interest (ROIs) and to extract total intracranial volume (ICV). Further details are available in [11]. All these measures were adjusted for the baseline ICV using the regression weights derived from the healthy control (HC) participants. The time points examined in this study for imaging markers included baseline (BL), Month 6 (M6), Month 12 (M12) and Month 24 (M24). All the participants with no missing BL/M6/M12/M24 MRI measurements were included in this study, including 96 AD samples, and 219 MCI samples and 174 health control (HC) samples.

5.4 Performance Comparison on ADNI Cohort

Here we assessed the ability of our method to predict a set of imaging biomarkers via genetic variations. We tracked the process along the time axis and intended to uncover the latent subspace structure maintained by phenotypes and meanwhile capture a subset of SNPs influencing the phenotypes in a certain subspace.

We examined the cases where the number of selected SNPs were $\{30, 40, \dots, 80\}$. The experimental results are summarized in Tables 1 and 2. We observe that our method consistently outperforms other methods in most cases. The reasons go as follows: multivariate regression and ridge regression assumed the imaging features at different time points to be independent, thus didn't consider the correlations within. Their neglects of the interrelations among the data was detrimental to their prediction performance.

As for MTTrace, MTL21 and their combination MTTrace+MTL21, even though they take into account the inner connection information of imaging phenotypes, they simply constrain all phenotypes to a global space thus cannot handle the possible group structure therein. That is why they may overweigh the standard methods in some cases, but cannot outperform our proposed method. As for our proposed method, not only did we capture the latent structure among the longitudinal phenotypes, but we also selected a set of responsible SNPs at the same time. All in all, our model can capture SNPs responsible for some but not necessarily all imaging phenotypes along the time continuum, which save more effective information in the prediction.

Table 1. Biomarker “VBM” and “FreeSurfer” (upper table for “VBM”, lower table for “FreeSurfer”) prediction comparison via root mean square error (RMSE) measurement with different number of selected SNPs. Better performance corresponds to lower RMSE.

# of SNPs	LR	RR	MTT	MTL21	MTT+L21	OURS
30	0.9277 ± 0.0122	0.9278 ± 0.0122	0.9270 ± 0.0135	0.9278 ± 0.0122	0.9276 ± 0.0126	0.9266 ± 0.0124
40	0.9106 ± 0.0147	0.9100 ± 0.0146	0.9099 ± 0.0141	0.9100 ± 0.0146	0.9099 ± 0.0141	0.9061 ± 0.0087
50	0.8914 ± 0.0177	0.8916 ± 0.0176	0.8934 ± 0.0159	0.8916 ± 0.0176	0.8919 ± 0.0176	0.8913 ± 0.0084
60	0.8843 ± 0.0216	0.8846 ± 0.0216	0.8831 ± 0.0192	0.8846 ± 0.0216	0.8848 ± 0.0215	0.8726 ± 0.0051
70	0.8661 ± 0.0204	0.8663 ± 0.0203	0.8675 ± 0.0211	0.8663 ± 0.0203	0.8666 ± 0.0202	0.8615 ± 0.0045
80	0.8509 ± 0.0250	0.8503 ± 0.0244	0.8500 ± 0.0242	0.8503 ± 0.0244	0.8500 ± 0.0242	0.8482 ± 0.0042
30	0.9667 ± 0.0145	0.9664 ± 0.0145	0.9667 ± 0.0145	0.9664 ± 0.0145	0.9664 ± 0.0146	0.9558 ± 0.0147
40	0.9569 ± 0.0113	0.9569 ± 0.0112	0.9569 ± 0.0113	0.9569 ± 0.0112	0.9569 ± 0.0114	0.9436 ± 0.0113
50	0.9502 ± 0.0141	0.9503 ± 0.0141	0.9502 ± 0.0141	0.9503 ± 0.0141	0.9503 ± 0.0142	0.9350 ± 0.0143
60	0.9416 ± 0.0106	0.9417 ± 0.0106	0.9416 ± 0.0106	0.9417 ± 0.0106	0.9415 ± 0.0107	0.9234 ± 0.0106
70	0.9319 ± 0.0105	0.9316 ± 0.0105	0.9319 ± 0.0105	0.9316 ± 0.0105	0.9321 ± 0.0106	0.9096 ± 0.0106
80	0.9246 ± 0.0093	0.9247 ± 0.0093	0.9246 ± 0.0093	0.9247 ± 0.0093	0.9247 ± 0.0094	0.9012 ± 0.0094

Table 2. Biomarker “VBM” and “FreeSurfer” (upper table for “VBM”, lower table for “FreeSurfer”) prediction comparison via correlation coefficient (CorCoe) measurement with different number of selected SNPs. Better performance corresponds to higher CorCoe.

# of SNPs	LR	RR	MTT	MTL21	MTT+L21	OURS
30	0.4186 ± 0.0092	0.4180 ± 0.0098	0.4232 ± 0.0088	0.4180 ± 0.0098	0.4216 ± 0.0077	0.6193 ± 0.0177
40	0.4284 ± 0.0163	0.4282 ± 0.0166	0.4333 ± 0.0181	0.4282 ± 0.0166	0.4312 ± 0.0157	0.6294 ± 0.0071
50	0.4476 ± 0.0395	0.4470 ± 0.0400	0.4526 ± 0.0384	0.4470 ± 0.0400	0.4508 ± 0.0385	0.6362 ± 0.0094
60	0.4477 ± 0.0391	0.4471 ± 0.0397	0.4513 ± 0.0388	0.4471 ± 0.0397	0.4510 ± 0.0380	0.6435 ± 0.0196
70	0.4502 ± 0.0345	0.4490 ± 0.0356	0.4547 ± 0.0335	0.4490 ± 0.0356	0.4529 ± 0.0339	0.6460 ± 0.0129
80	0.4467 ± 0.0287	0.4470 ± 0.0286	0.4514 ± 0.0271	0.4470 ± 0.0286	0.4508 ± 0.0268	0.6521 ± 0.0083
30	0.9007 ± 0.0145	0.9008 ± 0.0145	0.9007 ± 0.0145	0.9008 ± 0.0145	0.9010 ± 0.0146	0.9019 ± 0.0147
40	0.9049 ± 0.0113	0.9051 ± 0.0112	0.9049 ± 0.0113	0.9051 ± 0.0112	0.9052 ± 0.0114	0.9060 ± 0.0113
50	0.9045 ± 0.0141	0.9046 ± 0.0141	0.9045 ± 0.0141	0.9046 ± 0.0141	0.9048 ± 0.0142	0.9057 ± 0.0143
60	0.9079 ± 0.0106	0.9080 ± 0.0106	0.9079 ± 0.0106	0.9080 ± 0.0106	0.9081 ± 0.0107	0.9089 ± 0.0106
70	0.9094 ± 0.0105	0.9096 ± 0.0105	0.9094 ± 0.0105	0.9096 ± 0.0105	0.9097 ± 0.0106	0.9106 ± 0.0106
80	0.9114 ± 0.0093	0.9115 ± 0.0093	0.9114 ± 0.0093	0.9115 ± 0.0093	0.9117 ± 0.0094	0.9124 ± 0.0094

5.5 Identification of Top Selected SNPs

Shown in Fig. 3 are the heat maps of top regression weights between imaging QTs and SNPs. APOE-rs429358, the well-known major AD risk factor, demonstrated relatively strong predictive power in both analysis: (1) In FreeSurfer analysis, it predicts mainly the cerebral cortex volume at M06, M12 and M24. (2) In VBM analysis, it predicts the GM densities of amygdala, hippocampus, and parahippocampal gyrus at multiple time points (Fig. 4). Both patterns are reassuring, since APOE-rs429358 and atrophy patterns of the entire cortex as well as medial temporal regions (including amygdala, hippocampus, and parahippocampal gyrus) are all highly associated with AD.

In addition, APOE-rs429358 has been shown to be related to medial temporal lobe atrophy [34], hippocampal atrophy [35] and cortical atrophy [36]. Variants within membrane-spanning 4-domains subfamily A (MS4A) gene

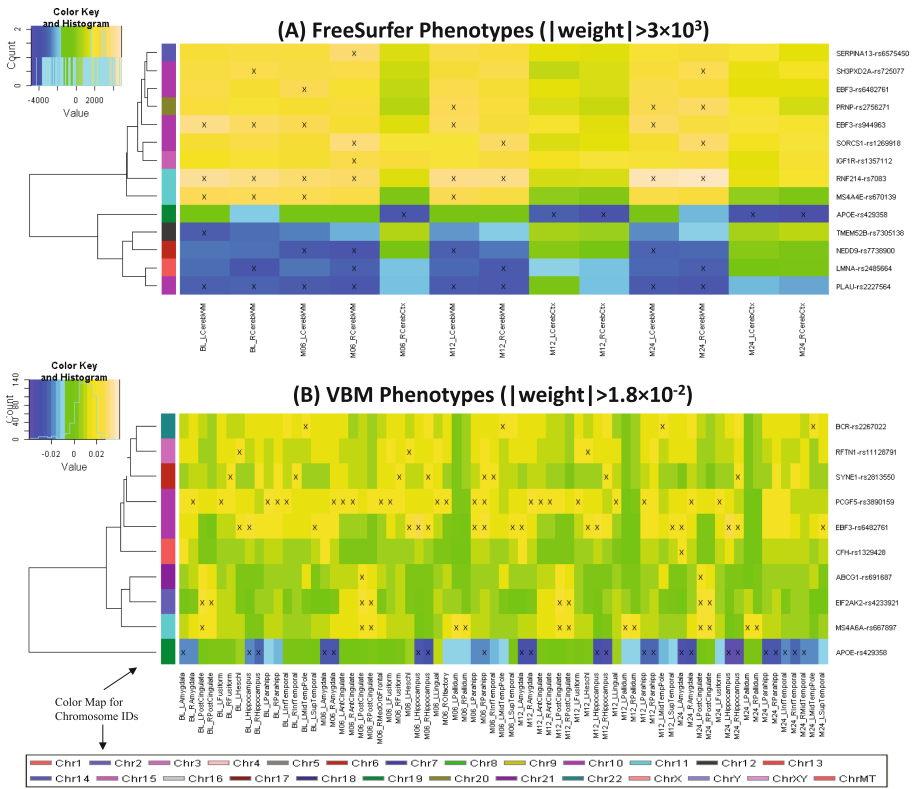


Fig. 3. Heat maps of top regression weights between quantitative traits (QTs) and SNPs filtered by a user-specified weight threshold: (A) FreeSurfer QTs and (B) VBM QTs. Weights from each regression analysis are color-mapped and displayed in the heat maps. Heat map blocks labeled with “x” reach the weight threshold. Only top SNPs and QTs are included in the heat maps, and so each row (SNP) and column (QT) have at least one “x” block. Dendrograms derived from hierarchical clustering are plotted for SNPs. The color bar on the left side of the heat map codes the chromosome IDs for the corresponding SNPs.

cluster are some other recently discovered AD risk factors [37]. Our analysis also demonstrated their associations with imaging QTs: (1) MS4A4E-rs670139 predicts cerebral white matter volume at BL, M06 and M12; and (2) MS4A6A-rs667897 predicts GM densities of posterior cingulate and pallidum at almost all time points. Another interesting finding is EBF3-rs482761, which is identified in both analyses: (1) In FreeSurfer analysis, it predicts left cerebral white matter volume at M06; and (2) in VBM analysis, it predicts GM densities of left Heschl’s gyri, left superior temporal gyri, hippocampus, left amygdala at multiple time points. EBF3 (early B-cell factor 3) is a protein-coding gene and has been associated with neurogenesis and glioblastoma. In general, both FreeSurfer and VBM analyses picked up similar regions across different time points. These

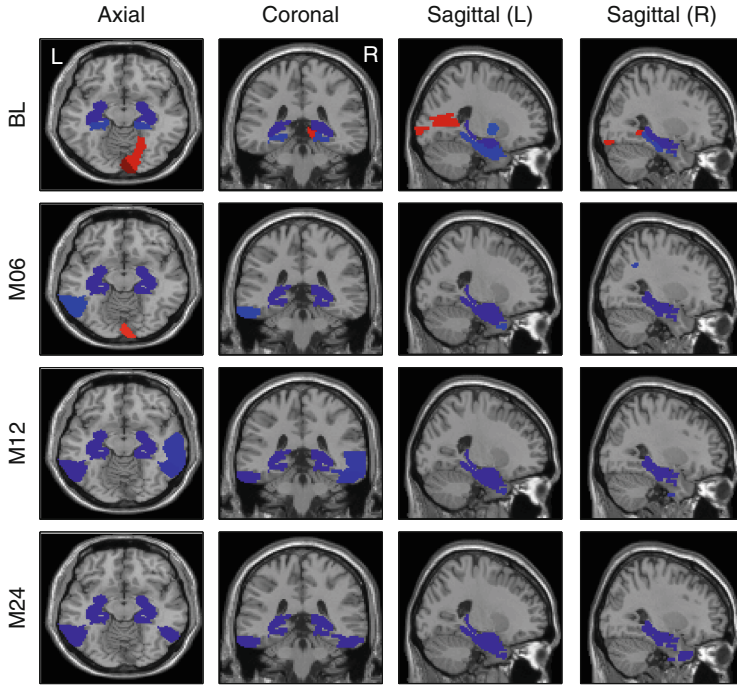


Fig. 4. Top 10 weights of APOE-rs429358 mapped on the brain for the VBM analysis.

identified imaging genomic associations warrant further investigation in independent cohorts. If replicated, these findings can potentially contribute to biomarker discovery for diagnosis and drug design.

6 Conclusions

In this paper, we proposed a novel temporal structure auto-learning model to study the associations between genetic variations and longitudinal imaging phenotypes. Our model can simultaneously uncover the interrelation structures existing in different prediction tasks and utilize such learned interrelated structures to enhance the feature learning model. Moreover, we utilized the Schatten p -norm to extract the common subspace shared by the prediction tasks. Our new model is applied to ADNI cohort for neuroimaging phenotypes prediction via SNPs. We conducted experiments on both synthetic and real benchmark data. Empirical results validated the effectiveness of our model by demonstrating the improved prediction performance compared with related methods. In real data analysis, we also identified a set of interesting and biologically meaningful imaging genomic associations, showing the potential for biomarker discovery in disease diagnosis and drug design.

References

1. Khachaturian, Z.S.: Diagnosis of Alzheimer's disease. *Arch. Neurol.* **42**(11), 1097–1105 (1985)
2. Burns, A., Iliffe, S.: Alzheimer's disease. *BMJ* **338**, b158 (2009)
3. Wenk, G.L., et al.: Neuropathologic changes in Alzheimer's disease. *J. Clin. Psychiatry* **64**, 7–10 (2003)
4. Association, A., et al.: 2016 Alzheimer's disease facts and figures. *Alzheimer's Dement.* **12**(4), 459–509 (2016)
5. Petrella, J.R., Coleman, R.E., Doraiswamy, P.M.: Neuroimaging and early diagnosis of Alzheimer disease: a look to the future 1. *Radiology* **226**(2), 315–336 (2003)
6. Hariri, A.R., Drabant, E.M., Weinberger, D.R.: Imaging genetics: perspectives from studies of genetically driven variation in serotonin function and corticolimbic affective processing. *Biol. Psychiatry* **59**(10), 888–897 (2006)
7. Potkin, S.G., Guffanti, G., et al.: Hippocampal atrophy as a quantitative trait in a genome-wide association study identifying novel susceptibility genes for Alzheimer's disease. *PLoS One* **4**(8), e6501 (2009)
8. Potkin, S.G., Turner, J.A., et al.: Genome-wide strategies for discovering genetic influences on cognition and cognitive disorders: methodological considerations. *Cogn. Neuropsychiatry* **14**(4–5), 391–418 (2009)
9. Harold, D., Abraham, R., et al.: Genome-wide association study identifies variants at *CLU* and *PICALM* associated with Alzheimer's disease. *Nat. Genet.* **41**(10), 1088–1093 (2009)
10. Bis, J.C., DeCarli, C., et al.: Common variants at 12q14 and 12q24 are associated with hippocampal volume. *Nat. Genet.* **44**(5), 545–551 (2012)
11. Shen, L., Kim, S., et al.: Whole genome association study of brain-wide imaging phenotypes for identifying quantitative trait loci in MCI and AD: a study of the ADNI cohort. *Neuroimage* **53**(3), 1051–1063 (2010)
12. Wang, H., Nie, F., Huang, H., Risacher, S., Saykin, A.J., Shen, L.: ADNI: joint classification and regression for identifying ad-sensitive and cognition-relevant imaging biomarkers. In: *The 14th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 115–123 (2011)
13. Wang, H., Nie, F., Huang, H., Kim, S., Nho, K., Risacher, S.L., Saykin, A.J., Shen, L.: Identifying quantitative trait loci via group-sparse multitask regression and feature selection: an imaging genetics study of the ADNI cohort. *Bioinformatics* **28**(2), 229–237 (2012)
14. Wang, H., Nie, F., Huang, H., Risacher, S.L., Saykin, A.J., Shen, L.: ADNI: identifying disease sensitive and quantitative trait relevant biomarkers from multi-dimensional heterogeneous imaging genetics data via sparse multi-modal multitask learning. *Bioinformatics* **28**(12), i127–i136 (2012). [20th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)]
15. Ashford, J.W., Schmitt, F.A.: Modeling the time-course of Alzheimer dementia. *Curr. Psychiatry Rep.* **3**(1), 20–28 (2001)
16. Sabatti, C., Service, S.K., et al.: Genome-wide association analysis of metabolic traits in a birth cohort from a founder population. *Nat. Genet.* **41**(1), 35–46 (2008)
17. Kim, S., Sohn, K.A., et al.: A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics* **25**(12), i204–i212 (2009)
18. Wang, H., Nie, F., Huang, H., et al.: From phenotype to genotype: an association study of longitudinal phenotypic markers to Alzheimer's disease relevant SNPs. *Bioinformatics* **28**(18), i619–i625 (2012)

19. Wang, H., Nie, F., et al.: High-order multi-task feature learning to identify longitudinal phenotypic markers for Alzheimer's disease progression prediction. In: *Advances in Neural Information Processing Systems*, pp. 1277–1285 (2012)
20. Nie, F., Huang, H., Ding, C.H.: Low-rank matrix recovery via efficient Schatten p -norm minimization. In: *AAAI (2012)*
21. Obozinski, G., Taskar, B., Jordan, M.: Multi-task feature selection. Technical report, Statistics Department, UC Berkeley (2006)
22. Candès, E.J.: The restricted isometry property and its implications for compressed sensing. *C. R. Math.* **346**(9), 589–592 (2008)
23. Cai, T.T., Zhang, A.: Sharp RIP bound for sparse signal and low-rank matrix recovery. *Appl. Comput. Harmon. Anal.* **35**(1), 74–93 (2013)
24. Bentler, P., Lee, S.Y.: Matrix derivatives with chain rule and rules for simple, Hadamard, and Kronecker products. *J. Math. Psychol.* **17**(3), 255–262 (1978)
25. Rockafellar, R.T.: *Convex Analysis*. Princeton Mathematical Series. Princeton University Press, Princeton (1970)
26. Bezdek, J.C., Hathaway, R.J.: Convergence of alternating optimization. *Neural Parallel Sci. Comput.* **11**(4), 351–368 (2003)
27. Ji, S., Ye, J.: An accelerated gradient method for trace norm minimization. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 457–464. ACM (2009)
28. Evgeniou, A., Pontil, M.: Multi-task feature learning. *Adv. Neural Inf. Process. Syst.* **19**, 41 (2007)
29. Saykin, A.J., Shen, L., et al.: Alzheimer's disease neuroimaging initiative biomarkers as quantitative phenotypes: genetics core aims, progress, and plans. *Alzheimers Dement.* **6**(3), 265–273 (2010)
30. Bertram, L., McQueen, M.B., et al.: Systematic meta-analyses of Alzheimer disease genetic association studies: the AlzGene database. *Nat. Genet.* **39**(1), 17–23 (2007)
31. Li, Y., Willer, C.J., et al.: MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genet. Epidemiol.* **34**(8), 816–834 (2010)
32. Ashburner, J., Friston, K.J.: Voxel-based morphometry—the methods. *Neuroimage* **11**(6 Pt 1), 805–821 (2000)
33. Fischl, B., Salat, D.H., et al.: Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. *Neuron* **33**(3), 341–355 (2002)
34. Pereira, J.B., Cavallin, L., et al.: Influence of age, disease onset and ApoE4 on visual medial temporal lobe atrophy cut-offs. *J. Intern. Med.* **275**(3), 317–330 (2014)
35. Andrawis, J.P., Hwang, K.S., et al.: Effects of ApoE4 and maternal history of dementia on hippocampal atrophy. *Neurobiol. Aging* **33**(5), 856–866 (2012)
36. Risacher, S.L., Kim, S., Shen, L., et al.: The role of apolipoprotein E (APOE) genotype in early mild cognitive impairment (E-MCI). *Front Aging Neurosci.* **5**, 11 (2013)
37. Ma, J., Yu, J.T., Tan, L.: MS4A cluster in alzheimer's disease. *Mol. Neurobiol.* **51**, 1240–1248 (2014)

Improving Imputation Accuracy by Inferring Causal Variants in Genetic Studies

Yue Wu¹, Farhad Hormozdiari^{1,2}, Jong Wha J. Joo^{1,3}, and Eleazar Eskin^{1,4}(✉)

¹ Department of Computer Science, UCLA, Los Angeles, USA
hormozdiari@hsph.harvard.edu, eskin@cs.ucla.edu

² Program in Genetic Epidemiology and Statistical Genetics, Harvard University, Cambridge, USA

³ Department of Molecular and Medical Pharmacology, UCLA, Los Angeles, USA

⁴ Department of Human Genetics, UCLA, Los Angeles, USA

Abstract. Genotype imputation has been widely utilized for two reasons in the analysis of Genome-Wide Association Studies (GWAS). One reason is to increase the power for association studies when causal SNPs are not collected in the GWAS. The second reason is to aid the interpretation of a GWAS result by predicting the association statistics at untyped variants. In this paper, we show that prediction of association statistics at untyped variants that have an influence on the trait produces overly conservative results. Current imputation methods assume that none of the variants in a region (locus consists of multiple variants) affect the trait, which is often inconsistent with the observed data. In this paper, we propose a new method, CAUSAL-Imp, which can impute the association statistics at untyped variants while taking into account variants in the region that may affect the trait. Our method builds on recent methods that impute the marginal statistics for GWAS by utilizing the fact that marginal statistics follow a multivariate normal distribution. We utilize both simulated and real data sets to assess the performance of our method. We show that traditional imputation approaches underestimate the association statistics for variants involved in the trait, and our results demonstrate that our approach provides less biased estimates of these association statistics.

1 Introduction

Genome-wide association studies (GWAS) have been used to discover the genetic variants that affect a trait of interest [1–7]. GWAS collect information on genetic variants, typically single nucleotide polymorphisms (SNPs), from two populations. In this case, the two populations are comprised of a large number of individuals who carry a specific disease (cases) and those who do not (controls). GWAS estimate correlations between disease status and collected genetic variants. After estimating the correlations, we perform a statistical test to indicate if each of the estimated correlations is statistically significant. The computed

Y. Wu and F. Hormozdiari—These authors contributed equally to this work.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 303–317, 2017.

DOI: 10.1007/978-3-319-56970-3_19

significant statistics are known as summary statistics or marginal statistics. In GWAS, due to cost considerations, only a subset of SNPs, called tag SNPs, are genotyped, and SNPs that are not collected are referred to as untyped SNPs. While genotypes of untyped SNPs are not collected, we can infer these variant genotypes using their correlations to the tag SNPs. The correlation between a pair of variants is referred to as linkage disequilibrium (LD) [8,9]. Imputation is a process that uses LD to compute the genotypes of the missing variants [10–17].

Genotype imputation requires two data sets. One data set is a set of individuals who are genotyped at all the SNPs, and this data set is referred to as the reference panel. The other data set, which is the data set of interest, consists of individuals who are only genotyped at the tag SNPs. We can impute the genotypes of untyped SNPs in the second data set by utilizing the correlations between SNPs that derive from the reference panel. In order to use the imputed genotypes for GWAS, we compute the summary statistics of the imputed genotypes by applying the same statistical test as if the imputed SNPs are collected in the second data set. In this paper, we use summary statistics and marginal statistics interchangeably. Summary statistics, such as z-scores, indicate the magnitude of the associations between genotypes and a phenotype of interest.

There are two methodologies for aiding GWAS analysis with imputation. The standard way of utilizing imputation in the GWAS analysis is to impute the genotypes and compute the summary statistics from the imputed genotypes [10–17]. More recently, a second class of methods has been developed that directly imputes the marginal statistics. These methods approximate the combined result of genotype imputation and association test results. It is shown that the statistics of tag SNPs and untyped SNPs follow a multivariate normal distribution (MVN) [18–22]. Thus, given the LD between tag SNPs and untyped SNPs, we get a conditional distribution of statistics of untyped SNPs conditioning on the statistics of tag SNPs. Having the statistics of tag SNPs, we can impute the untyped SNPs with mean of the conditional distribution [23,24]. These methods are shown to have similar accuracy of genotype imputation and are much faster to use for GWAS. In addition, the second class of methods only require summary statistics to perform imputation, while the first class of methods require a level of genotype data that is not always available (e.g. individual genomic data).

Genotype imputation has been widely utilized for two reasons in the analysis of GWAS. One reason is to increase the statistical power of association studies when the causal SNPs are not collected in the GWAS. The second reason is to aid the interpretation of GWAS results by predicting the association statistics at untyped variants. Unfortunately, all the existing methods assume a null-based model where all the variants are not causal. As a result, when there exists a causal variant, the computed summary statistics for untyped SNPs are lower than the true summary statistics. Thus, the null-based imputation approach is conservative and may lose the real causal signals. These approaches are reasonable when the goal is to identify more genetic variants associated with the trait [10–17]. However, when the goal is to interpret the associated regions to identify

the actual causal variants, this assumption will cause bias at variants that are actually causal.

In this paper, we introduce a novel method for imputation of summary statistics under the assumption that some SNPs in a locus can be causal. Our approach uses the statistics at tag SNPs and LD patterns to infer which of the variants are causal and performs imputation with this information taken into account. As shown in previous works [18–21], the joint distribution of marginal statistics follows MVN, and the mean of the distribution depends on which SNPs are causal. We compute the marginal statistics of the untyped SNPs conditional on the marginal statistics of tag SNPs and the knowledge of which SNPs are causal. Since we do not know which variants are causal within a region, we impute the marginal statistics of the untyped SNPs as a weighted average of all possible subsets of SNPs in the region to be causal. Unfortunately, considering all possible subsets of SNPs is intractable, so we assume that we have at most three causal SNPs in a locus. This assumption makes our approach applicable to larger loci in the genome without reducing the accuracy of our method. The idea of bounding the number of causal SNPs is widely used in Fine-mapping literature [20–22].

We show that our method (CAUSAL-Imp) performs favorably in both simulated and real data. We apply our method to simulated data sets where we generated the marginal statistics. Then, we treat some of the SNPs as untyped and others as tagged. We apply CAUSAL-Imp and DIST*, which is our implementation of DIST [23]. We use simulated data to illustrate that CAUSAL-Imp tends to impute summary statistics that are closer to the true generated summary statistics than DIST*. Next, we evaluate our performance utilizing the Northern Finland Birth Cohort (NFBC) data set [25]. We treat the previously reported significant SNPs as untyped and try to impute their summary statistics using CAUSAL-Imp and DIST*. We show that CAUSAL-Imp imputes the associated statistics more accurately than previous approaches.

2 Results

2.1 Overview of CAUSAL-Imp

CAUSAL-Imp builds on methods that perform imputation on summary statistics. It is known that the statistics for a set of SNPs (SNPs in a locus) follow an MVN distribution with a variance-covariance matrix equal to the pairwise correlation between the genotypes [18–21]. For simplicity, let’s consider the case where one SNP is untyped and the rest are tag SNPs in a region; we have ℓ SNPs and the ℓ -th SNP is untyped. Let s_i be the marginal statistics of the i -th SNP. Let $S_{-\ell} = \{s_1, s_2 \cdots s_{\ell-1}\}$ and s_ℓ indicate the marginal statistics for the tag and untyped SNPs, respectively. In traditional methods that impute the summary statistics, the model of the joint distribution is as follows:

$$\left(\begin{bmatrix} S_{-\ell} \\ s_\ell \end{bmatrix} \right) \sim \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{-\ell} & R_{-\ell\ell} \\ R_{-\ell\ell}^T & 1 \end{bmatrix} \right) \quad (1)$$

where $\Sigma_{-\ell}$ be a $((\ell - 1) \times (\ell - 1))$ matrix of LD for all the SNPs, excluding the ℓ -th SNP, and $R_{-\ell\ell}$ be a $((\ell - 1) \times 1)$ vector, which represent the correlation of all the variants with the ℓ -th SNP, excluding the ℓ -th SNP. We can obtain the variance-covariance matrix of the model utilizing the correlation of genotypes from a reference panel, such as the 1000 Genomes data [26,27]. Then, given the association statistics at observed variants, we can use the conditional form of the multivariate normal to estimate the association statistics at the untyped variants. In traditional methods, marginal statistics of untyped SNPs conditioned on the marginal statistics of tag SNP is as follows:

$$\left(s_\ell | S_{-\ell} = \hat{S}_{-\ell} \right) \sim \mathcal{N} \left(R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell}, 1 - R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} R_{-\ell\ell} \right) \tag{2}$$

where $\hat{S}_{-\ell}$ is the observed marginals statistics for all the tag SNPs. We impute the untyped SNP with the mean of above distribution $R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell}$ [23,24].

Our method, CAUSAL-Imp, takes into account the fact that some variants can be causal. Let's assume we only have one causal SNP and the i -th SNP is causal. Then, the marginal statistics for this SNP follows a normal distribution as follows: $s_i \sim N(\lambda_i, 1)$ where λ_i is the non-centrality parameter (NCP) for the i -th SNP that depends on the true effect size of the SNP towards the phenotype. We extend this to the case where the j -th SNP is not causal and is in LD with the causal SNP i . Then the marginal statistics for the j -th SNP is as follows: $s_j \sim N(r_{ij}\lambda_i, 1)$ where r_{ij} is the LD (genotype Pearson's correlation) between SNP i and j .

To provide a simplified description of this section, we assume that all causal variants have the same NCP. However, CAUSAL-Imp takes into account that causal variants can have different NCP values. We define any subset of SNPs that are causal as the causal status. Causal status indicates which SNPs are causal and which are not. We use 1 to indicate the variants which are causal and 0 to indicate the variants that are not causal. Let $C_{-\ell}$ be a vector of size $\ell - 1$ to represent the causal status of the first $\ell - 1$ SNPs. Similarly, Let c_ℓ be a binary variable which indicates the causal status of the ℓ -th SNP. As shown in previous works [18,20,21], the joint marginal statistics given the causal statistics is as follows:

$$\left(\begin{bmatrix} S_{-\ell} \\ s_\ell \end{bmatrix} \middle| \begin{bmatrix} C_{-\ell} \\ c_\ell \end{bmatrix} \right) \sim \mathcal{N} \left(\lambda\sqrt{N} \begin{bmatrix} \Sigma_{-\ell} & R_{-\ell\ell} \\ R_{-\ell\ell}^T & 1 \end{bmatrix} \begin{bmatrix} C_{-\ell} \\ c_\ell \end{bmatrix}, \begin{bmatrix} \Sigma_{-\ell} & R_{-\ell\ell} \\ R_{-\ell\ell}^T & 1 \end{bmatrix} \right)$$

The summary statistics of untyped SNP (s_ℓ) conditioning on the statistics of the tag SNPs ($S_{-\ell}$) and the given causal status, $C = C^*$ is as follows:

$$\left(s_\ell | S_{-\ell} = \hat{S}_{-\ell}, C = C^* \right) \sim \mathcal{N} \left(\underbrace{\lambda\sqrt{N}(1 - R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} R_{-\ell\ell}) c_\ell^*}_{\text{Contribution of causal status for the } \ell\text{-th SNP}} + \underbrace{R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell}}_{\text{Contribution of Null}}, 1 - R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} R_{-\ell\ell} \right) \tag{3}$$

However, the true causal status is not known. Thus, CAUSAL-Imp considers all the possible causal statuses. We impute summary statistics as a weighted average of all the summary statistics computed for the unobserved variants for different causal status.

$$\sum_{C^*} \left(\lambda \sqrt{N} (1 - R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} R_{-\ell\ell}) c_{\ell}^* + R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell} \right) Pr \left(C = C^* | S_{-\ell} = \hat{S}_{-\ell} \right) \quad (4)$$

where $Pr \left(C = C^* | S = \hat{S} \right)$ is the posterior probability of a causal status given the observed marginal statistics. Although we describe the method to consider all possible causal status, in practice, we allow up to three causal variants in a locus to reduce the computational complexity.

2.2 A Motivating Example

Figure 1 illustrates a simple region where we have 10 SNPs. In this example, we observe the statistics of 3 SNPs (SNP3, SNP7, and SNP10), which are indicated by the black arrows. The green triangles indicate the real marginal statistics for all the 10 SNPs. The rest of the SNPs are untyped. Given, the marginal statistics of these three SNPs, we want to impute the marginal statistics of other SNPs. In this example, as the marginal statistic of SNP10 is slightly inflated, we assume one of the SNPs in the region should be causal. In CAUSAL-Imp, we do not know the real causal SNPs, thus we consider all the possible causal statuses in this region.

In this example, there are 2^{10} possible causal statuses. For a specific causal status, we impute the summary statistics of the seven unobserved SNPs utilizing the conditional MVN. The red dots indicate the marginal statistics imputed by CAUSAL-Imp. The blue dots indicate the marginal statistics imputed by DIST* (our implementation of DIST [23]), which assumes the null-model where all variants are not causal. In this example, our imputed marginal statistics are closer to the true marginal statistics than those of DIST*. Note that we perform our evaluations using our own implementation of the standard summary statistic method (DIST) [23], which we refer to as DIST*. The reason we used our own implementation is that these methods rely on many matrix operations that may result in numerical issues. The differences in linear algebra libraries dealing with numerical issues can cause differences in the results. By reimplementing DIST, our approach and DIST* share many parts of the implementation to eliminate this issue from the evaluation.

2.3 CAUSAL-Imp Achieves Better Statistics Compared to the Existing Methods in Simulated Data Sets

In order to assess the performance of our method, we simulated marginal statistics utilizing the Northern Finland Birth Cohort (NFBC) data set. The NFBC data set consists of 10 phenotypes and 331,476 genotypes measured in 5,327 individuals. Since imputation is a regional analysis, we selected 20 regions from the NFBC and computed the LD between each pair of SNPs. In this setting, we used 100 SNPs for each locus. Then, we simulated the marginal statistics from the MVN distribution similar to the previous studies [20, 21, 28, 29], where we implant one causal SNP. We generated 1000 sets of summary statistics.

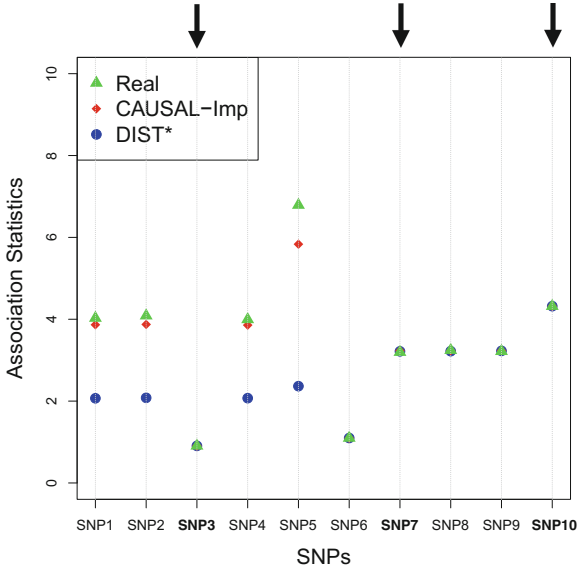


Fig. 1. Motivating example for CAUSAL-Imp. The black arrows indicate the observed (tag) SNPs. Utilizing the fact that the observed marginal statistics of SNP10 is inflated, we can assume one of the SNPs in this region is causal.

We assume 30% of the SNPs are tagged and that the rest of SNPs, including the causal SNP, are untyped. Then, we ran CAUSAL-Imp and DIST* on the simulated data.

We compute the average distance between the imputed marginal statistics and the true simulated marginal statistics as a measure of accuracy. We use the ℓ_1 distance as a measure of accuracy, which is computed as follows: $d(x, y) = \frac{1}{N} \sum_{i=1}^N |x_i - y_i|$. We compute this distance for the causal SNP, shown in Fig. 2A, and the other SNPs, shown in Fig. 2B. We vary the power from 20% to 80%. We observe that the statistics imputed by our method are closer to the true statistics. We perform similar experiment where we implant two causal variants in a locus. In this experiment, the imputed statistics from CAUSAL-Imp are closer to true statistics compared to DIST*. The results for this experiment are not shown due to space limitation.

2.4 CAUSAL-Imp Controls Type I Error

We illustrate that CAUSAL-Imp performs better than existing methods. In addition, we need to show that these methods control the Type I error. Imputed summary statistics that are controlled for Type I error under the null (no variant is causal) are not inflated or deflated. Genomic inflation is a metric used to check whether or not the Type I error is controlled [30]. We expect the genomic inflation to be close to one when there exists no inflation or deflation of

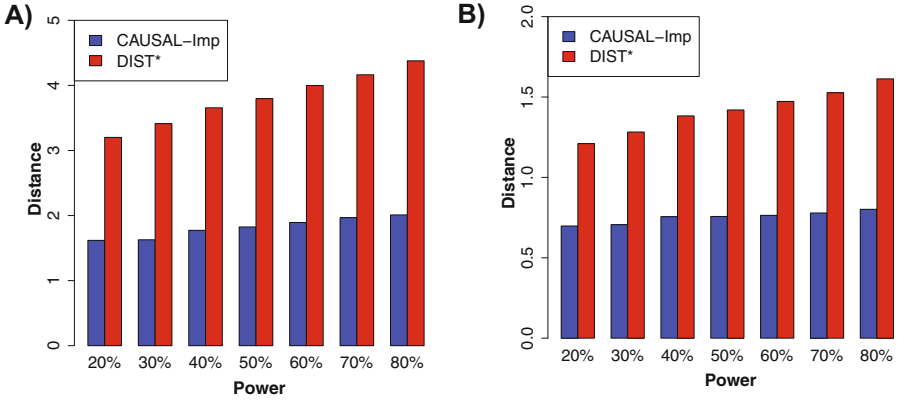


Fig. 2. CAUSAL-Imp achieves better statistics compared to the existing methods in simulated data sets. We simulated marginal statistics for regions that are obtained from the NFBC data. We compared the imputed marginal statistics of our method and DIST*. Our method tends to impute statistics that are closer to the true estimated marginal statistics both for causal and non-causal SNPs. We use ℓ_1 norm to compute the distance. We range the power on the causal SNPs from 20% to 80%. Panel (A) illustrates the results of the causal variants. Panel (B) illustrates the results of non-causal variants.

statistics. In the experiment, we take 10 regions from the NFBC data set to compute the LD. For each region, we simulated 100 sets of statistics sampling from normal distribution under the null where no variant is causal. We consider 30% of the variants to be missing, and then we impute their summary statistics. The genomic inflation factor is computed using the mean of the squared statistics divided by 0.452. The genomic inflation for the true summary statistics is 0.98, and the genomic inflation for CAUSAL-Imp is 0.93. However, the genomic inflation of DIST* and IMPUTE2 [12] are 0.80 and 1.02, respectively. Thus, our results indicate that CAUSAL-Imp controls the Type I error.

2.5 CAUSAL-Imp Achieves Better Statistics Compared to the Existing Methods in Northern Finland Birth Cohort (NFBC)

The actual utility of our approach is in examining regions that contain associations where the actual causal variants are not collected. We simulate this scenario by taking actual associated regions in the NFBC dataset and removing the peak associated SNPs from each associated regions (which were reported in previous study [25]). We then apply CAUSAL-Imp, DIST*, and IMPUTE2 [12] to evaluate the accuracies of these methods on the peak SNPs. The results are shown in Table 1. To compare the performance between CAUSAL-Imp, DIST*, and IMPUTE2, we showed the predicted errors (PE) of the methods. The predicted error is the absolute distance to the true statistics. We observe that the imputed summary statistics from CAUSAL-Imp are closer to the estimated summary statistics than those of DIST*.

Table 1. CAUSAL-Imp achieves better statistics in NFBC data set. We run association on the NFBC dataset. We consider the SNPs that are reported significant in a previous study [25]. Then, we treat these SNPs as untyped and impute the marginal statistics using CAUSAL-Imp, DIST*, and IMPUTE2. We compute the marginal statistics and predicted errors (PE) of the methods. Our method tends to produce summary statistics closer to the estimated marginal statistics than the two other methods.

Phenotype	chr	rsID	True statistics	DIST*		CAUSAL-Imp		IMPUTE2	
				Statistics	PE	Statistics	PE	Statistics	PE
TG	2	rs673548	-5.444	-5.37	0.074	-5.38	0.064	-4.46	0.984
	8	rs10096633	-5.679	-5.63	0.049	-5.64	0.039	-5.17	0.509
	15	rs2624265	4.22	3.55	0.67	4.15	0.07	3.60	0.62
HDL	15	rs1532085	7.13	5.59	1.54	7.17	0.04	6.47	0.66
	16	rs3764261	12.01	8.23	3.78	8.28	3.73	6.47	5.54
	16	rs255049	6.06	5.11	0.95	5.61	0.45	5.70	0.36
	17	rs9891572	4.25	3.99	0.26	4.02	0.23	4.40	0.15
LDL	1	rs646776	-7.70	-7.92	0.22	-7.92	0.22	-6.96	0.74
	2	rs693	6.81	6.27	0.54	6.63	0.18	5.91	0.9
	11	rs102275	-4.51	-4.43	0.08	-4.44	0.07	-4.54	0.03
	11	rs174546	-4.52	-4.43	0.09	-4.45	0.07	-4.58	0.06
	11	rs174556	-4.69	-4.73	0.04	-4.75	0.06	-4.62	0.07
	11	rs1535	-4.43	-4.46	0.03	-4.46	0.03	-4.45	0.02
	19	rs11668477	-5.96	-3.78	2.18	-3.78	2.18	-5.33	0.63
	19	rs157580	-5.161	-2.6	2.561	-5.24	0.079	-4.20	0.961
CRP	12	rs2650000	-7.08	-5.25	1.83	-7.36	0.28	-6.05	1.03
GLU	2	rs560887	-6.97	-6.21	0.78	-6.80	0.17	-5.69	1.28
	7	rs10244051	5.31	4.34	0.97	4.67	0.64	4.97	0.34
	7	rs2191348	5.30	4.33	0.97	4.66	0.64	4.97	0.33
	11	rs1447352	-6.35	-5.08	1.27	-5.39	0.96	-4.75	1.6
	11	rs7121092	-5.50	-4.93	0.57	-5.78	0.28	-4.60	0.9

3 Methods

3.1 A Standard Association Statistics

Here we have a quantitative phenotype collected for n individuals at m SNPs. Let Y be a $(n \times 1)$ vector of phenotypic values where y_j is the phenotypic values for j -th individual. Let G be a $(n \times m)$ matrix of minor allele counts where $g_{ji} \in \{0, 1, 2\}$ is the minor allele count for j -th individual at i -th SNP, and let X be the normalized allele counts matrix G . Define β to be the a $(m \times 1)$ effect size vector, and β_i is the effect size of i -th SNP. For simplicity, we assume both the phenotypic values and the allele counts at each SNP are normalized to have mean zero and variance one. Let $x_{ji} \in \left\{ \frac{-2p_i}{\sqrt{p_i(1-p_i)}}, \frac{1-2p_i}{\sqrt{p_i(1-p_i)}}, \frac{2-2p_i}{\sqrt{p_i(1-p_i)}} \right\}$, which is the normalized value for g_{ji} where p_i is the frequency of i -th SNP in the population. Assuming Fisher's polygenic model holds, we use the generative model, $Y = \mathbf{1}^T \mu + \sum_{i=1}^m X_i \beta_i + \mathbf{e}$ where μ is the phenotypic mean of population, $\mathbf{1}$ is a $(n \times 1)$ vector of one, X_i is normalized minor allele counts at i -th SNP, β_i is effect

size of i -th SNP, and \mathbf{e} is a vector of measurement noise and environment contributions. We assume \mathbf{e} has a normal distribution with mean zero and variance, $\sigma^2 \mathbf{I}$ ($\mathbf{e} \sim N(0, \sigma^2 \mathbf{I})$).

In standard GWAS, effect size for each SNP is estimated one SNP at a time. Thus, to compute the marginal statistics for each SNP, we use the following model, $Y = \mathbf{1}^T \mu + X_i \beta_i + \mathbf{e}$. We note there is a discrepancy between the generative model and testing model; as long as there is no population structure in the data, the estimated effect size is unbiased and follows a normal distribution with mean equal to the true value of effect size. Thus, we have: $\hat{\beta}_i = \frac{X_i^T Y}{X_i^T X_i}$ and $\hat{\beta}_i \sim N(\beta_i, \sigma(X_i^T X_i)^{-1})$. We use “hat” for each variable to indicate the estimated value for that variable.

It is known that the marginal statistics for each SNP is computed as the ratio between the estimated effect size and the estimated variance. Let s_i indicate the marginal statistics estimated for i -th SNP. As the marginal statistics follow a normal distribution, we can define the statistics as follows:

$$s_i = \frac{\hat{\beta}_i}{\hat{\sigma}} \sqrt{n} \sim N\left(\frac{\beta_i}{\sigma} \sqrt{n}, 1\right) = N(\lambda_i, 1)$$

where λ_i is the non-centrality parameter (NCP) for the i -th SNP and $\lambda_i = \frac{\beta_i}{\sigma} \sqrt{n}$.

3.2 Indirect Association Statistics

To show the indirect association statistics, we assume i -th variant is associated with the phenotype and j -th variant is correlated with the i -th variant. Thus, the estimated effect size and the marginal statistics for the j -th variant is computed as $\hat{\beta}_j = \frac{X_j^T Y}{X_j^T X_j}$, $\hat{\beta}_j \sim N(\beta_j, \sigma(X_j^T X_j)^{-1})$, $s_j \sim N(r_{ij} \lambda_i, 1)$, where r_{ij} is the correlation between genotypes of i -th and j -th SNP. Moreover, we estimate the correlation between the genotypes as $\frac{1}{n} X_i^T X_j$. We compute the covariance between the estimated marginal statistics for the i -th and j -th SNP as $\text{Cov}(s_i, s_j) = r_{ij}$. Thus, the joint distribution of the marginal association statistics for the two SNPs given their NCPs follows a multivariate normal distribution (MVN):

$$\left(\begin{array}{c|c} [s_i] & [\lambda_i] \\ \hline [s_j] & [\lambda_j] \end{array} \right) \sim \mathcal{N} \left(\begin{array}{c} [\lambda_i] \\ [\lambda_j] \end{array}, \begin{array}{cc} 1 & r_{ij} \\ r_{ij} & 1 \end{array} \right).$$

3.3 Traditional Summary Statistics Imputation When One SNP is Untyped

In this section, we show how traditional summary statistics imputation approaches [23, 24] work under the scenario when only one SNP is untyped in a locus. Let’s say we have ℓ SNPs in a region where $\ell - 1$ of the SNPs are tagged and only the last SNPs is untyped. For simplicity, We select the ℓ -th SNP to be untyped. Let s_i indicate the marginal statistics of i -th SNP. Let $S_{-\ell} = \{s_1, s_2, \dots, s_{\ell-1}\}$ be a $(\ell - 1 \times 1)$ vector of association statistics,

$\Lambda_{-\ell} = \{\lambda_1, \lambda_2, \dots, \lambda_{\ell-1}\}$ be a $(\ell - 1 \times 1)$ vector of NCPs, and $\Sigma_{-\ell}$ be a $(\ell - 1 \times \ell - 1)$ matrix of the pairwise correlation coefficients for the tag SNPs. For the untyped SNP, we use λ_ℓ to indicate the unknown NCP. We want to impute the association statistic s_ℓ , and let $R_{-\ell\ell}$ denote the $(\ell - 1 \times 1)$ vector of the correlation coefficients between s_ℓ and the $\ell - 1$ tag SNPs. Thus the joint distribution of the association statistics of the untyped SNP, s_ℓ , and the $\ell - 1$ tag SNPs, $S_{-\ell}$, follows a multivariate normal distribution, which can be expressed as follows:

$$\begin{bmatrix} S_{-\ell} \\ s_\ell \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \Lambda_{-\ell} \\ \lambda_\ell \end{bmatrix}, \begin{bmatrix} \Sigma_{-\ell} & R_{-\ell\ell} \\ R_{-\ell\ell}^T & 1 \end{bmatrix} \right) \tag{5}$$

Under the null assumption where s_ℓ and $S_{-\ell}$ are not associated, λ_ℓ and $\Lambda_{-\ell}$ are zeros. Using this equation, we can generate a distribution of the statistics of untyped SNP; s_ℓ condition on the observed summary statistics, $S_{-\ell} = \hat{S}_{-\ell}$. The conditional distribution follows a multivariate normal distribution, which is computed as follows:

$$(s_\ell | S_{-\ell} = \hat{S}_{-\ell}) \sim \mathcal{N} \left(R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell}, 1 - R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} R_{-\ell\ell} \right). \tag{6}$$

Thus, utilizing this equation the traditional summary statistics imputation approaches impute the statistics of the untyped SNP as $R_{-\ell\ell}^T \Sigma_{-\ell}^{-1} \hat{S}_{-\ell}$.

3.4 Traditional Summary Statistics Imputation When More Than One SNP Is Untyped

In this section, we show how traditional summary statistics imputation approaches [23, 24] work under the scenario where more than one SNP is untyped in a locus. We use \mathcal{U} and \mathcal{T} to indicate the set of untyped and tag SNPs, respectively. Let $S_{\mathcal{U}}$ indicate the unobserved summary statistics of untyped SNPs, and let $S_{\mathcal{T}}$ indicate observed summary statistics of tag SNPs, respectively. We use $\Sigma_{\mathcal{U}}$ and $\Sigma_{\mathcal{T}}$ to denote $(p \times p)$ and $(\ell \times \ell)$ matrices of pairwise correlation coefficients obtained from the untyped SNPs and tag SNPs, respectively. We want to impute unobserved summary statistics $S_{\mathcal{U}}$ using both observed ℓ SNPs and p unobserved SNPs. In this case, $\Lambda_{\mathcal{U}}$ is a $(p \times 1)$ vector of NCPs of untyped SNPs and $\Sigma_{\mathcal{U}, \mathcal{T}}$ denotes the $(p \times \ell)$ matrix of the correlation coefficients between the p untyped SNPs and the ℓ tag SNPs. The joint distribution of the association statistics of the untyped SNP $S_{\mathcal{U}}$ and the tag SNPs $S_{\mathcal{T}}$ follows a multivariate normal distribution, which can be expressed as follows:

$$\begin{bmatrix} S_{\mathcal{U}} \\ S_{\mathcal{T}} \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \Lambda_{\mathcal{U}} \\ \Lambda_{\mathcal{T}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathcal{U}} & \Sigma_{\mathcal{U}, \mathcal{T}}^T \\ \Sigma_{\mathcal{U}, \mathcal{T}} & \Sigma_{\mathcal{T}} \end{bmatrix} \right) \tag{7}$$

Under the null assumption that the untyped SNPs and tag SNPs are not associated, the NCPs of both $\Lambda_{\mathcal{U}}$ and $\Lambda_{\mathcal{T}}$ are zeros. Using the Eq. (7), we can generate a distribution of the statistics of the untyped SNPs, $S_{\mathcal{U}}$, conditioned on the observed statistics, $S_{\mathcal{T}} = \hat{S}_{\mathcal{T}}$. The conditional distribution follows a multivariate normal distribution, which is computed as follows:

$$\left(S_U | S_T = \hat{S}_T\right) \sim \mathcal{N}\left(\Sigma_{U,T}^T \Sigma_T^{-1} \hat{S}_T, \Sigma_U - \Sigma_{U,T}^T \Sigma_T^{-1} \Sigma_{U,T}\right) \quad (8)$$

Thus, utilizing the above equation, the traditional summary statistics imputation approaches impute the statistic of the untyped SNPs as $\Sigma_{U,T}^T \Sigma_T^{-1} \hat{S}_T$.

3.5 CAUSAL-Imp Summary Statistics Imputation with Fixed NCP

Recall that having ℓ SNPs, whose summary statistics are observed, and p SNPs whose summary statistics are unobserved, we have a multivariate normal distribution expressed as Eq. (7). Instead of assuming all Λ_U and Λ_T are zeros, our method considers any subset of SNPs to be causal. We introduce C to denote the causal status of the SNPs. Causal status is a $((\ell + p) \times 1)$ vector of zeros and ones where c_i indicates the causal status of the i -th SNP. Each SNP can have two possible causal statuses 0 or 1, where 0 indicates the SNP is not causal and 1 indicates the SNP is causal. For simplicity, we assume the NCPs for all the causal variants are the same and equal to $\lambda\sqrt{N}$. Later, we will relax this assumption. There are $2^{\ell+p}$ possible causal statuses for C , which is denoted by the set \mathcal{C} (in practice we only consider up to three causal variants in locus, thus CAUSAL-Imp needs to consider at most $(\ell + p)^3$ causal statuses). The causal status is consists of two parts, the causal status of tag SNPs, which we denote by C_T , and the causal status of untyped SNPs, which we denote by C_U . The joint distribution of observed and unobserved summary statistics in Eq. (8) can be expressed as follows: $\left(\begin{bmatrix} S_U \\ S_T \end{bmatrix} \middle| \begin{bmatrix} C_U \\ C_T \end{bmatrix}\right) \sim \mathcal{N}\left(\lambda\sqrt{n} \begin{bmatrix} \Sigma_U & \Sigma_{U,T}^T \\ \Sigma_{U,T} & \Sigma_T \end{bmatrix} C, \begin{bmatrix} \Sigma_U & \Sigma_{U,T}^T \\ \Sigma_{U,T} & \Sigma_T \end{bmatrix}\right)$. Using this Equation, we can compute the distribution of the untyped statistics, S_U , conditional on the observed statistics, $S_T = \hat{S}_T$ and the known causal status, $C = \mathbf{c}^*$. This conditional distribution follows a multivariate normal that is expressed as follows:

$$\left(S_U | S_T = \hat{S}_T, C = \mathbf{c}^*, \lambda\right) \sim \mathcal{N}\left(\lambda\sqrt{n}(\Sigma_U - \Sigma_{U,T}\Sigma_T^{-1}\Sigma_{U,T})C_U + \Sigma_{U,T}^T\Sigma_T^{-1}\hat{S}_T, \Sigma_U - \Sigma_{U,T}^T\Sigma_T^{-1}\Sigma_{U,T}\right) \quad (9)$$

We want to compute the probability of summary statistics of untyped SNPs given the summary statistics of the tag SNPs, $\Pr\left(S_U | S_T = \hat{S}_T\right)$. Utilizing the total probability and the Baye's rule, we have:

$$\begin{aligned} \Pr\left(S_U | S_T = \hat{S}_T\right) &= \sum_{C^* \in \mathcal{C}, \lambda} \Pr\left(S_U, C = C^* | S_T = \hat{S}_T\right) \\ &= \sum_{C^* \in \mathcal{C}} \Pr\left(S_U | S_T = \hat{S}_T, C = C^*\right) \Pr\left(C = C^* | S_T = \hat{S}_T\right) \end{aligned} \quad (10)$$

where $\Pr\left(S_U | S_T = \hat{S}_T, C = C^*\right)$ is computed from Eq. (9), and $\Pr\left(C = C^* | S_T = \hat{S}_T\right)$ is computed as follows:

$$Pr\left(C = C^* | S_{\mathcal{T}} = \hat{S}_{\mathcal{T}}\right) = \frac{Pr\left(S_{\mathcal{T}} = \hat{S}_{\mathcal{T}} | C = C^*\right) Pr\left(C = C^*\right)}{\sum_{C^\dagger \in \mathcal{C}} Pr\left(S_{\mathcal{T}} = \hat{S}_{\mathcal{T}} | C = C^\dagger\right) Pr\left(C = C^\dagger\right)} \quad (11)$$

where $Pr(C = C^\dagger)$ is the prior of the causal status. Similar to most of the fine-mapping methods, for the prior, we assume that SNPs are independent and the probability of a SNP to be causal is equal to 0.01 [20, 21, 31]. This prior implies a sparsity prior on the causal status. Moreover, $Pr(S_{\mathcal{T}} = \hat{S}_{\mathcal{T}} | C = C^*)$ is the likelihood of observed summary statistics given the causal status C^* . The observed summary statistics, given the causal status, follows a normal distribution and is computed as follows:

$$(S_{\mathcal{T}} = \hat{S}_{\mathcal{T}} | C = C^*, \lambda) \sim \mathcal{N}\left(\lambda\sqrt{n}(\Sigma_{\mathcal{U},\mathcal{T}}C_{\mathcal{U}}^* + \Sigma_{\mathcal{T}}C_{\mathcal{T}}^*), \Sigma_{\mathcal{T}}\right) \quad (12)$$

Utilizing Eqs. (9), (11), and (12), we compute the value of $Pr(S_{\mathcal{U}} | S_{\mathcal{T}}, \lambda)$ from Eq. (10). Thus, we impute $S_{\mathcal{U}}$ as the mean of $(S_{\mathcal{U}} | S_{\mathcal{T}}, \lambda)$ as follows:

$$\sum_{C^* \in \mathcal{C}} \left(\lambda\sqrt{n}(\Sigma_{\mathcal{U}} - \Sigma_{\mathcal{U},\mathcal{T}}\Sigma_{\mathcal{T}}^{-1}\Sigma_{\mathcal{U},\mathcal{T}})C_{\mathcal{U}} + \Sigma_{\mathcal{U},\mathcal{T}}^T\Sigma_{\mathcal{T}}^{-1}\hat{S}_{\mathcal{T}}\right) P(C = C^* | S_{\mathcal{T}} = \hat{S}_{\mathcal{T}}) \quad (13)$$

3.6 CAUSAL-Imp Summary Statistics Imputation

In previous sections, we assume the NCPs of the causal variants are fixed and their values are known. In this section, we relax this assumption. We utilize a CAVIAR-model [20–22] that is used in fine-mapping frameworks. In CAVIAR-model, the joint distribution of marginal statistics (S), given the vector of NCPs (Λ), follows a MVN distribution that is expressed as $(S | \Lambda) \sim \mathcal{N}(\Lambda, \Sigma)$. In addition, the vector of NCPs given the causal status (C), follows a MVN distribution that is expressed as $(\Lambda | C) \sim \mathcal{N}(0, \Sigma\Sigma_C\Sigma)$. Here $\Sigma_C = \sigma^2\text{diag}(C)$ and $\text{diag}(X)$ creates a diagonal matrix where the i -th diagonal element is assigned to x_i . Using the conjugate prior, we have the following:

$$(S | C) \sim \mathcal{N}(0, \Sigma + \Sigma\Sigma_C\Sigma), \quad (14)$$

Thus, utilizing the same statistical framework in CAUSAL-Imp, we have the following:

$$\left(\begin{bmatrix} S_{\mathcal{U}} \\ S_{\mathcal{T}} \end{bmatrix} \middle| \begin{bmatrix} C_{\mathcal{U}} \\ C_{\mathcal{T}} \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}\right) \quad (15)$$

where:

$$\begin{aligned} V_{11} &= \Sigma_{\mathcal{U}} + \Sigma_{\mathcal{U}}\sigma^2\text{diag}(C_{\mathcal{U}})\Sigma_{\mathcal{U}} + \Sigma_{\mathcal{U},\mathcal{T}}^T\sigma^2\text{diag}(C_{\mathcal{T}})\Sigma_{\mathcal{U},\mathcal{T}} \\ V_{12} &= \Sigma_{\mathcal{U},\mathcal{T}}^T + \Sigma_{\mathcal{U}}\text{diag}(C_{\mathcal{U}})\Sigma_{\mathcal{U},\mathcal{T}}^T + \Sigma_{\mathcal{U},\mathcal{T}}^T\text{diag}(C_{\mathcal{T}})\Sigma_{\mathcal{T}} \\ V_{21} &= \Sigma_{\mathcal{U},\mathcal{T}} + \Sigma_{\mathcal{T}}\text{diag}(C_{\mathcal{T}})\Sigma_{\mathcal{U},\mathcal{T}} + \Sigma_{\mathcal{U},\mathcal{T}}\text{diag}(C_{\mathcal{U}})\Sigma_{\mathcal{U}} \\ V_{22} &= \Sigma_{\mathcal{T}} + \Sigma_{\mathcal{T}}\sigma^2\text{diag}(C_{\mathcal{T}})\Sigma_{\mathcal{T}} + \Sigma_{\mathcal{U},\mathcal{T}}\sigma^2\text{diag}(C_{\mathcal{U}})\Sigma_{\mathcal{U},\mathcal{T}}^T. \end{aligned}$$

Using the MVN conditional distribution, we have:

$$\left(S_U | S_T = \hat{S}_T, C = C^*\right) \sim \mathcal{N}\left(V_{12}V_{22}^{-1}\hat{S}_T, V_{11} - V_{12}V_{22}^{-1}V_{21}\right) \quad (16)$$

Thus, for a given causal status, the optimal value for the imputed marginal statistics is the mean of above distribution, which is $V_{12}V_{22}^{-1}\hat{S}_T$. It is worth mentioning that both V_{12} and V_{22} depend on the vector of causal status $C = C^*$. CAUSAL-Imp utilizes Eq. (16) instead of Eq. (9).

4 Discussion

Genotype imputation is widely used to predict the genotypes of untyped SNPs that are not collected in a data set. This approach utilizes the correlation (LD) between the untyped SNPs and the tag SNPs whose genotypes are collected. We propose a new method, CAUSAL-Imp, which combines the principle of fine mapping and summary statistics imputation. CAUSAL-Imp computes the summary statistics for unobserved SNPs by conditioning on the statistics of the observed SNPs and given causal status. CAUSAL-Imp considers all the possible causal statuses where any subset of SNPs can be causal. Thus, the imputed summary statistic is the weighted average of all the summary statistics computed for the unobserved variants for different causal statuses.

Our approach builds upon the recently developed summary statistics framework for imputation [23, 24, 32]. Imputation methods utilizing HMMs to impute individual level data were developed almost 10 years ago [10–17] and have been improved ever since. In our approach, we incorporate idea of a causal variant and implicitly take the phenotype into account when performing the imputation. It is theoretically possible to extend the HMM-based imputation approaches to take into account causal variants and phenotypes. However, the implementation of such an approach would be more complicated and required more effort.

References

1. Zeggini, E., Weedon, M.N., Lindgren, C.M., et al.: Replication of genome-wide association signals in UK samples reveals risk loci for type 2 diabetes. *Science* **316**(5829), 1336–1341 (2007)
2. Sladek, R., Rocheleau, G., Rung, J., et al.: A genome-wide association study identifies novel risk loci for type 2 diabetes. *Nature* **445**(7130), 881–885 (2007)
3. Hakonarson, H., Grant, S.F.A., Bradfield, J.P., et al.: A genome-wide association study identifies *kiaa0350* as a type 1 diabetes gene. *Nature* **448**(7153), 591–594 (2007)
4. Yang, J., Manolio, T.A., Pasquale, L.R., et al.: Genome partitioning of genetic variation for complex traits using common SNPs. *Nat. Genet.* **43**(6), 519–525 (2011)
5. Kottgen, A., Albrecht, E., Teumer, A., et al.: Genome-wide association analyses identify 18 new loci associated with serum urate concentrations. *Nat. Genet.* **45**(2), 145–154 (2013)

6. Yi, L., Vitart, V., Burdon, K.P., et al.: Genome-wide association analyses identify multiple loci associated with central corneal thickness and keratoconus. *Nat. Genet.* **45**(2), 155–163 (2013)
7. Ripke, S., O’Dushlaine, C., Chambert, K., et al.: Genome-wide association analysis identifies 13 new risk loci for schizophrenia. *Nat. Genet.* **45**(10), 1150–1159 (2013)
8. Reich, D.E., Cargill, M., Bolk, S., et al.: Linkage disequilibrium in the human genome. *Nature* **411**(6834), 199–204 (2001)
9. Pritchard, J.K., Przeworski, M.: Linkage disequilibrium in humans: models and data. *Am. J. Hum. Genet.* **69**(1), 1–14 (2001)
10. Browning, S.R.: Missing data imputation and haplotype phase inference for genome-wide association studies. *Hum. Genet.* **124**(5), 439–450 (2008)
11. Howie, B., Fuchsberger, C., Stephens, M., Marchini, J., Abecasis, G.R.: Fast and accurate genotype imputation in genome-wide association studies through pre-phasing. *Nat. Genet.* **44**(8), 955–959 (2012)
12. Howie, B.N., Donnelly, P., Marchini, J.: A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS Genet.* **5**(6), e1000529 (2009)
13. Li, Y., Willer, C., Sanna, S., Abecasis, G.: Genotype imputation. *Annu. Rev. Genomics Hum. Genet.* **10**, 387–406 (2009)
14. Li, Y., Willer, C.J., Ding, J., Scheet, P., Abecasis, G.R.: Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genet. Epidemiol.* **34**(8), 816–834 (2010)
15. Marchini, J., Howie, B.: Genotype imputation for genome-wide association studies. *Nat. Rev. Genet.* **11**(7), 499–511 (2010)
16. Marchini, J., Howie, B.: Comparing algorithms for genotype imputation. *Am. J. Hum. Genet.* **83**(4), 535–539 (2008). (author reply 539–540)
17. 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**(7), 906–913 (2007)
18. Han, B., Kang, H.M., Eskin, E.: Rapid and accurate multiple testing correction and power estimation for millions of correlated markers. *PLoS Genet.* **5**(4), e1000456 (2009)
19. Kostem, E., Lozano, J.A., Eskin, E.: Increasing power of genome-wide association studies by collecting additional single-nucleotide polymorphisms. *Genetics* **188**(2), 449–460 (2011)
20. Hormozdiari, F., Kostem, E., Kang, E.Y., Pasaniuc, B., Eskin, E.: Identifying causal variants at loci with multiple signals of association. *Genetics* **198**(2), 497–508 (2014)
21. Hormozdiari, F., Kichaev, G., Yang, W.-Y., Pasaniuc, B., Eskin, E.: Identification of causal genes for complex traits. *Bioinformatics* **31**(12), i206–i213 (2015)
22. Hormozdiari, F., van de Bunt, M., Segrè, A.V., et al.: Colocalization of GWAS and eQTL signals detects target genes. *Am. J. Hum. Genet.* **99**(6), 1245–1260 (2016)
23. Lee, D., Bigdeli, T.B., Riley, B.P., Fanous, A.H., Bacanu, S.A.: DIST: direct imputation of summary statistics for unmeasured SNPs. *Bioinformatics* **29**(22), 2925–2927 (2013)
24. Pasaniuc, B., Zaitlen, N., Shi, H., et al.: Fast and accurate imputation of summary statistics enhances evidence of functional enrichment. *Bioinformatics* **30**(20), 2906–2914 (2014)
25. Sabatti, C., Service, S.K., Hartikainen, A.-L., et al.: Genome-wide association analysis of metabolic traits in a birth cohort from a founder population. *Nat. Genet.* **41**(1), 35–46 (2009)

26. Durbin, R.M., Altshuler, D.L., Durbin, R.M., et al.: A map of human genome variation from population-scale sequencing. *Nature* **467**(7319), 1061–1073 (2010)
27. McVean, G.A., Altshuler, D.M., Durbin, R.M., et al.: An integrated map of genetic variation from 1,092 human genomes. *Nature* **491**(7422), 56–65 (2012)
28. Zaitlen, N., Kang, H.M., Eskin, E., Halperin, E.: Leveraging the hapmap correlation structure in association studies. *Am. J. Hum. Genet.* **80**(4), 683–691 (2007)
29. Joo, J.W.J., Hormozdiari, F., Han, B., Eskin, E.: Multiple testing correction in linear mixed models. *Genome Biol.* **17**(1), 62 (2016)
30. Devlin, B., Roeder, K.: Genomic control for association studies. *Biometrics* **55**(4), 997–1004 (1999)
31. Duong, D., Zou, J., Hormozdiari, F., et al.: Using genomic annotations increases statistical power to detect eGenes. *Bioinformatics* **32**(12), i156–i163 (2016)
32. Hormozdiari, F., Kang, E.Y., Bilow, M., et al.: Imputing phenotypes for genome-wide association studies. *Am. J. Hum. Genet.* **99**(1), 89–103 (2016)

The Copy-Number Tree Mixture Deconvolution Problem and Applications to Multi-sample Bulk Sequencing Tumor Data

Simone Zaccaria^{1,2}, Mohammed El-Kebir^{2,3}, Gunnar W. Klau^{2,4,5},
and Benjamin J. Raphael^{2,3}(✉)

¹ Dipartimento di Informatica, Univ. degli Studi di Milano-Bicocca, Milan, Italy

² Department of Computer Science, Brown University, Providence, RI 02912, USA

³ Department of Computer Science, Princeton University, Princeton, NJ 08540, USA

braphael@princeton.edu

⁴ Life Sciences Group, Centrum Wiskunde & Informatica (CWI),
Amsterdam, The Netherlands

⁵ Algorithmic Bioinformatics, Heinrich Heine University, Düsseldorf, Germany

Abstract. Cancer is an evolutionary process driven by somatic mutation. This process can be represented as a phylogenetic tree. Constructing such a phylogenetic tree from genome sequencing data is a challenging task due to the mutational complexity of cancer and the fact that nearly all cancer sequencing is of bulk tissue, measuring a superposition of somatic mutations present in different cells. We study the problem of reconstructing tumor phylogenies from copy number aberrations (CNAs) measured in bulk-sequencing data. We introduce the Copy-Number Tree Mixture Deconvolution (CNTMD) problem, which aims to find the phylogenetic tree with the fewest number of CNAs that explain the copy number data from multiple samples of a tumor. CNTMD generalizes two approaches that have been researched intensively in recent years: deconvolution/factorization algorithms that aim to infer the number and proportions of clones in a mixed tumor sample; and phylogenetic models of copy number evolution that model the dependencies between copy number events that affect the same genomic loci. We design an algorithm for solving the CNTMD problem and apply the algorithm to both simulated and real data. On simulated data, we find that our algorithm outperforms existing approaches that perform either deconvolution or phylogenetic tree construction under the assumption of a single tumor clone per sample. On real data, we analyze multiple samples from a prostate cancer patient, identifying clones within these samples and a phylogenetic tree that relates these clones and their differing proportions across samples. This phylogenetic tree provides a higher-resolution view of copy number evolution of this cancer than published analyses.

1 Introduction

Cancer results from an evolutionary process where somatic mutations accumulate in a population of cells during the lifetime of an individual [21]. Thus, a

S. Zaccaria and M. El-Kebir—Joint first authorship.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 318–335, 2017.

DOI: 10.1007/978-3-319-56970-3_20

tumor consists of heterogeneous subpopulations of cells, or *clones*. Each clone comprises cells that share a unique complement of somatic mutations. Quantifying this intra-tumor heterogeneity has been shown to be important in cancer treatment [29]. While intra-tumor heterogeneity complicates the identification of mutations in bulk-sequencing data from a tumor sample containing millions of cells, it also provides a signal for inferring the tumor composition—the number and proportion of clones within a sample—as well as the ancestral history of somatic mutations during cancer development [12]. Thus, a number of methods have been developed to infer phylogenetic trees from DNA sequencing data from one or more samples of a tumor [5, 8, 12–14, 17, 19, 20, 28].

One class of mutations that are particularly useful for inferring tumor composition and tumor evolution are copy-number aberrations (CNAs), which include duplications and deletions of large genomic regions. CNAs are ubiquitous in solid tumors and can be readily detected from DNA sequencing data, making them good candidates for phylogenetic analysis. However, there are two major challenges in using CNAs to quantify intra-tumor heterogeneity and evolution.

The first challenge is that nearly all cancer sequencing studies perform bulk sequencing, where mutations are measured in tumor samples composed of mixtures of millions of different cells. While single-cell sequencing provides a higher resolution measurement of tumor heterogeneity, it remains a specialized technique that is cost prohibitive and error prone for whole genome analysis of thousands of cells [11]. Thus, we require techniques to deconvolve CNA measurements from mixed tumor samples. Typically, CNAs are detected in sequencing data by examining the depth of aligned sequencing reads to genomic regions. More specifically, segmentation algorithms use this signal to partition the genome into *segments* with the same *integer* copy number [1, 16]. When a sample is heterogeneous, i.e. composed of a mixture of distinct clones, a *fractional* copy number may be obtained for each segment instead of an integer copy number. A number of methods have been developed to infer tumor composition from fractional copy numbers, taking advantage of the fact that larger CNAs perturb thousands-millions of sequencing reads, providing a signal to infer their proportions, even with modest coverage sequencing [2, 9, 10, 16, 20, 23]. However, these methods have certain limitations that limit their applicability and performance. For example, ASCAT [16] and ABSOLUTE [2] use the data from heterogeneous samples for inferring the tumor purity (the proportion of normal clone in a sample), but they do not distinguish the copy numbers of different tumor clones. Other methods, such as THetA [23], Battenberg [20], cloneHD [9] and TITAN [10], infer the clonal composition independently for each sample by deconvolving the fractional copy numbers into the integer copy numbers of the extant clones and their proportions. However, one can obtain more information by jointly considering more samples from the same tumor [12], as successfully done for single-nucleotide mutations [5, 8, 14, 17] or non-integer copy numbers [24]. Moreover, there may be multiple ways to deconvolve fractional copy numbers, especially without imposing a structure on the inferred CNAs. Therefore, the inference of distinct clones may benefit from jointly inferring their evolution.

The second challenge in using CNAs to reconstruct tumor evolution is that one requires a model of the evolution of CNAs. Defining such a model is not straightforward because CNAs can overlap, and thus positions in the genome cannot be treated independently. Standard phylogenetic models represent a genome as a sequence of “characters” with mutations acting independently on individual characters. A number of models have been introduced to study CNA evolution, and these models can be classified into two categories. The first considers *single events* such that each of those independently affects the copy number of a single segment [3, 19]. However, these models do not account for dependency between adjacent segments in the genome. The second category considers the effects of CNAs on multiple segments as *interval events* that amplify or delete copies of contiguous segments; the most prominent such approach is MEDICC [25]. Recently, [27] and [7] improved the model in MEDICC. Specifically, [27] formally investigated the effects of interval events on segments of a single clone. In [7], the authors formalized the *Copy-Number Tree (CNT)* problem that aims to find the most parsimonious evolution of clones explained by the minimum number of interval events, and derived an integer linear program (ILP) that solves this problem. However, all of the studies applying these methods either assume that each sample is homogeneous and consisting of a single clone [26, 28] or first attempt to infer the clones independently on each sample before performing a phylogenetic analysis of CNAs [19].

In this paper, we propose an approach combining the deconvolution of fractional copy numbers from multiple samples with the inference of CNAs that describes the evolution of the clones. We introduce the *Copy-Number Tree Mixture Deconvolution (CNTMD)* problem that aims to deconvolve the fractional copy numbers into the integer copy numbers of the extant clones and their proportions such that the evolution of the clones is explained by a minimum number of copy number aberrations modeled as interval events (Fig. 1). We design a coordinate-descent algorithm for solving this problem and we compare our method with alternative approaches on real-size simulations. We find that combining the deconvolution of fractional copy numbers with a phylogenetic tree outperforms other methods. We apply our method on multi-sample sequencing data of a prostate-cancer patient [13]. Our inference shows well-supported patterns that reveal the clonal composition in terms of CNAs. The software is available at <http://compbio.cs.brown.edu/software/>.

2 Copy-Number Tree Mixture Deconvolution Problem

We start by reviewing the CNT problem, where given integer copy-number profiles one is asked to infer a *copy-number tree*, whose leaves correspond to the profiles with the minimum of events. Specifically, we define the interval events that label the edges of this tree. We conclude this section by introducing the problem of deconvolving fractional copy numbers from multiple heterogeneous samples into integer copy-number profiles of distinct clones and their proportions such that the resulting profiles form the leaves of a parsimonious copy-number tree.

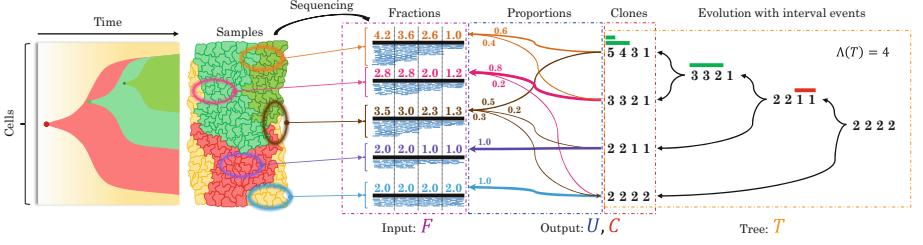


Fig. 1. Copy-Number Tree Mixture Deconvolution (CNTMD) problem. A tumor consists of heterogeneous subpopulations of cells, or clones. The normal clone is colored yellow. Five samples are bulk sequenced yielding fractional copy numbers F . We model the evolution of CNAs by a copy-number tree T (right). We combine the deconvolution of F with the inference of T . Thus, CNTMD factors F into the integer copy numbers C of the extant clones and their proportions U such that $F = CU$ and C generates a copy-number tree T with the minimum number $\Lambda(T)$ of interval events.

Following the model in [7, 25, 27], we represent a chromosome as a sequence of m segments. A *copy-number profile*, or *profile* for short, specifies the number of copies of each segment in a clone. Formally, a profile $\mathbf{c}_i = [c_{s,i}]$ is a (column) vector of m integers whose entries $c_{s,i} \in \mathbb{N}$ indicate the number of copies of segment s in a clone i . For brevity, we consider a single chromosome.

We consider mutations that amplify or delete contiguous segments. An *interval event*, or *event*, increases or decreases the copy numbers of contiguous segments of a profile \mathbf{c}_i . Formally, an event is a triple (s, t, b) with segments $s \leq t$ and integer $b \in \mathbb{Z}$. If b is positive then the event is an *amplification* and the non-zero segments between s and t are incremented by b , whereas for negative b the events is a *deletion* and the same segments are decremented by at most $|b|$. Thus, the event (s, t, b) applied on $\mathbf{c}_i = [c_{\ell,i}]$ results in $\mathbf{c}'_i = [c'_{\ell,i}]$ such that, for each segment ℓ , $c'_{\ell,i} = \max\{c_{\ell,i} + b, 0\}$ if $s \leq \ell \leq t$ and $c_{\ell,i} \neq 0$, or $c'_{\ell,i} = c_{\ell,i}$ otherwise. Thus, once a segment ℓ has been lost, i.e. $c_{\ell,i} = 0$, it can never be regained (or deleted).

We model the evolutionary process that led to n extant tumor clones by a *copy-number tree* T defined as follows.

Definition 1. Given a number n of clones, a copy-number tree is a rooted full binary tree on n leaves, such that each vertex $v_i \in V(T)$ is labeled by a profile \mathbf{c}_i and each edge (v_i, v_j) is labeled by a set $\mathcal{E}_{i,j}$ of events. The root vertex $r(T)$, corresponding to the normal clone, is diploid, i.e. $c_{s,r(T)} = 2$ for each segment s .

The requirement that T is a full binary tree is without loss of generality, as each vertex with out-degree greater than 2 of a general tree can be split into vertices of out-degree 2, and each vertex with out-degree 1 can be removed and the associated events assigned to the outgoing edge. Thus, each vertex $v_i \in V(T)$ has either zero or two children and is labeled by a profile \mathbf{c}_i . To avoid degenerate solutions, we impose a maximum copy number $c_{\max} \in \mathbb{N}$ for each segment s of any vertex v_i of T such that $c_{s,i} \leq c_{\max}$. Moreover, each leaf $v_i \in L(T)$ corresponds

to the clone i . As such, we order the vertices $V(T) = \{v_1, \dots, v_{2n-1}\}$ such that $L(T) = \{v_1, \dots, v_n\}$ and $r(T) = v_{2n-1}$. An edge $(v_i, v_j) \in E(T)$ relates a parent vertex v_i to its child v_j such that the label $\mathcal{E}(i, j)$ is a set of events that transform \mathbf{c}_i to \mathbf{c}_j . In general, the order of $\mathcal{E}(i, j)$ matters. Following a result by Shamir et al. [27], it suffices to consider an unordered set of events instead of an ordered sequence. In fact, any sequence of events, where amplifications and deletions occur in an arbitrary order, can be transformed into a *sorted* sequence, where deletions are followed by amplifications, without changing the cost of events, as defined in the following. The cost of an event (s, t, b) is the number of changes in the segment and is thus equal to $|b|$. Therefore, the cost $\Lambda(i, j)$ of an edge (v_i, v_j) is the total cost of the events in $\mathcal{E}(i, j)$, i.e. $\Lambda(i, j) = \sum_{(s,t,b) \in \mathcal{E}(i,j)} |b|$. The cost $A(T)$ of the tree T is the sum of the costs of all edges.

In the ideal case of single-cell sequencing data with no errors, each clone is a single cell and we observe the copy-number profiles $\mathbf{c}_1, \dots, \mathbf{c}_n$ of n tumor clones. As such, we wish to find the most parsimonious explanation, i.e. a minimum-cost copy-number tree T^* whose n leaves are labeled by $\mathbf{c}_1, \dots, \mathbf{c}_n$. Previously, we have shown that this problem, the Copy-Number Tree (CNT) problem, is NP-hard and we introduced an ILP formulation for solving it [7]. However, with bulk-sequencing data the observations correspond to k *samples* obtained from a single tumor in different regions or at different time points. Each sample corresponds to a *mixture* of n extant clones (leaves) of an unknown copy-number tree in unknown proportions. Recall that m is the number of segments. Our observations are thus described by the $m \times k$ *fractional copy-number matrix* $F = [f_{s,p}]$ where the *fraction* $f_{s,p} \in \mathbb{R}_{\geq 0}$ is the average copy number of segment s in sample p .

Let T be a copy-number tree with n leaves. We represent the profiles of the clones of T by the $m \times n$ *copy-number matrix* $C = [c_{s,i}]$ such that the i -th column of C corresponds to the profile \mathbf{c}_i of clone i , i.e. $C = (\mathbf{c}_1, \dots, \mathbf{c}_n)$. We say that C *generates* T if the leaves of T are labeled by the profiles in C and such that each internal vertex v_i is labeled by a profile $\mathbf{c}_i = [c_{s,i}]$ with $c_{s,i} \leq c_{\max}$ for each segment s . The $n \times k$ *usage matrix* $U = [u_{i,p}]$ describes the *mixing proportion* $u_{i,p} \in \mathbb{R}_{\geq 0}$ of clone i in sample p such that the sum $\sum_{1 \leq i \leq n} u_{i,p}$ of the mixing proportions for each sample p is 1. The observed fractional copy-numbers F are thus modeled by $F = CU$. We have the following problem (Fig. 1).

Problem 1 (Copy-Number Tree Mixture Deconvolution (CNTMD)). Given an $m \times k$ fractional copy-number matrix F , a number n of clones, and a maximum copy number c_{\max} , find an $m \times n$ copy-number matrix C generating T^* and an $n \times k$ usage matrix U such that $F = CU$ and $\Lambda(T^*)$ is minimum.

3 Method

The hardness of CNTMD is an open question. However, we suspect the problem to NP-hard, as the related unmixed version, the CNT problem, is NP-hard [7]. Moreover, other similar deconvolution problems under a tree constraint are NP-hard as well [6, 8]. As such, we design a heuristic algorithm based on the

coordinate-descent paradigm for solving a distance-based version of CNTMD where we aim to infer copy-numbers C with n clones (columns) and mixing proportions U that minimize the distance between the *observed* fractional copy numbers F and the *inferred* fractional copy numbers CU :

$$\|F - CU\| = \sum_{1 \leq s \leq m} \sum_{1 \leq p \leq k} \left| f_{s,p} - \sum_{1 \leq i \leq n} c_{s,i} u_{i,p} \right|. \quad (1)$$

Under a parsimony constraint, we impose a maximum cost Λ_{\max} on the copy-number tree T generated by C . That is, we require that C generates T such that $\Lambda(T) \leq \Lambda_{\max}$ and we consider the following problem.

Problem 2 (d-CNTMD). Given an $m \times k$ fractional copy-number matrix F , a number n of clones, a maximum copy-number c_{\max} , and a maximum cost Λ_{\max} , find an $m \times n$ copy-number matrix $C = [c_{s,i}]$ generating T and an $n \times k$ usage matrix U such that $c_{s,i} \leq c_{\max}$, $\Lambda(T) \leq \Lambda_{\max}$ and $\|F - CU\|$ is minimum.

Following the coordinate-descent paradigm, we split the variables of d-CNTMD and obtain two subproblems, where either matrix C or matrix U is fixed, with the same objective of minimizing the distance $\|F - CU\|$. An iteration t consists of two steps. In the *C-step*, we are given a usage matrix U_{t-1} and we search for a copy-number matrix $C_t = [c_{s,i}]$ minimizing $\|F - C_t U_{t-1}\|$ such that $c_{s,i} \leq c_{\max}$ and C generates T with cost $\Lambda(T) \leq \Lambda_{\max}$. Conversely, in the *U-step* we take the matrix C_t as input and seek a usage matrix U_t such that $\|F - C_t U_t\|$ is minimized.

To account for local optima, we use Q restarts with different initial usage matrices $U_{0,0}, \dots, U_{Q,0}$. We generate these usage matrices in a sparse way. This procedure yields a sequence of pairs of matrices, where for consecutive pairs $(C_{q,t}, U_{q,t}), (C_{q,t+1}, U_{q,t+1})$ it holds that $\|F - C_{q,t} U_{q,t}\| \geq \|F - C_{q,t+1} U_{q,t+1}\|$. This is because both $C_{q,t+1}$ and $U_{q,t+1}$ can be chosen equal to the previous matrices $C_{q,t}$ and $U_{q,t}$, respectively, resulting in the same distance. We iterate until $\|F - C_{q,t} U_{q,t}\|$ drops below a convergence threshold or the number of iterations reaches a specified number K .

Our algorithm thus computes Q pairs $(C_{q,K}, U_{q,K})$ of matrices for each restart $U_{q,0}$ and returns a pair (C', U^*) of matrices that minimize the distance $\|F - C_{q,K} U_{q,K}\|$. In the distance-based formulation we do not directly optimize for the cost $\Lambda(T')$ of a tree T' generated by C' . Instead, we only require that each identified matrix $C_{q,K}$ generates a copy-number tree $T_{q,K}$ with cost $\Lambda(T_{q,K}) \leq \Lambda_{\max}$ and, consequently, we have that the final matrix C' generates a copy-number tree T' with cost $\Lambda(T') \leq \Lambda_{\max}$. Thus, it may be the case that for the same usage matrix U^* there exist another copy-number matrix C'' different from C' that generates a copy-number tree T'' whose cost is $\Lambda(T'') < \Lambda(T')$ while having the same distance $\|F - C' U^*\| = \|F - C'' U^*\|$. To find the best such matrix C^* that generates a tree T^* with the smallest cost $\Lambda(T^*)$, we introduce a *refinement step* with a slightly adjusted integer linear programming (ILP) formulation of the *C-step*. Figure 2 depicts the entire procedure of the coordinate-descent algorithm.

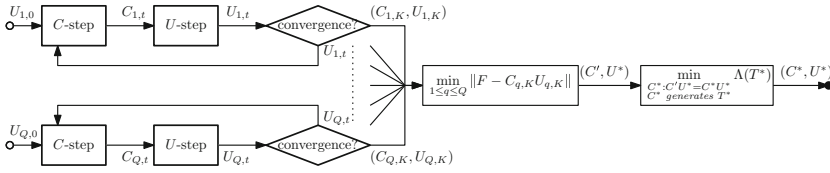


Fig. 2. Coordinate-descent algorithm. Given an initial usage matrix $U_{q,0}$, the algorithm alternately solves two distinct steps for at most K iterations. The C -step computes a copy-number matrix $C_{q,t}$ given the previous usage matrix $U_{q,t-1}$ and is followed by the U -step, which computes a usage matrix $U_{q,t}$ given $C_{q,t}$. We repeat the procedure using Q restarts with different initial usage matrices, yielding Q pairs $(C_{q,K}, U_{q,K})$ of matrices. Given these final matrices, the refinement step searches for a copy-number matrix that generates a copy-number tree with minimum cost.

We present a linear programming (LP) formulation for the U -step in Sect. 3.1 followed by an integer linear programming (ILP) formulation for the C -step in Sect. 3.2. Since the distance-based variant of the problem does not directly minimize the cost of the tree, we present in Sect. 3.3 an algorithm for finding the smallest maximum cost Λ^* with the largest decrease in the distance $\|F - CU\|$.

3.1 U-Step

In the U -step, we are given a fractional matrix F and a copy-number matrix C , and seek a usage matrix $U = [u_{i,p}]$ with real-valued entries $u_{i,p}$ minimizing the distance $\|F - CU\|$. We linearize the distance function $\|F - CU\|$ and formulate the resulting optimization problem as an LP with $O(km)$ variables and $O(km)$ constraints. To model the absolute difference in (1), we introduce variables $\bar{f}_{s,p}$ for each segment s and sample p , and model $\bar{f}_{s,p} = |f_{s,p} - \sum_{1 \leq i \leq n} c_{s,i} u_{i,p}|$ using the following linear constraints.

$$\bar{f}_{s,p} \geq f_{s,p} - \sum_{1 \leq i \leq n} c_{s,i} u_{i,p} \quad 1 \leq s \leq m, 1 \leq p \leq k \quad (2)$$

$$\bar{f}_{s,p} \geq \sum_{1 \leq i \leq n} c_{s,i} u_{i,p} - f_{s,p} \quad 1 \leq s \leq m, 1 \leq p \leq k \quad (3)$$

Moreover, we introduce variables $0 \leq u_{i,p} \leq 1$ that represent the usage of a clone i in sample p . We constraint the usages of each sample to sum to 1 using the following constraint.

$$\sum_{1 \leq i \leq n} u_{i,p} = 1 \quad 1 \leq p \leq k \quad (4)$$

Thus, we have the following LP: $\min_{\mathbf{u}, \bar{\mathbf{f}}} \sum_{1 \leq s \leq m, 1 \leq p \leq k} \bar{f}_{s,p}$ s.t. (2), (3) and (4).

3.2 C-Step

In the C -step, we are given a fractional matrix F and a usage matrix U , and seek a copy-number matrix $C = [c_{s,i}]$ with integer entries $c_{s,i}$ minimizing the

distance $\|F - CU\|$ such that $c_{s,i} \leq c_{\max}$ and C generates a tree T with $\Lambda(T) \leq \Lambda_{\max}$. Similarly, to the U -step we model the distance function $\|F - CU\|$ with variables $\tilde{f}_{s,p}$ and their corresponding constraints (2) and (3). We formulate the optimization problem of the C -step as an ILP with $O(n^2m + nm \log \Lambda_{\max} + km)$ variables and constraints. Our formulation introduces new constraints that improve upon the model introduced in [7].

We introduce binary variables $X = [x_{i,j}]$ to model the topology of T and integer variables \tilde{C} to label the vertices and edges of T . Note that C is a submatrix of \tilde{C} . Recall that T is a full binary tree (Definition 1). We construct a directed acyclic graph $G = (V, E)$ that contains all copy-number trees T with n leaves as spanning trees. More specifically, we order the vertices $V = \{v_1, \dots, v_{2n-1}\}$ such that $L(T) = \{v_1, \dots, v_n\}$ and $r(T) = v_{2n-1}$. The edge set E contains edges $\{(v_i, v_j) \mid n+1 \leq i < 2n-1, 1 \leq j < i \leq 2n-1\}$. We introduce a variable $x_{i,j}$ for each edge $(v_i, v_j) \in E$, which indicates whether (v_i, v_j) is an edge of T . To encode that T is a full binary spanning tree of G , we require that each non-root vertex has exactly one incoming edge and that each internal vertex has two outgoing edges with the following constraints.

$$\sum_{i \geq j, i \geq n+1} x_{i,j} = 1 \quad 1 \leq j < 2n-1 \quad (5)$$

$$\sum_{1 \leq j < i} x_{i,j} = 2 \quad n < i \leq 2n-1 \quad (6)$$

Integer variables $\tilde{C} = [c_{s,i}]$ where $c_{s,i} \in \{0, \dots, c_{\max}\}$ encode the profiles of each vertex v_i . Since the root vertex is diploid, we add the following constraints.

$$c_{s,2n-1} = 2 \quad 1 \leq s \leq m \quad (7)$$

From these profiles and the topology of T (as captured by variables $x_{i,j}$), we obtain the events $\mathcal{E}(i, j)$ that transform the profile \mathbf{c}_i into the profile \mathbf{c}_j and thereby the cost for the edge (v_i, v_j) . Recall that an event is a triple (s, t, b) and corresponds to an amplification if $b > 0$ and a deletion otherwise. We model the amplifications and deletions covering any segment s in $\mathcal{E}(i, j)$ with two separate variables $a_{s,i,j} \in \{0, \dots, c_{\max}\}$ and $d_{s,i,j} \in \{0, \dots, c_{\max}\}$, respectively. Note that we require $\mathcal{E}(i, j)$ to be empty when the corresponding edge (v_i, v_j) is not in T . As such, we introduce the following constraints that force variables $a_{s,i,j}$ and $d_{s,i,j}$ to be 0 when (v_i, v_j) is not in T .

$$a_{s,i,j}, d_{s,i,j} \leq c_{\max} x_{i,j} \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (8)$$

Due to these constraints, the cost of every pair (v_i, v_j) of vertices that do not form an edge of T , i.e. $x_{i,j} = 0$, is fixed to 0. Therefore, only the cost of the edges of T is computed, which significantly constrains the model and improves the performance over the formulation presented for the unmixed CNT problem [7].

Now, we consider the effect of amplifications and deletions on a segment s . As described above, we assume that deletions are applied before amplifications. Moreover, if a subset of deletions results in segment s reaching value 0, the remaining amplifications and deletions will not change the value of that segment. Similarly to [7], we distinguish four different cases. Case (a) is $c_{s,i} = 0$ and

$c_{s,j} = 0$: Since both segments have value 0, we have that, following a result in [27], the number of amplifications $a_{s,i,j}$ and deletions $d_{s,i,j}$ must be between 0 and c_{\max} . Case (b) is $c_{s,i} \neq 0$ and $c_{s,j} \neq 0$: Since $c_{s,j} > 0$, the number of deletions $d_{s,i,j}$ must be strictly smaller than $c_{s,i}$. Moreover, it must hold that $c_{s,j} + d_{s,i,j} = c_{s,i} + a_{s,i,j}$. Case (c) is $c_{s,i} \neq 0$ and $c_{s,j} = 0$: Since deletions precede amplifications, the number of deletions $d_{s,i,j}$ must be at least $c_{s,i}$. Case (d) is $c_{s,i} = 0$ and $c_{s,j} \neq 0$: Once a segment s has been lost it cannot be regained. As such, this case is infeasible.

To capture the conditions of the four cases, we introduce binary variables $z_{i,s,q}$ that provide a binary representation of the integer variable $c_{s,i}$. We define $L := \lceil \log_2(c_{\max}) \rceil + 1$. In addition, we introduce binary variables $\bar{c}_{s,i} \in \{0, 1\}$ and the following constraints such that $\bar{c}_{s,i} = 1$ iff $c_{s,i} \neq 0$.

$$c_{s,i} = \sum_{q=0}^L 2^q \cdot z_{i,s,q} \quad 1 \leq i \leq 2n - 1, 1 \leq s \leq m \quad (9)$$

$$z_{i,s,q} \leq \bar{c}_{s,i} \leq \sum_{q'=0}^L z_{i,s,q'} \quad 1 \leq i \leq 2n - 1, 1 \leq s \leq m, 0 \leq q \leq L \quad (10)$$

Since $a_{s,i,j}, d_{s,i,j} \in \{0, \dots, c_{\max}\}$, the upper bound constraints involving c_{\max} are covered. In particular, case (a) is captured in its entirety. We capture case (b) with the following constraints where $(v_i, v_j) \in E(G)$.

$$c_{s,j} \leq c_{s,i} - d_{s,i,j} + a_{s,i,j} + 2c_{\max}(3 - \bar{c}_{i,s} - \bar{c}_{j,s} - x_{i,j}) \quad 1 \leq s \leq m \quad (11)$$

$$c_{s,j} + 2c_{\max}(3 - \bar{c}_{s,i} - \bar{c}_{s,j} - x_{i,j}) \geq c_{s,i} - d_{s,i,j} + a_{s,i,j} \quad 1 \leq s \leq m \quad (12)$$

$$d_{i,j,s} \leq c_{s,i} - 1 + (c_{\max} + 1)(2 - \bar{c}_{s,i} - \bar{c}_{s,j}) \quad 1 \leq s \leq m \quad (13)$$

In fact, in the case of $x_{i,j} = 1$ (i.e., (v_i, v_j) is in T), $\bar{c}_{s,i} = 1$, and $\bar{c}_{s,j} = 1$, constraints (11) and (12) model the equation $c_{s,j} + d_{s,i,j} = c_{s,i} + a_{s,i,j}$, whereas constraint (13) ensures that $d_{s,i,j} < c_{s,i}$. Otherwise, in the case of $x_{i,j} = 0$, the constraints are always satisfied and the corresponding variables $a_{s,i,j}, d_{s,i,j}$ for every segment s are forced to 0 (which is different from the ILP formulation in [7]). Note that $d_{s,i,j}$ can be always equal to zero by constraint (13), hence we do not need to distinguish whether $x_{i,j} = 0$ or $x_{i,j} = 1$. Next, we model case (c), when $x_{i,j} = 1$, using the following constraints.

$$c_{s,i} \leq d_{s,i,j} + c_{\max}(2 - \bar{c}_{s,i} + \bar{c}_{s,j} - x_{i,j}) \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (14)$$

Finally, the following constraints, which encode that if $x_{i,j} = 1$ then $\bar{c}_{s,i} = 0$ implies $\bar{c}_{s,j} = 0$, prevent case (d) from happening.

$$1 - x_{i,j} + \bar{c}_{s,i} \geq \bar{c}_{s,j} \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (15)$$

We model the cost of an edge (v_i, v_j) as the sum of the amplifications and deletions starting at each segment s by introducing variables $\bar{a}_{s,i,j} \in \{0, \dots, c_{\max}\}$ and $\bar{d}_{s,i,j} \in \{0, \dots, c_{\max}\}$. Variables $\bar{a}_{s,i,j}$ correspond to the amplifications starting at segment s and is equal to $\max\{a_{s,i,j} - a_{s-1,i,j}, 0\}$. Symmetrically, variables $\bar{d}_{s,i,j}$ corresponds to the deletions starting at segment s and is equal to

$\max\{d_{s,i,j} - d_{s-1,i,j}, 0\}$. We model this using the following constraints.

$$\bar{a}_{s,i,j} \geq a_{s,i,j} - a_{s-1,i,j} \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (16)$$

$$\bar{d}_{s,i,j} \geq d_{s,i,j} - d_{s-1,i,j} \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (17)$$

$$a_{0,i,j} = d_{0,i,j} = 0 \quad (v_i, v_j) \in E(G) \quad (18)$$

As before, we force $\bar{a}_{s,i,j}$ and $\bar{d}_{s,i,j}$ to 0 when the corresponding pair (v_i, v_j) of vertices is not an edge of T using the following constraints.

$$\bar{a}_{s,i,j}, \bar{d}_{s,i,j} \leq c_{\max} x_{i,j} \quad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (19)$$

Now, the cost of an edge (v_i, v_j) can indeed be expressed as $\sum_{1 \leq s \leq m} (\bar{a}_{s,i,j} + \bar{d}_{s,i,j})$. Hence, the cost $\Lambda(T)$ is simply the sum of the costs of all the edges, and we require that this cost is at most Λ_{\max} with the following constraint.

$$\sum_{(v_i, v_j) \in E(G)} \sum_{1 \leq s \leq m} (\bar{a}_{s,i,j} + \bar{d}_{s,i,j}) \leq \Lambda_{\max} \quad (20)$$

The ILP is thus: $\min_{\mathbf{c}, \bar{\mathbf{f}}} \sum_{1 \leq s \leq m, 1 \leq p \leq k} \bar{f}_{s,p}$ s.t. (2), (3), (5)–(20).

3.3 Choosing Λ_{\max} to Balance Cost $\Lambda(T)$ and Distance $\|F - CU\|$

We indicate by (C^A, U^A) the matrices found by our approach with maximum cost $\Lambda_{\max} = \Lambda$ and we define $d(\Lambda) = \|F - C^A U^A\|$. First, observe that the objective function $d(\Lambda_t)$ is non-increasing with larger values of Λ_t . That is, if $\Lambda_t \geq \Lambda$ then $d(\Lambda_t) \leq d(\Lambda)$, as C^A generates T with cost $\Lambda(T) < \Lambda_t$. The parameter Λ_{\max} controls the tradeoff between the cost $\Lambda(T)$ of the tree T and the distance $\|F - CU\|$. In the following, we describe an algorithm for finding the smallest maximum cost Λ^* such that $d(\Lambda^*) = 0$.

However, requiring that $d(\Lambda^*) = 0$ is too stringent as the value $d(\Lambda_t)$ depends on the number of restarts and is further confounded by the presence of noise that may result from mapping errors or amplification biases (such as GC-content bias). It is thus reasonable to expect that $d(\Lambda^*) > 0$ and that small decreases in the value of $d(\Lambda_t)$ for any $\Lambda_t > \Lambda^*$ may be not significant due to these confounding factors. We therefore introduce the parameter ε and say that $\Lambda_2 > \Lambda_1$ provides a better solution than Λ_1 if and only if $d(\Lambda_1) - d(\Lambda_2) > \varepsilon$. Intuitively, the user-specified threshold ε controls the tradeoff between greater robustness to noise (larger ε) or more precision (smaller ε). We redefine Λ^* as the smallest integer whose solution cannot be improved by increasing the maximum cost, that is $d(\Lambda^*) - d(\Lambda_t) \leq \varepsilon$ for any $\Lambda_t \geq \Lambda^*$. Note that in a similar fashion ε plays a role in the refinement step described previously. We use the monotonicity of the function $d(\Lambda_t)$ and employ binary search for finding the value Λ^* .

4 Results

We applied our algorithm for CNTMD to simulated data and to data from two patients from a prostate cancer dataset [13]. We ran every experiment in this

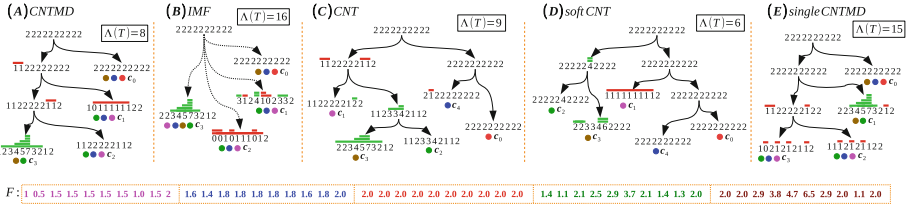


Fig. 3. Alternative methods infer trees that differ significantly from the true tree, which is inferred by our approach CNTMD. Copy-number trees inferred by the alternative methods where deletions ($s, t, -1$) are red and amplifications ($s, t, 1$) are green. (A) Shows the true tree composed of four clones c_0 (normal), c_1, c_2, c_3 with a cost of 8. This tree is correctly retrieved by CNTMD. All the alternative methods fail to infer the clonal mutation (1, 2, -1). (B) The tree inferred by IMF contains too many events and differs significantly from the true tree. (C-D) CNT and soft CNT infer clones that are very different from the true clones (E) single CNTMD splits the effect of the deletion (1, 8, -1) across two distinct clones c_2 and c_3 resulting in a cost of 15.

section on a compute cluster, and every execution lasted up to 2 days, with 160 restarts for the simulated data and 300 restarts for the real data. The implementation of our method and related details, as well as the implementation of the alternative methods are available at <http://compbio.cs.brown.edu/software/>.

We benchmarked CNTMD on simulated data, comparing its performance to several other approaches, which we now describe. The first alternative approach is a “factorization-only” approach that aims to factorize a fractional copy-number matrix F into a copy-number matrix C and a usage matrix U such that $F = CU$ without imposing a tree constraint. Published approaches to this problem perform this factorization (sometimes called deconvolution) independently on each sample [9, 19, 20, 23]—one exception is [24], but this infers non-integer copy numbers and it has not been applied to multiple samples from the same tumor. These methods do not take into account any information from the context and may provide unlikely profiles characterized by many interval events without a reasonable structure (Fig. 3B). To the best of our knowledge, there is no published method that solves the matrix factorization problem for the case where F comprises multiple vectors and C is composed of integers. Thus, we implemented *Integer Matrix Factorization (IMF)* which performs the factorization by splitting the variables, C and U , and applying a coordinate-descent algorithm in a similar fashion as the procedure described in Sect. 3.

Another class of approaches use the same copy number model as CNTMD, but assume that each sample is unmixed. One strategy is to first round the entries of F before inferring a copy-number tree. We will do this by solving the CNT problem with an ILP model [7], mimicking the strategy that has been used with MEDICC [26, 28]. We also consider a second rounding approach, which we call *soft CNT*, where we round the fractions in F either up or down such that we obtain a copy-number matrix C that generates T with minimum cost. We do this by extending the ILP formulation of the CNT problem described in [7].

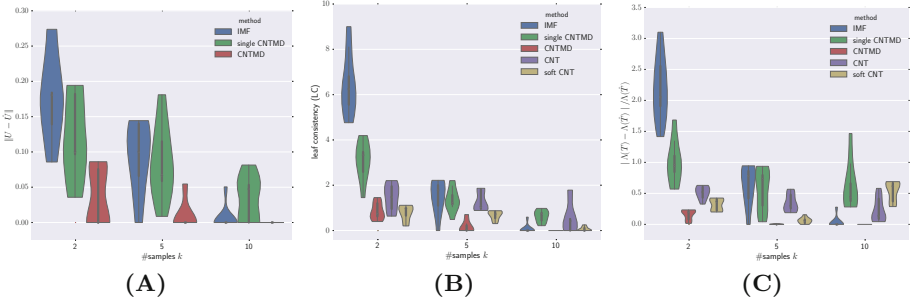


Fig. 4. CNTMD outperforms alternative methods on simulated data. Comparison of five methods across 27 simulated datasets with $k \in \{2, 5, 10\}$ samples, consisting of 4 tumor clones and a normal diploid clone, each with a total of 350 segments across 22 chromosomes. Each simulated instance was solved with n set to the true number of clones. (A) Normalized usage difference $\|U - \hat{U}\|$. (B) Leaf consistency (LC) measure. (C) Difference $|A(T) - A(\hat{T})|/A(\hat{T})$.

Finally, we also consider a variant of CNTMD, which we call *single CNTMD*. Here, we replace the interval events by single events; this is equivalent to a model where the cost of an interval event depends on the number of segments in the interval. However, the single event model is not a good representation of true copy number aberrations in cancer, as the length distribution of somatic copy number aberrations is not simply a function of length [30]. Such a copy number model was used by [19] and [3] for inferring the evolution comprising the minimum number of single events from the profile of clones inferred independently from each sample. Figure 3 shows an example highlighting the weaknesses of all the alternative methods presented above.

We compare CNTMD with the methods described above on simulated instances composed of 22 chromosomes with a total of 350 segments. These instances have the same size as real data. The number of segments per chromosome ranges from 5 to 50 and follows the distribution of the number of segments in the prostate-cancer datasets available in [13]. Using a procedure similar to the one described in [7], we randomly generate three copy-number trees, denoted by \hat{T} , which in turn were generated by copy number matrices \hat{C} composed of four tumor clones plus the normal diploid clone. We mix the leaves of each tree according to a usage matrix \hat{U} and obtain fractional copy-number matrices with $k \in \{2, 5, 10\}$ samples. For each tree and value for k , we generate three instances with different usage matrices. Thus, we consider 27 simulated instances in total.

We use three quality measures to compare the inferred tree T , inferred copy-number matrix C , and inferred usage matrix U to the simulated \hat{T} , \hat{C} and \hat{U} . We compare T to \hat{T} by considering the relative difference of events $|A(T) - A(\hat{T})|/A(\hat{T})$. To compare U to \hat{U} , we need to associate each inferred clone i to a corresponding true clone \hat{i} . Similarly to [6, 17], we search for a maximum-weight bipartite matching that minimizes the value of the *usage difference* $\|U - \hat{U}\|$ in a bipartite graph where there is an edge $(v_i, v_{\hat{i}})$ with weight $|c_i - c_{\hat{i}}|$ for

all pairs (i, \hat{i}) . To compare C to \hat{C} , we compute a maximum-weight bipartite matching on the same complete bipartite graph where the edges are weighted by a similarity metric, called *leaf consistency* (LC). This value is computed by solving an instance of CNT [7] for every pair $(\mathbf{c}_i, \mathbf{c}_j)$ of profiles where \mathbf{c}_i is a column of C and \mathbf{c}_j is a column of \hat{C} . More specifically, the LC value of $(\mathbf{c}_i, \mathbf{c}_j)$ is the minimum cost of a copy-number tree with two leaves labeled by $\mathbf{c}_i, \mathbf{c}_j$ and with an unfixed root. Note that LC is 0 if and only if $\mathbf{c}_i, \mathbf{c}_j$ are equal. Similarly to the other metrics, we compute a maximum weight bipartite matching where the edges are weighted by the LC values for every pair $\mathbf{c}_i, \mathbf{c}_j$ of columns from C and \hat{C} , respectively. We normalize the matching weight by the number of clones and chromosomes.

Figure 4 shows the results on the simulations. First, we observe that CNTMD, which combines both factorization and a proper interval tree-based model, outperforms all other methods across all number of samples. Second, we see that the quality metrics improve with increasing number k of samples for all the methods. This is especially the case for the factorization-based methods (IMF, single CNTMD, CNTMD), where differences in the clonal composition across samples provide a strong signal for deconvolution (Fig. 4A–C). In contrast, the rounding methods (CNT and soft CNT), show only a modest improvement with increasing number of samples (Fig. 4B and C), which is not surprising since rounding does not directly exploit differences in clonal composition across samples. Finally, observe that with a small number of samples ($k = 2$), CNTMD dramatically outperforms IMF (Fig. 4A and C), demonstrating how CNTMD leverages the extra information given by the tree constraint. Moreover, by not accounting for interval events, single CNTMD results in copy-number trees that are inconsistent with the simulated trees and have many more events (Fig. 4C).

4.1 Application to Prostate Cancer Dataset

We apply our approach on prostate cancer patient A22 from the dataset of Gundem et al. [13]. Patient A22 comprises 10 samples. We use the published fractional copy numbers that were obtained by the Battenberg algorithm [20], which relies on the sample purity and tumor ploidy estimated by ASCAT [16].

Since the true clonal structure of these samples is unknown, we examine the consistency of different measures on the results obtained by running CNTMD with varying number of clones $n \in \{2, \dots, 8\}$. We observe a number of patterns that suggest that there are six clones in the tumor that are distinguishable by copy number aberrations; in comparison [13] estimate 16 clones using SNVs.

First, we observe that the value of $\|F - CU\|$ decreases significantly with increasing values of n (Fig. 5). However, the rate of decrease declines for $n > 6$, suggesting that additional clones are not providing substantial gain in fitting the observed copy number fractions. Second, we find that the entries of the usage matrix U for $n \leq 6$ have well-supported proportions with reasonable mixing proportions for each clone in several samples (data not shown). In contrast, for $n > 6$, we identify clones with very low mixing proportions across samples (such

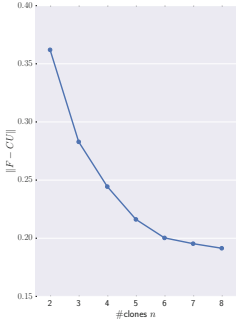


Fig. 5. The distance decreases with increasing number of clones n and stabilizes with $n > 6$ clones. The y -axis shows the normalized value of the distance $\|F - CU\|$ for each n .

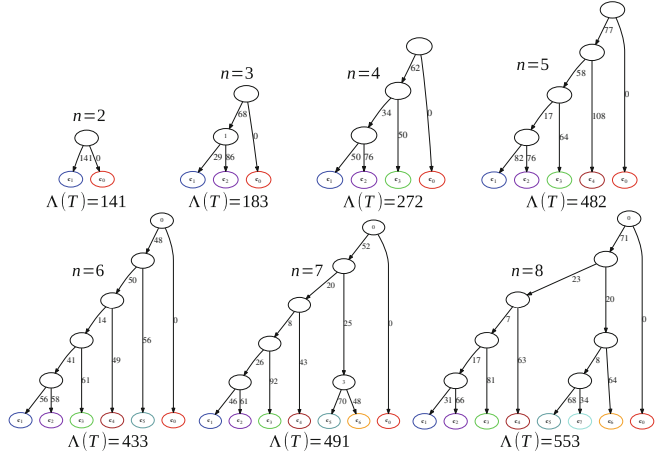


Fig. 6. Trees with $n \leq 6$ clones have a cascading topology and well-supported edges, whereas trees with $n > 6$ clones have the same cascading topology but have less-supported edges. For each copy-number tree T , we show the cost $\Lambda(T)$ and label the edges by their corresponding costs. The colors of leaves map corresponding clones in the topologies. The normal clone is red.

as \mathbf{c}_5 for $n = 7$ and \mathbf{c}_4 for $n = 8$) suggesting that the additionally inferred clones are not supported by the data. Third, we consider the topologies and costs of inferred trees with varying number of clones and find that the tree with $n = 6$ clones best describes the data. We find that most of the clonal events, which are events that are shared by all tumor clones and occur on the first branch of the tree, are consistent across the majority of the trees with $n \leq 6$ clones (Fig. 8). Moreover, the trees with $n \leq 6$ clones have a cascading topology with an additional branch for every increase in n . In contrast, with $n > 6$ clones, the trees conserve the same cascading topology and each additional clone splits a previous clone (from the tree with $n - 1$ clones) into two new sibling clones, potentially overfitting the data (Fig. 6). The total number of events, $\Lambda(T)$, stabilizes between $n = 5$ and $n = 6$ before increasing again for $n \geq 6$. The trees with $n > 6$ have several edges with only a few events as opposed to the trees with $n \leq 6$ clones. In sum, these findings suggest that the tree with $n = 6$ clones provides a good explanation of the data in comparison with the other trees that either overfit ($n > 6$) or do not accurately represent the clonal structure of the data ($n < 6$).

Finally, we examine the relationship between the inferred matrix C and the observed fractional copy number matrix F , checking whether segments with close values of F across samples are assigned the same copy number values in C , as we vary the number n of clones. We do this by partitioning the segments into *classes*

with the same evolutionary history in the inferred tree T (which is derived from the inferred C). Specifically, we define a class to be a set of segments that have the same copy-number change on all edges of T . Consequently, segments in the same class have the same copy number in all the clones. We observe that with increasing n the number of classes increases, whereas their size decreases (data not shown). However, the size and number of classes do not significantly change with $n \geq 6$. Next, we assess the consistency between these classes and F . For each pair p_1, p_2 of samples, we plot the fractional copy numbers of each segment in these samples, coloring segments by their class (overlapping segments with the same values result in larger dots). Figure 7 gives a schematic of this procedure. We see that for $n < 6$, segments in the same class are apart in at least one pair of samples (red, dark blue, and green clusters in Fig. 7), suggesting a poor fit to the data. On the other hand, for $n > 6$, segments with slightly different fractional copy numbers are separated (red/white clusters for $k = 7$ and light blue/cyan clusters for $k = 8$), suggesting overfitting of the data. Thus, this analysis also indicates that $n = 6$ appears to provide a reasonable partition into classes.

We also compare our inferred clonal copy number aberrations (CNAs) to the published clonal CNAs in [13]: We observe that several clonal events in our inferred T correspond to the these CNAs (Fig. 8): three inferred deletions on chr12 match the reported 12p LOH; a deletion with a subsequent amplification on chr13 matches the reported 13q LOH; a deletion on chr8 matches the 8p LOH; an amplification on the same chr8 matches the 8q gain; and two chr16 deletions match the reported 16q LOH. More interestingly, most of these events are clonal in the majority of the inferred trees for every n (Fig. 8). Thus, other recurrent and well-supported events in the inferred tree T are likely to be real, giving additional information about the clonal composition of these samples.

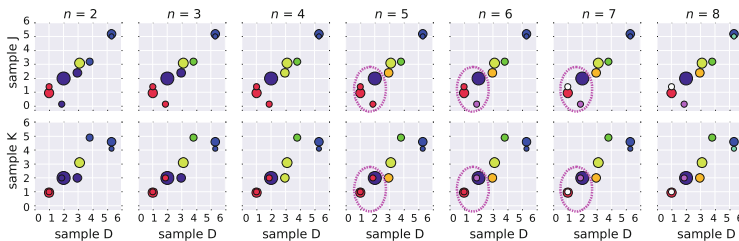


Fig. 7. Classes of segments with the same evolutionary history highlight consistency of the inferred solutions with the input data. Fractional copy numbers for three A22 prostate cancer samples: D, K and J. The largest dot contains 14 segments. The consistency of the classes improves with increasing n . The red class in $n = 5$ is composed of segments that have one copy in all the considered samples, and segments that have two copies in samples D, K and zero copies in sample J . With $n = 6$ these two subsets are separated into different classes (red and purple), while with $n = 7$ one more class (white) is introduced, potentially overfitting the data.

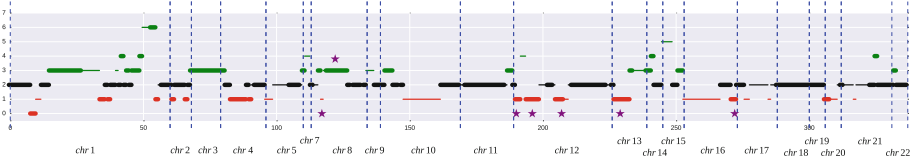


Fig. 8. Well-supported clonal events correspond to published clonal CNAs. This plot shows the copy numbers of the clonal events inferred with $n = 6$ clones. We indicate separate chromosomes with dashed blue lines. Green lines indicate amplifications and red lines indicate deletions. The lengths are proportional to the number of segments. Thick lines indicate events that are shared by the majority of the inferred trees T (with varying n). Purple stars indicate events that correspond to published clonal CNAs [13].

5 Discussion

In the paper, we formulated the Copy-Number Tree Mixture Deconvolution (CNTMD) Problem, and derived a coordinate-descent algorithm, with alternating ILP and LP steps, to solve this problem. CNTMD builds a phylogenetic tree describing copy number evolution directly from mixed samples, thus addressing an important issue in applying phylogenetic analysis to tumor samples. We show that CNTMD outperforms approaches that only perform deconvolution—thus ignoring the phylogenetic relationship between samples—or that build phylogenetic trees assuming that each sample is homogeneous, i.e. consisting of a single clone. We also apply CNTMD to a complex metastatic prostate cancer dataset, and build a phylogenetic tree containing multiple distinct clones, mixed in different proportions across samples. These results demonstrate the feasibility of our approach to real-sized datasets.

There are a number of directions for future work. On the theoretical side, the hardness of CNTMD remains open. Assuming the problem is intractable, better heuristics for solving the C -step would improve the performance with increasing number of clones. An additional avenue of investigation is to incorporate uncertainty in the segmentation of the genome into the model. Finally, one could extend the approach using more sophisticated models of genome evolution, including models that include additional genome rearrangements and complex patterns of duplication—some promising work in this direction is found in [15, 18, 22]. For practical applications, a number of improvements would be helpful. First, approaches to better address noise in the copy number fractions, using confidence intervals or posterior distributions to model the uncertainty in entries of F , are needed. Next, model selection or regularization approaches to estimate the number of clones in a tree and avoid overfitting would be helpful. For example, we report $n = 6$ clones in the prostate cancer sample A22, while the original analysis [13] reports 16 clones. This difference is likely due to the fact that [13] use single-nucleotide variants (SNVs) to identify clones. Thus, methods that simultaneously identify CNAs and perform phylogeny inference from CNAs and SNVs are an important direction for future work. Finally, one

could augment the phylogenetic reconstructions with single-cell measurements including FISH [3] or single-cell sequencing [4]. Together, these improvements would enable high-fidelity phylogenetic reconstructions of tumor evolution.

Acknowledgements. This work is supported by a US National Science Foundation (NSF) CAREER Award (CCF-1053753) and US National Institutes of Health (NIH) grants R01HG005690 and R01HG007069 to BJR. BJR is supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund, an Alfred P. Sloan Research Fellowship.

References

1. Baumbusch, L.O., et al.: Comparison of the agilent, ROMA/NimbleGen and Illumina platforms for classification of copy number alterations in human breast tumors. *BMC Genom.* **9**(1), 379 (2008)
2. Carter, S.L., et al.: Absolute quantification of somatic DNA alterations in human cancer. *Nat. Biotechnol.* **30**(5), 413–421 (2012)
3. Chowdhury, S.A., et al.: Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comput. Biol.* **10**(7), 1–19 (2014)
4. Davis, A., et al.: Computing tumor trees from single cells. *Genome Biol.* **17**, 1 (2016)
5. Deshwar, A.G., et al.: PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome Biol.* **16**(1), 1 (2015)
6. El-Kebir, M., et al.: Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics* **31**(12), i62–i70 (2015)
7. El-Kebir, M., Raphael, B.J., Shamir, R., Sharan, R., Zaccaria, S., Zehavi, M., Zeira, R.: Copy-number evolution problems: complexity and algorithms. In: Frith, M., Storm Pedersen, C.N. (eds.) *WABI 2016*. LNCS, vol. 9838, pp. 137–149. Springer, Heidelberg (2016). doi:[10.1007/978-3-319-43681-4_11](https://doi.org/10.1007/978-3-319-43681-4_11)
8. El-Kebir, M., et al.: Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell Syst.* **3**(1), 43–53 (2016)
9. Fischer, A., et al.: High-definition reconstruction of clonal composition in cancer. *Cell Rep.* **7**(5), 1740–1752 (2014)
10. Gavin, H., et al.: Titan: inference of copy number architectures in clonal cell populations from tumor whole genome sequence data. *Genome Res.* **24**, 1881–1893 (2014)
11. Gawad, C., et al.: Single-cell genome sequencing: current state of the science. *Nat. Rev. Genet.* **17**(3), 175–188 (2016)
12. Gerlinger, M., et al.: Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. *N. Engl. J. Med.* **366**(10), 883–892 (2012)
13. Gudem, G., et al.: The evolutionary history of lethal metastatic prostate cancer. *Nature* **520**(7547), 353–357 (2015)
14. Jiang, Y., et al.: Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing. *PNAS* **113**, E5528–E5537 (2016)
15. Li, Y., et al.: Allele-specific quantification of structural variations in cancer genomes. *Cell Syst.* **3**(1), 21–34 (2016)

16. Van Loo, P., et al.: Allele-specific copy number analysis of tumors. *PNAS* **107**, 16910–16915 (2010)
17. Malikic, S., et al.: Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics* **31**(9), 1349–1356 (2015)
18. McPherson, A., Roth, A., Chauve, C., Sahinalp, S.C.: Joint inference of genome structure and content in heterogeneous tumor samples. In: Przytycka, T.M. (ed.) *RECOMB 2015*. LNCS, vol. 9029, pp. 256–258. Springer, Cham (2015). doi:[10.1007/978-3-319-16706-0_25](https://doi.org/10.1007/978-3-319-16706-0_25)
19. McPherson, A., et al.: Divergent modes of clonal spread and intraperitoneal mixing in high-grade serous ovarian cancer. *Nat. Genet.* **48**, 758–767 (2016). doi:[10.1038/ng.3573](https://doi.org/10.1038/ng.3573)
20. Nik-Zainal, S., et al.: The life history of 21 breast cancers. *Cell* **149**(5), 994–1007 (2012)
21. Nowell, P.C.: The clonal evolution of tumor cell populations. *Science* **194**, 23–28 (1976)
22. Oesper, L., et al.: Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinform.* **13**(6), S10 (2012)
23. Oesper, L., et al.: THetA: inferring intra-tumor heterogeneity from high-throughput DNA sequencing data. *Genome Biol.* **14**(7), R80 (2013)
24. Roman, T., et al.: Medoidshift clustering applied to genomic bulk tumor data. *BMC Genom.* **17**(1), 6 (2016)
25. Schwarz, R.F., et al.: Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comput. Biol.* **10**(4), 1–11 (2014)
26. Schwarz, R.F., et al.: Spatial and temporal heterogeneity in high-grade serous ovarian cancer: a phylogenetic analysis. *PLoS Med* **12**(2), 1–20 (2015)
27. Shamir, R., et al.: A linear-time algorithm for the copy number transformation problem. In: *LIPICs*, vol. 54. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
28. Sottoriva, A., et al.: A Big Bang model of human colorectal tumor growth. *Nat. Genet.* **47**(3), 209–216 (2015)
29. Venkatesan, S., et al.: Tumor evolutionary principles: How intratumor heterogeneity influences cancer treatment and outcome. *ASCO* **35**, e141–e149 (2015)
30. Zack, T.I., et al.: Pan-cancer patterns of somatic copy number alteration. *Nat. Genet.* **45**(10), 1134–1140 (2013)

Quantifying the Impact of Non-coding Variants on Transcription Factor-DNA Binding

Jingkang Zhao^{1,2}, Dongshunyi Li³, Jungkyun Seo², Andrew S. Allen^{1,3},
and Raluca Gordân^{1,3,4} (✉)

¹ Center for Genomic and Computational Biology,
Duke University, Durham, NC 27708, USA
raluca.gordan@duke.edu

² Program in Computational Biology and Bioinformatics,
Duke University, Durham, NC 27708, USA

³ Department of Biostatistics and Bioinformatics,
Duke University, Durham, NC 27708, USA

⁴ Department of Computer Science, Duke University, Durham, NC 27708, USA

Abstract. Many recent studies have emphasized the importance of genetic variants and mutations in cancer and other complex human diseases. The overwhelming majority of these variants occur in non-coding portions of the genome, where they can have a functional impact by disrupting regulatory interactions between transcription factors (TFs) and DNA. Here, we present a method for assessing the impact of non-coding mutations on TF-DNA interactions, based on regression models of DNA-binding specificity trained on high-throughput *in vitro* data. We use ordinary least squares (OLS) to estimate the parameters of the binding model for each TF, and we show that our predictions of TF binding changes due to DNA mutations correlate well with measured changes in gene expression. In addition, by leveraging distributional results associated with OLS estimation, for each predicted change in TF binding we also compute a normalized score (*z*-score) and a significance value (*p*-value) reflecting our confidence that the mutation affects TF binding. We use this approach to analyze a large set of pathogenic non-coding variants, and we show that these variants lead to significant differences in TF binding between alleles, compared to a control set of common variants. Thus, our results indicate that there is a strong regulatory component to the pathogenic non-coding variants identified thus far.

Keywords: TF-DNA binding · Non-coding variants · Regression models

1 Introduction

Single nucleotide variants (SNVs) play important roles in the pathogenesis of many complex diseases [16]. For mutations that occur within protein-coding

J. Zhao and D. Li—These authors contributed equally to this work.

genes, there are established metrics (e.g. SIFT [29] and PolyPhen [1]) that attempt to quantify the effect of a variant on gene function. However, coding variants are only a small fraction of all genetic variants: recent studies estimate that $\sim 93\%$ of disease- and trait-associated human genetic variants fall within non-coding genomic regions [24], and their functional impact is difficult to assess and quantify.

Non-coding variants can play a functional role in the cell by disrupting interactions between transcription factors (TFs) and their genomic target sites [16]. TFs are regulatory proteins that bind short DNA sites, typically in the neighborhood of the regulated genes, and promote or repress gene expression. Predicting the effect of SNVs on TF binding is an important area of research still lacking good solutions. Binding models for many human TFs are currently available [14,22,31,41] in the form of position weight matrices (PWMs). A PWM is a matrix of scores (or weights) for each nucleotide at each position in a TF binding site. Although they are easy to use and visualize, PWMs make the assumption that individual base pairs in a TF binding site (TFBS) contribute independently to the binding affinity. This assumption does not always hold [5,11,37,38,42]. Nevertheless, although it is now recognized that PWMs cannot accurately capture TF-DNA binding affinity [20,33,35], current methods for determining whether a SNV is likely to affect TF-DNA binding are based on differences in PWM scores [2,26,36,39]. Such methods are generally able to detect large changes in TF binding affinity (from high affinity to non-specific binding), but they ignore less drastic changes, which can have important phenotypic effects (e.g. [32]).

Another drawback of using PWM models to predict the effect of SNVs is the fact that many mammalian TFs have several PWMs available in the literature, oftentimes from databases such as Transfac [23], Jaspar [21], UniPROBE [28], or Cis-BP [41]. Different PWMs can result in different predictions on whether or not a SNV will affect binding of the TF of interest, and there is no objective method to choose the best PWM to use, as quality metrics are not reported for these models. Ideally, a method for characterizing the effect of non-coding SNVs on TF binding should be able to capture both large and small changes in binding, as long as the changes are ‘significant’ given the quality/precision of the model.

Here, we present a new method for assessing the impact of non-coding variants/mutations on TF-DNA binding. Based on high-throughput data from protein-binding microarray (PBM) experiments [8,9], we build k -mer-based models of TF binding specificity, estimating the model parameters with ordinary least squares (OLS). We use the estimated regression coefficients, as well as the variance-covariance matrix, to compute for any given mutation: (1) a quantitative prediction of the change in TF binding due to the mutation, and (2) a z -score and p -value indicating the significance of the predicted change, given the model properties. Our approach is novel compared to previous regression models trained on PBM data [3,40] because, by using OLS, we obtain not only estimates of the regression coefficient for each k -mer, but also the variance

of the coefficient estimates. Thus, our predictions of the effects of mutations on TF-DNA binding implicitly take into account the quality of the training data and model, such that in the case of poor predictive models we require a larger change in binding for a mutation to be called significant. In addition, the computed variance in the estimates of the model parameters allows us to make objective choices between different models corresponding to the same TF. We validate our models using gene expression data from high-throughput reporter assays [15, 27], and we apply them to predict the effects of pathogenic SNVs [34] on TF binding.

2 Data and Methods

2.1 Universal Protein-Binding Microarray (PBM) Data

Accurate methods for predicting the effect of SNVs on TF binding require accurate models of TF-DNA binding specificity. Here, to train such models we use high-throughput *in vitro* data from universal PBM assays [9]. Each universal PBM data set is specific to one TF, and it contains quantitative measurements of the binding specificity of that TF for $\sim 40,000$ DNA sequences. The PBM protocol is described in detail in [8]. Briefly, double-stranded DNA molecules attached to a glass slide (microarray) are incubated with an epitope-tagged TF. To detect the amount of TF bound to each DNA spot, the microarray is labeled with a fluorophore-conjugated antibody specific to the epitope tag, and scanned using a microarray scanner. The fluorescence intensity of each DNA spot provides a quantitative measurement of the TF specificity for the DNA sequence in that spot.

PBM experiments are typically performed using Agilent microarrays printed with custom 60-bp DNA sequences [8]. For a ‘universal’ PBM array design, the DNA sequences printed on the array are computationally designed according to a deBruijn sequence of order 10 over the {A,C,G,T} alphabet, which, by definition, is guaranteed to contain all possible 10-bp DNA sequences, with each 10-mer occurring once and only once. To computationally generate the DNA library, the deBruijn sequence is split into sequences of 35 or 36 bases, depending on the design [9, 41], and the remaining 25 or 24 bases, respectively, are set to the complement of a primer used to double-strand the DNA molecules. Table 1 shows as example one of the 973 PBM data sets [6, 41] used in our analysis of pathogenic non-coding variants (Sect. 3.4). The PBM data sets used in Sects. 3.2 and 3.3 are: pTH5080_HK and pTH5080_ME for Creb [41], Tcf1_2666.2 for Hnf1 [7], Foxa2_2830.2_v1 for Foxa, Hnf4a_2640.2_v1 for Hnf4, and Gata3_1024.3_v1 for Gata [5].

2.2 Massively Parallel Reporter Assay (MPRA) Data

To validate that the quantitative predictions of TF binding changes made using our OLS k -mer models (Sect. 2.3) are biologically relevant, we leveraged high-throughput gene expression data from massively parallel reporter

Table 1. Example of universal PBM data set for transcription factor Arid5a [5].

DNA sequences of length $L = 60$	TF binding intensity
TTGAATCAAT.....GTCCGTGCTG	74573.8653
CCAAGACAGT.....GTCCGTGCTG	45399.3011
CGCAAATATT.....GTCCGTGCTG	40440.2397
.....
ACTTCCGATA.....GTCCGTGCTG	39895.9250

assays (MPRA) [27]. Briefly, in an MPRA experiment one first synthesizes tens of thousands of oligonucleotides that contain a library of regulatory elements (enhancers), each coupled to a short DNA tag. The oligonucleotides are used to generate a pool of plasmids, where each plasmid contains one of the regulatory elements of interest upstream of an open reading frame followed by the sequence tag corresponding to that regulatory element. The pool of plasmids is co-transfected into cells, where the regulatory elements drive transcription of mRNA molecules containing the tags. The tags in the reporter mRNAs, as well as the original plasmid pool, are sequenced and counted. The ratio of these counts, or the logarithm of the ratio, is taken as a measurement of the gene expression driven by each regulatory element [27].

Here, we use MPRA data from two recent studies. Melnikov et al. [27] reported the expression levels of a reporter gene (an inert open reading frame) downstream of variants of a synthetic, 87-nt cAMP-regulated enhancer. The mutants were either generated by single nucleotide substitutions (for a total of $87 \times 3 = 261$ variants) or by random multiple 1-bp nucleotide substitutions, introduced at a rate of 10% per position ($\sim 27,000$ variants). The expression level of each variant was reported as the median of the mRNA-based counts normalized by the DNA-based counts, taken over multiple tags. In our analyses, we used the natural logarithm of the ratios of the expression levels of mutants to the expression level of the wild-type sequence.

Kheradpour et al. [15] reported the expression levels of a small number of enhancer variants for four TFs: Hnf1, Foxa, Hnf4 and Gata. Selected wild-type enhancer regions were centered on motif matches and the mutants were generated by multiple approaches such as motif removal, maximum 1-bp decrease, least 1-bp change, etc. The expression level was expressed as the binary logarithm of the mean value of the ratio of the mRNA to plasmid counts. In our analyses, we used the proportion of change in gene expression due to the mutations, relative to the expression of the wild-type sequence.

2.3 Training k -mer Regression Models of TF Binding Specificity Using Ordinary Least Squares (OLS)

In a universal PBM experiment, TF binding to each of the $\sim 40,000$ pre-designed L -bp DNA sequence is measured as fluorescent signal (Table 1). We apply a

logarithmic transformation to the fluorescent signal, which makes the experimental noise uncorrelated with the signal, and we use the natural log-transformed fluorescence intensities as the dependent variable Y . As independent variables X , we use the counts of each k -mer within the L -bp DNA sequences, with the value of k decided based on validation experiments (Sect. 3.1). Since the DNA is double-stranded, and binding of TFs is not strand-specific, we regard each sequence and its reverse complement as the same feature. Thus, the number of features n_k for a k -mer model is $4^k/2$ when k is an odd number, and $(4^k - 2^k)/2 + 2^k$ when k is even.

Suppose there are a total of N L -bp sequences. We convert each sequence into the counts of all n_k k -mers in an overlapping fashion, generating a $N \times n_k$ covariate matrix X . There is an inherent restriction for the rows of the matrix. For any row i , the sum of the counts is $x_{i1} + \dots + x_{in_k} = L - k + 1$, which is due to the fact that every L -bp sequence contains $L - k + 1$ overlapping k -mers. The linear dependency of the n_k features renders the intercept term redundant, and we therefore train our models without the intercept term:

$$Y_i = \beta_1 x_{i1} + \dots + \beta_{n_k} x_{in_k} + \epsilon_i \quad (1)$$

After the intercept term is removed from the model, multicollinearity among the covariates is no longer a problem and we can compute the ordinary least square (OLS) estimates for the β 's, as well as the covariance matrix $\hat{\Sigma}$, whose diagonal contains the variance in the coefficient estimates:

$$\hat{\beta} = (X'X)^{-1}X'Y \quad (2)$$

$$\hat{\Sigma} = \frac{(Y - X\hat{\beta})'(Y - X\hat{\beta})}{N - n_k}(X'X)^{-1} \quad (3)$$

2.4 Statistical Testing Using OLS k -mer Models of TF Binding Specificity

By assuming normality on the error vector $\epsilon \sim N(0, \sigma^2 I)$ in (1), we can perform statistical tests on β 's as well as linear combination of β 's. Given a vector c of length n_k , the null and alternative hypotheses to test a linear combination of β 's are the following:

$$H_0 : c'\beta = 0$$

$$H_1 : c'\beta \neq 0$$

A t-statistic can be built using the estimated covariance matrix:

$$t = \frac{c'\hat{\beta}}{\sqrt{c'\hat{\Sigma}c}} \sim t_{N-n_k} \quad (4)$$

In fact, since we have a large number of observations, the distribution of the test statistics is approximately normal, and we can thus compute a z -score for $c'\hat{\beta}$.

2.5 Using OLS k -mer Models to Predict the Effect of SNVs on TF-DNA Binding

The method above can be directly applied to predict the effect of single base-pair variants on TF binding. To illustrate this, we provide an example for a mutation (A to C) that affects a binding site for TF Creb1 (Table 2). We used 6-mer features to train a regression model from universal PBM data for Creb1, available from [41]. In a 6-mer model, there are a total of 2080 features. From the modeling step, we can derive the estimates of coefficients $\hat{\beta}$ for all 6-mers, as well as the covariance matrix estimate $\hat{\Sigma}$.

Table 2. Single base-pair mutation overlapping a binding site for TF Creb1. The wild-type and mutated binding sites are shown in bold. The mutated position is underlined. (The 6-mers in parentheses are the reverse complements of 6-mers in the original sequence. In these cases the reverse complement 6-mers were used as features because they are alphabetically ranked lower than the corresponding 6-mers on the forward strand.)

Wild-Type	Mutant
CCCAT TG<u>A</u>CGTCAATGGG	CCCAT TG<u>C</u>CGTCAATGGG
CATTG <u>A</u>	CATTG <u>C</u>
ATTG <u>A</u> C	ATTG <u>C</u> C
TTG <u>A</u> CG (CGTCAA)	TTG <u>C</u> CG (CGGCAA)
TG <u>A</u> CGT (ACGTCA)	TG <u>C</u> CGT (ACGGCA)
G <u>A</u> CGTC	G <u>C</u> CGTC (GACGGC)
<u>A</u> CGTCA	<u>C</u> CGTCA

Given a k -mer model, a single base-pair mutation leads to a change in every k -mer in a $2k - 1$ bp region centered at the mutated base (Table 2). Thus, the total change is:

$$\sum_{p=1}^k (\hat{\beta}_{j_p} - \hat{\beta}_{i_p})$$

where j_p is the index of the p th k -mer in the mutated sequence, and i_p is the index of the corresponding k -mer in the original sequence.

For the example in Table 2, the mutation causes a change in 6 consecutive 6-mers, and the total effect of the mutation is:

$$\hat{\beta}_{CATTGC} + \hat{\beta}_{ATTGCC} + \hat{\beta}_{CGGCAA} + \hat{\beta}_{ACGGCA} + \hat{\beta}_{GACGGC} + \hat{\beta}_{CCGTCA} - \hat{\beta}_{CATTGA} - \hat{\beta}_{ATTGAC} - \hat{\beta}_{CGTCAA} - \hat{\beta}_{ACGTCA} - \hat{\beta}_{GACGTC} - \hat{\beta}_{ACGTCA}$$

In vector notation, this effect can be written in terms of 6-mer coefficients as:

$$c' = (0, \dots, 1, \dots, -2, \dots, -1, \dots, 1, \dots, -1, \dots, 1, \dots, 1, \dots, 1, \dots, -1, \dots, -1, \dots, -1, \dots, 0)$$

where the coefficients are null for all 6-mers that do not contribute to the total effect.

Next, we use (4) to compute the test statistic t for the difference in predicted binding affinity ($c'\hat{\beta}$) between the mutant and the wild-type sequences. For the specific mutation in our example, the difference in binding affinity is -1.69 . After normalization, the z -score for this mutation is -42.07 (p -value $< 10^{-10}$), indicating a significant decrease in Creb1 binding affinity.

3 Results

3.1 OLS 6-mer Models Can Accurately Predict TF Binding Intensity

To check the accuracy of the k -mer models and determine the best value for k , we used 115 TFs from the Cis-BP database [41] for which universal PBM data is available from two distinct array designs. We learned OLS k -mer models from one array design, and tested them on the independent data obtained using the second array design. The Pearson correlation coefficient (R) between our predicted TF binding intensities and the measured intensities from both the training and the test data sets are summarized in Fig. 1. We note that PBM experiments performed on different arrays are not replicate experiments, as the array designs contain different DNA sequences. In addition, data quality is highly variable across the PBM data sets, so we expect the performance of any model trained on these data sets to also vary. Importantly though, our 6-mer OLS

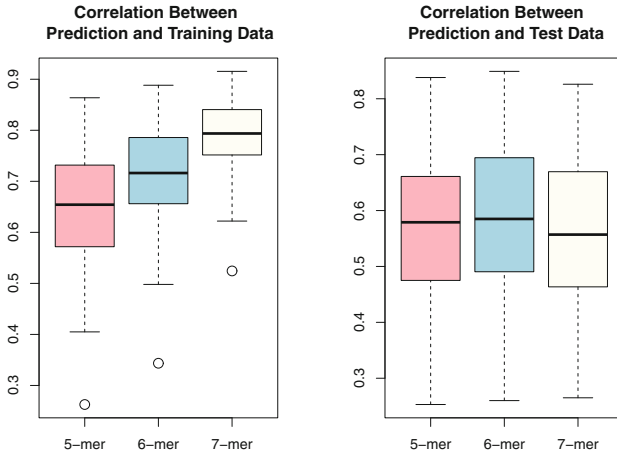


Fig. 1. Performance of OLS k -mer models for $k = 5, 6$, and 7 . Box plots show Pearson correlation coefficients between predicted and measured TF binding intensities, for 115 TFs [41] with data available from two universal PBM designs. Left: predictions compared to binding data from the PBM design used for training. Right: predictions compared to binding data from an independent PBM design.

models are designed to implicitly take data quality into account when predicting changes in TF binding due to mutations, as described in Sect. 2.5.

Figure 1 shows the performance of 5-mer, 6-mer, and 7-mer OLS models, which have 512, 2080, and 8192 features, respectively. For $k = 8$, the models have a total of 32,896 features. Since we only have $\sim 40,000$ observations, models with $k \geq 8$ run into dimensionality problems and we cannot get OLS estimates for the parameters. Among 5-mer, 6-mer, and 7-mer models, we found that 7-mers models perform best on the training data (Fig. 1, left panel). However, on independent test data from a different array design, 7-mer models perform worse than 6-mers models (Fig. 1, right panel), indicating that they are likely over-fitting the data. Thus, our results indicate that $k = 6$ results in models that have the best accuracy in predicting TF binding intensity for new DNA sequences. All results presented below use 6-mer OLS models.

The main goal of our method is to predict *changes* in TF binding, not absolute TF binding levels. Nevertheless, to ensure that our 6-mer OLS models are accurate in predicting TF binding levels, we compared them to previous models trained and tested on PBM data from different array designs. For this comparison, we used the PBM data from the DREAM5 TF-DNA Motif Recognition Challenge [40], which includes independent data sets obtained using two different array designs, for 66 mouse TFs. In the challenge, PBM intensity data were provided only for one array design, and the performance of each algorithm was evaluated by assessing the prediction accuracy on the other array design, using the Pearson correlation coefficient (R). Weirauch et al. [40] used several normalization techniques to transform the PBM data before using it for training and testing, and for each algorithm they selected the combination of normalization steps that resulted in the best prediction accuracy on the test data. In contrast to their approach, we use the PBM data directly in our algorithm, applying only a logarithmic transformation to all PBM intensities, and thus keeping the test PBM data completely independent from the training step. The performance of our 6-mer OLS method was above average compared to the 15 methods tested in [40]. Thus, we conclude that the accuracy of our method in predicting TF binding intensity is comparable to existing algorithms, with our method having the unique advantage that it implicitly incorporates data quality into the TF binding models.

3.2 TF Binding Change Predictions Based on OLS 6-mer Models Correlate Well with Gene Expression Changes

To validate that our OLS 6-mer models are able to quantitatively predict the effect of nucleotide mutations, we leveraged high-throughput reporter expression data generated using massively parallel reporter assays (MPRA) [15, 27]. First, we focused on MRPA data for an 87-bp synthetic enhancer that contains four binding sites for transcription factor Creb1 [27]. Melnikov et al. [27] reported expression measurements for the wild-type enhancer (Fig. 2a), for all possible single base pair mutations, as well as a large number of enhancer variants with multiple mutations randomly distributed across the enhancer region.

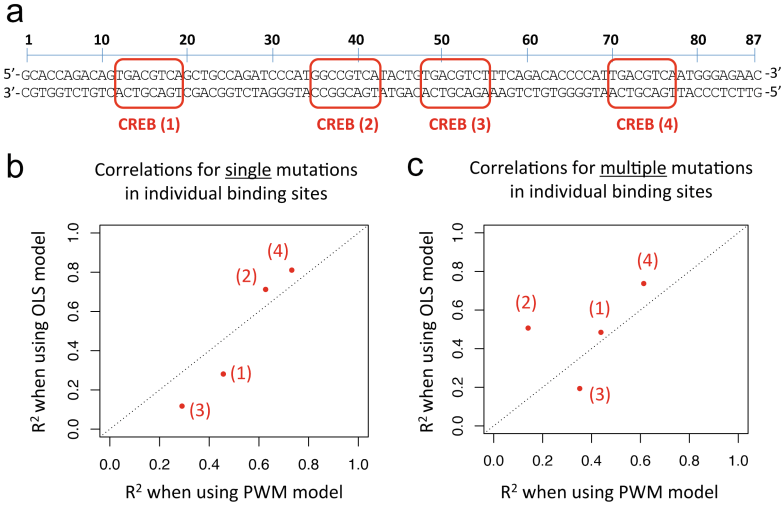


Fig. 2. Correlations between measured gene expression changes and TF binding changes predicted by OLS and PWM models, for individual TF binding sites. **(a)** Creb1-regulated enhancer. Red rectangles mark the four annotated Creb1 binding sites. **(b)** Correlations for variant enhancers with single 1-bp mutations in individual binding sites. **(c)** Correlations for variant enhancers with multiple 1-bp mutations in individual binding sites. R^2 represents the squared Pearson correlation coefficient between measured change in gene expression and predicted change in TF binding.

The expression values are reported as ratios of tag counts in the reporter mRNA versus tag counts in the plasmid pool (see Sect. 2.2). Based on the expression values reported in [27], we computed for each mutant enhancer the natural logarithm of the ratio between the expression of the mutant and the expression of the wild-type enhancer. We asked whether these changes in gene expression can be explained, at least in part, by changes in Creb1-DNA binding predicted according to: (1) our OLS 6-mer model for TF Creb1; and (2) the mouse Creb1 PWM reported in the Cis-BP database (motif identifier M0297.1.02) [41]). To score DNA sites according to the PWM we used the log-likelihood (LLR) score, i.e. we computed the base 2 logarithm of the ratio between the probability of the site according to the PWM model, and the probability of the site according to a uniform background model over the four nucleotides.

Before applying our approach to predict the effect of mutations on Creb1-DNA binding, we verified the accuracy of Creb1 OLS 6-mer models trained on PBM data, and we selected the most accurate model. There are two universal PBM datasets available for mammalian Creb1, from two distinct universal designs, denoted HK and ME [41] (Sect. 2.1). We trained OLS 6-mer models on each array design, and we compared the models according to their predicted variance for the parameter estimates, i.e. the diagonal of the covariance matrix $\hat{\Sigma}$ (see (3)). The parameter estimates for the model trained on the ME data set showed lower variances (Mann-Whitney U test p -value $< 2.2 \times 10^{-16}$), and thus it was selected as the final Creb1 OLS 6-mer model.

We first compared the OLS and PWM models on variant enhancers with single bp mutations in each of the four Creb1 binding sites (defined as shown in Fig. 2a). For each binding site, we asked how well the measured gene expression changes due to 1-bp mutations within the binding site correlate with the predicted changes in TF binding. The OLS model performed better than the PWM for mutations in sites 2 and 4 (where both models have good prediction) and worse than the PWM for mutations in sites 1 and 3 (where both models performed poorly) (Fig. 2b).

Next, we compared the OLS and PWM models on enhancers with multiple 1-bp mutations in each of the four Creb1 binding sites. The OLS model outperformed the PWM on three of the four binding sites (Fig. 2c). This result was somewhat expected. Unlike our OLS k -mer models, PWM models cannot capture dependencies between positions within TF binding sites, and this shortcoming can lead to poor predictions when multiple mutations are introduced in a site.

Finally, we compared the OLS and PWM models on enhancers with multiple 1-bp mutations in regions that cover several of the Creb1 binding sites (Fig. 3). For such regions, using the OLS 6-mer model is straightforward, since the model can be applied to predict TF binding for sequences of any width. In contrast, PWM models have a fixed width. To apply the PWM model to longer regions, we used a sliding window of size 8 (the same size as the Creb1 PWM), we scored each window according to the PWM, and we summed up the scores above a certain cutoff, expressed in terms of the maximum LLR score that can be obtained using the PWM model (e.g. 20% the maximum score, 30%, 40%, 50%, 60%, etc.). We also tested other approaches to score long DNA regions using PWMs, such as the GOMER model [13], but the thresholding approach described above worked best. We found that a cutoff of 60% leads to the best performance of the PWM model, so we used this cutoff in our comparisons. Figure 3 shows that as we include more binding sites in our analysis, the performance of the PWM decreases, reaching an R^2 of 0.27 when all four binding sites are included (Fig. 3c, left panel). In contrast, the OLS model continues to perform well regardless of the number of binding sites included in the analysis, and it constantly achieves correlations of 0.49 or higher (Fig. 3, right panels). We also tested additional Creb1 PWM models, including the curated human Creb1 motif from the Hoco-Moco database [17] (downloaded from Cis-BP [41], motif identifier M6180.1.02), which achieved correlations <0.1 in all analyses of mutations in multiple binding sites. Overall, the Cis-BP motif M0297.1.02 resulted in the highest correlations with the gene expression data. Thus, we focused on this motif for all comparisons described in this section.

Our results show that changes in TF binding, predicted using the OLS 6-mer model, can explain $\sim 50\%$ of the change in gene expression due to DNA mutations. This fraction is remarkable, given the complexity of gene regulation. We do not expect TF binding changes to completely explain gene expression changes, nor to correlate linearly with expression changes observed in the cell. The large correlation between changes predicted by the OLS model and measured changes in gene expression demonstrate that our predictions are quantitative and biologically relevant.

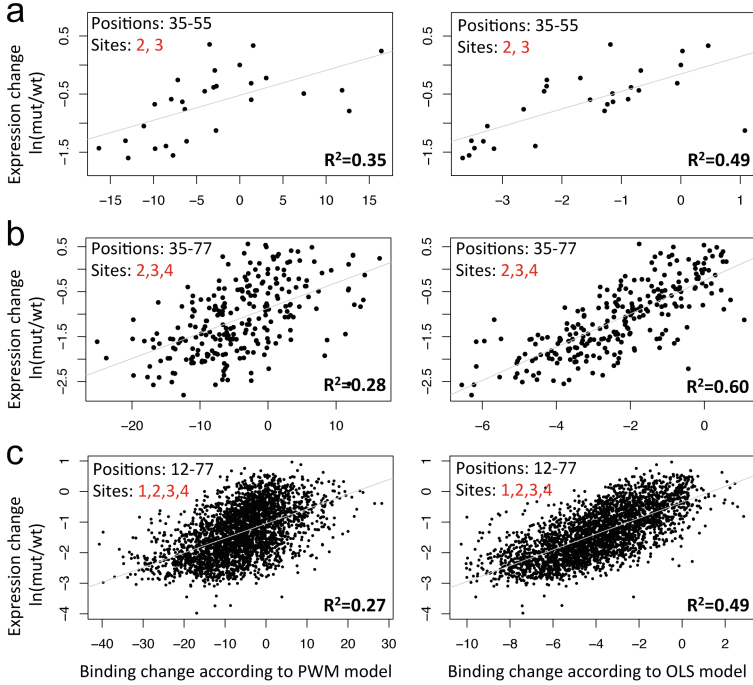


Fig. 3. Correlations between gene expression changes and TF binding changes predicted by OLS and PWM models, for regions containing multiple TF binding sites and multiple mutations. **(a)** Correlations for variant enhancers with multiple 1-bp mutations at positions 35–55 in the Creb1-regulated enhancer (see Fig. 2a), covering two Creb1 binding sites. **(b)** Similar to panel (a), but for mutations at positions 35–77 in the Creb1-regulated enhancer, covering 3 binding sites. **(c)** Similar to panel (a), but for mutations at positions 12–77 in the Creb1-regulated enhancer, covering 4 binding sites. Gray lines show the linear fit obtained using the R `lm` function. R^2 represents the squared Pearson correlation coefficient.

3.3 Making Binary Predictions of TF Binding Changes Using OLS 6-mer Models

Another application of our 6-mer OLS models is binary classification of mutations into those that affect TF binding (and thus are likely to affect gene expression) versus those that do not affect TF binding (and thus are less likely to affect gene expression). To illustrate this application of our models, we used MRPA data generated by Kheradpour *et al.* [15] for putative regulatory regions centered at binding sites for four TFs (Hnf1, Foxa, Hnf4 and Gata), and variants of these regions where the TFBSs are mutated. Several types of mutants were tested for each regulatory region: scrambled binding site, removal of binding sites, 1-bp mutations that caused the maximum increase or decrease in PWM score, 1-bp mutation that caused the minimum change in PWM score, and a random 1-bp change. Thus, compared to the MPRA data used in Sect. 3.2, the data used

here does not comprehensively cover all possible single-bp mutations. In addition, unlike the well-characterized Creb1-regulated enhancer used in Sect. 3.2, the genomic regions included in this MPRA data set [15] are *putative* enhancers. Thus, some of the enhancers may not be active, or they may not be regulated through the TFBSs in the tested genomic regions. For this reason, we filtered out the wild-type regulatory sequences with expression levels lower than 0.5, as suggested by the authors [15]. In addition, for both wild-type and mutant sequences, we used the replicate MPRA data sets to filter out sequences for which the replicates did not agree (p -value < 0.05 according to a Mann-Whitney U test applied to the reporter expression data for the two replicates, over the 10 different tags used for each sequence).

After the filtering steps described above, the number of sequences for each TF was: 27 (wild-type, mutant) pair sequences for Hnf1, 53 pairs for Hnf4, 29 pairs for Gata, and 20 pairs for Foxa. For each TF, the set of (wild-type, mutant) pairs was dichotomized into pairs for which the wild-type and mutant sequences have either similar or distinct expression values. The calls were made using a Mann-Whitney U test that compared the expression values of the two sequences, over all tags used in the MPRA experiment. The U test p -value cutoff was set individually for each TF, as the 5% quantile of the empirical distribution of p -values obtained by testing the differences between replicate experiments. Thus, for each TF we obtained a ‘positive’ set of (wild-type, mutant) pairs, for which the mutation significantly affected gene expression, and a ‘negative’ set of (wild-type, mutant) pairs, for which the mutation did not have an effect on gene expression.

Next, using the dichotomized expression data for (wild-type, mutant) pairs, we asked whether differences in TF binding, as predicted by our 6-mer OLS models or by PWMs of the four TFs (Hnf1, Foxa, Hnf4 and Gata), are predictive of changes in gene expression. To evaluate each binding model we used the area under the receiver-operating characteristic (ROC) curve for low false positive rates (< 0.2). For each mutation, we predicted that it either has or does not have an effect on TF binding and gene expression, according to whether the predicted TF binding change is above a cutoff. For PWM models we used cutoffs for the LLR score. For OLS 6-mer models we used cutoffs according to the z -score. As shown in Table 3, for three of the four TFs our OLS models outperformed the PWM models.

3.4 Analysis of Pathogenic Non-coding Variants

To further illustrate how OLS models can be used to analyze non-coding variants, we performed a broad analysis of all non-coding pathogenic SNVs annotated in the Human Gene Mutation Database (HGMD[®]) [34] and ClinVar [18]. Starting with 101,833 SNVs, we excluded any variants that overlapped with consensus coding sequences, leaving 4,655 unique variants. Next, we removed variants considered to reside within coding/canonical splice of any Ensembl coding transcript. We also excluded the variants on sex chromosomes or mitochondrial

Table 3. Comparison between 6-mer OLS models and PWM models for four TFs with MPRA data available from [15]. The numbers represent areas under the receiver-operating characteristic (ROC) curve, for false positive rates between 0 and 0.2. The maximum value for the area under the curve is 0.2. The expected value for random models is 0.02. Results show that 6-mer OLS models outperform PWM models for 3 out of the 4 TFs tested.

	Hnf1	Hnf4	Gata	Foxa
PWM models	0.04556	0.02082	0.07014	0.04835
6-mer OLS models	0.06667	0.04868	0.05046	0.05275

chromosomes, leaving a total of 3,422 unique non-coding pathogenic autosomal variants for analysis.

We also selected a set of control variants among the common variants annotated in the 1000 Genomes Project [4], following similar filtering steps. We first downloaded all SNVs from phase 3 of the 1000 Genomes Project, and excluded all rare variants (i.e. variants with minor allele frequency < 0.01). To obtain non-coding variants, we annotated the filtered variants using the Variant Effect Predictor (VEP) tool [25], and we excluded variants annotated to reside in a coding region. After this process, a total of 11.9 million non-coding SNVs were retained from 84.4 million 1000 Genomes variants. Finally, we randomly selected 3,422 non-coding autosomal variants that followed a similar genomic distribution as the pathogenic variants.

We trained 6-mer regression models for all 973 PBM data sets available for human and mouse TFs [6, 41], and applied our models to predict the binding changes due to SNVs in the pathogenic and control data sets. For each SNV, we took the maximum absolute value of the 973 predicted z -scores as the measure of the binding change due to the SNV. Figure 4 displays the empirical cumulative density functions of the predicted binding changes for the 3,422 variants in each data set. Our result shows that the binding changes caused by the pathogenic variants are significantly larger than the changes caused by the control variants (Mann-Whitney U test p -value $< 2.2 \times 10^{-16}$), indicating that there is a strong regulatory component for the annotated pathogenic variants.

4 Discussion

We developed a new method to assess the impact of non-coding mutations on TF-DNA binding, using high-throughput PBM data. Such data is currently available for almost 1,000 mammalian TFs [6, 41] covering a broad range of TF families. Each PBM data set contains binding measurements for $\sim 40,000$ short DNA sequences. We utilize the data to build k -mer linear regression models, estimating the model parameters with OLS. The novelty of our approach, compared to previous work [3, 40], is that we can use the estimated regression coefficients together with the estimated covariance matrix to compute not only

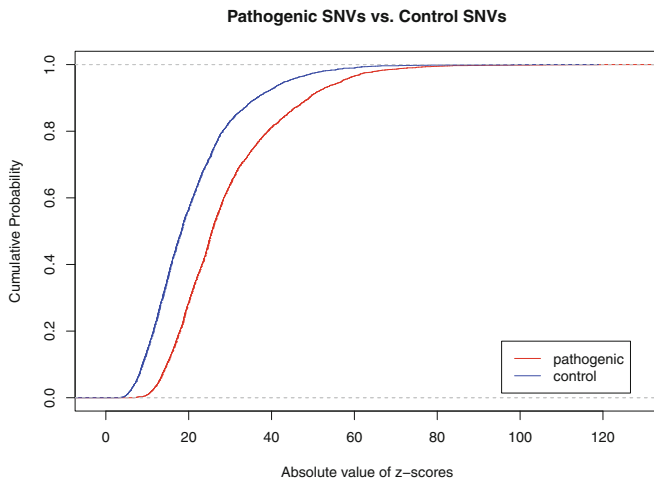


Fig. 4. Comparison of predicted TF binding changes between pathogenic SNVs (red line) and control SNVs (blue line). Overall, pathogenic SNVs have a larger effect on TF binding, as predicted by our OLS 6-mer models.

the change in TF binding due to a mutation (or set of mutations), but also a z -score and a p -value indicating the significance of the change.

Importantly, for any given mutation, the z -scores and p -values obtained for different TFs are directly comparable and can be combined to assess the broad regulatory effects of mutations, as illustrated in Sect. 3.4. In contrast, given that PWM scores are not directly comparable across different models, combining differences in PWM scores for large sets of TFs is not straightforward. As another advantage of OLS k -mer models over PWMs, we note that our k -mer models can be used to assess the effect of mutations over long regions containing multiple mutations and binding sites, without the need to call binding sites according to some score cutoff. As shown in Sect. 3.2 for mutations in an enhancer regulated by Creb1, our OLS model was able to quantitatively capture the effects of DNA mutations over long regions, explaining $\sim 50\%$ of the change in gene expression. Thus, we expect any method that uses PWM models to assess the functional effects of non-coding variants (e.g. [10, 12, 25, 30]) to benefit from using our OLS models instead of PWMs.

We note that individual k -mer features in our models are not independent, so their estimated coefficients ($\hat{\beta}$) should not be interpreted individually. Overall, though, the change in binding score and the corresponding z -scores and p -values, computed as described in Sect. 2.5, can be interpreted directly because they take into account all overlapping k -mers affected by the mutation of interest, and the z -scores and p -values also take into consideration the correlation between features through the estimated variance-covariance matrix. One concern about the z -scores and p -values is their dependence on the normality of the random error, which can be approximately achieved by exploring the transformation of

the raw intensity score. In our study we did not elaborate on finding the optimal transformation, since we have a sufficiently large sample size for the statistical tests to be applicable even in cases of non-normality [19]. The main limitation of our OLS approach is that the number of features cannot exceed the number of observations. In future work we will focus on Bayesian methods that can be applied to higher dimensional data, while at the same time providing posterior distributions that allow us to make inferences about the model parameters.

Acknowledgements. This research was supported in part by awards number P01CA142538 from the National Cancer Institute, and R01GM117106 from the National Institute of General Medical Sciences (to RG). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of Health.

References

1. Adzhubei, I.A., Schmidt, S., Peshkin, L., et al.: A method and server for predicting damaging missense mutations. *Nat. Methods* **7**(4), 248–249 (2010)
2. Andersen, M.C., Engstrom, P.G., Lithwick, S., et al.: In silico detection of sequence variations modifying transcriptional regulation. *PLoS Comput. Biol.* **4**(1), e5 (2008)
3. Annala, M., Laurila, K., Lahdesmaki, H., Nykter, M.: A linear model for transcription factor binding affinity prediction in protein binding microarrays. *PLoS One* **6**(5), e20059 (2011)
4. Auton, A., Brooks, L.D., Durbin, R.M., et al.: A global reference for human genetic variation. *Nature* **526**(7571), 68–74 (2015)
5. Badis, G., Berger, M.F., Philippakis, A.A., et al.: Diversity and complexity in DNA recognition by transcription factors. *Science* **324**(5935), 1720–1723 (2009)
6. Barrera, L.A., Vedenko, A., Kurland, J.V., et al.: Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science* **351**(6280), 1450–1454 (2016)
7. Berger, M.F., Badis, G., Gehrke, A.R., et al.: Variation in homeodomain DNA binding revealed by high-resolution analysis of sequence preferences. *Cell* **133**(7), 1266–1276 (2008)
8. Berger, M.F., Bulyk, M.L.: Universal protein-binding microarrays for the comprehensive characterization of the DNA-binding specificities of transcription factors. *Nat. Protoc.* **4**(3), 393–411 (2009)
9. Berger, M.F., Philippakis, A.A., Qureshi, A.M., et al.: Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. *Nat. Biotechnol.* **24**(11), 1429–1435 (2006)
10. Boyle, A.P., Hong, E.L., Hariharan, M., et al.: Annotation of functional variation in personal genomes using RegulomeDB. *Genome Res.* **22**(9), 1790–1797 (2012)
11. Bulyk, M.L., Johnson, P.L., Church, G.M.: Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. *Nucleic Acids Res.* **30**(5), 1255–1261 (2002)
12. Fu, Y., Liu, Z., Lou, S., et al.: FunSeq2: a framework for prioritizing noncoding regulatory variants in cancer. *Genome Biol.* **15**(10), 480 (2014)
13. Granek, J.A., Clarke, N.D.: Explicit equilibrium modeling of transcription-factor binding and gene regulation. *Genome Biol.* **6**(10), R87 (2005)

14. Jolma, A., Yan, J., Whittington, T., et al.: DNA-binding specificities of human transcription factors. *Cell* **152**(1–2), 327–339 (2013)
15. Kheradpour, P., Ernst, J., Melnikov, A., et al.: Systematic dissection of regulatory motifs in 2000 predicted human enhancers using a massively parallel reporter assay. *Genome Res.* **23**(5), 800–811 (2013)
16. Khurana, E., Fu, Y., Chakravarty, D., et al.: Role of non-coding sequence variants in cancer. *Nat. Rev. Genet.* **17**(2), 93–108 (2016)
17. Kulakovskiy, I.V., Medvedeva, Y.A., Schaefer, U., et al.: HOCOMOCO: a comprehensive collection of human transcription factor binding sites models. *Nucleic Acids Res.* **41**(Database issue), 195–202 (2013)
18. Landrum, M.J., Lee, J.M., Benson, M., et al.: ClinVar: public archive of interpretations of clinically relevant variants. *Nucleic Acids Res.* **44**(D1), D862–868 (2016)
19. Lumley, T., Diehr, P., Emerson, S., Chen, L.: The importance of the normality assumption in large public health data sets. *Annu. Rev. Public Health* **23**, 151–169 (2002)
20. Maerkl, S.J., Quake, S.R.: A systems approach to measuring the binding energy landscapes of transcription factors. *Science* **315**(5809), 233–237 (2007)
21. Mathelier, A., Fornes, O., Arenillas, D.J., et al.: JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.* **44**(D1), D110–115 (2016)
22. Mathelier, A., Zhao, X., Zhang, A.W., et al.: JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Res.* **42**(Database issue), D142–D147 (2014)
23. Matys, V., Kel-Margoulis, O.V., Fricke, E., et al.: TRANSFAC and its module TRANSCOMP: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**(Database issue), D108–D110 (2006)
24. Maurano, M.T., Humbert, R., Rynes, E., et al.: Systematic localization of common disease-associated variation in regulatory DNA. *Science* **337**(6099), 1190–1195 (2012)
25. McLaren, W., Gil, L., Hunt, S.E., et al.: The ensembl variant effect predictor. *Genome Biol.* **17**(1), 122 (2016)
26. McVicker, G., van de Geijn, B., Degner, J.F., et al.: Identification of genetic variants that affect histone modifications in human cells. *Science* **342**(6159), 747–749 (2013)
27. Melnikov, A., Murugan, A., Zhang, X., et al.: Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. *Nat. Biotechnol.* **30**(3), 271–277 (2012)
28. Newburger, D.E., Bulyk, M.L.: UniPROBE: an online database of protein binding microarray data on protein-DNA interactions. *Nucleic Acids Res.* **37**(Database issue), 77–82 (2009)
29. Ng, P.C., Henikoff, S.: SIFT: predicting amino acid changes that affect protein function. *Nucleic Acids Res.* **31**(13), 3812–3814 (2003)
30. Perera, D., Chacon, D., Thoms, J.A., et al.: OncoCis: annotation of cis-regulatory mutations in cancer. *Genome Biol.* **15**(10), 485 (2014)
31. Robasky, K., Bulyk, M.L.: UniPROBE, update 2011: expanded content and search tools in the online database of protein-binding microarray data on protein-DNA interactions. *Nucleic Acids Res.* **39**(Database issue), D124–D128 (2011)
32. Rowan, S., Siggers, T., Lachke, S.A., et al.: Precise temporal control of the eye regulatory gene Pax6 via enhancer-binding site affinity. *Genes Dev.* **24**(10), 980–985 (2010)

33. Siggers, T., Gordan, R.: Protein-DNA binding: complexities and multi-protein codes. *Nucleic Acids Res.* **42**(4), 2099–2111 (2014)
34. Stenson, P.D., Mort, M., Ball, E.V., et al.: The human gene mutation database: building a comprehensive mutation repository for clinical and molecular genetics, diagnostic testing and personalized genomic medicine. *Hum. Genet.* **133**(1), 1–9 (2014)
35. Stormo, G.D.: Modeling the specificity of protein-DNA interactions. *Quant. Biol.* **1**(2), 115–130 (2013)
36. Thomas-Chollier, M., Defrance, M., Medina-Rivera, A., et al.: RSAT 2011: regulatory sequence analysis tools. *Nucleic Acids Res.* **39**(Web Server issue), 86–91 (2011)
37. Tomovic, A., Oakeley, E.J.: Position dependencies in transcription factor binding sites. *Bioinformatics* **23**(8), 933–941 (2007)
38. Udalova, I.A., Mott, R., Field, D., Kwiatkowski, D.: Quantitative prediction of NF-kappa B DNA-protein interactions. *Proc. Natl. Acad. Sci. U.S.A.* **99**(12), 8167–8172 (2002)
39. Ward, L.D., Kellis, M.: Interpreting noncoding genetic variation in complex traits and human disease. *Nat. Biotechnol.* **30**(11), 1095–1106 (2012)
40. Weirauch, M.T., Cote, A., Norel, R., et al.: Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol.* **31**(2), 126–134 (2013)
41. Weirauch, M.T., Yang, A., Albu, M., et al.: Determination and inference of eukaryotic transcription factor sequence specificity. *Cell* **158**(6), 1431–1443 (2014)
42. Zhao, Y., Ruan, S., Pandey, M., Stormo, G.D.: Improved models for transcription factor binding site identification using nonindependent interactions. *Genetics* **191**(3), 781–790 (2012)

aBayesQR: A Bayesian Method for Reconstruction of Viral Populations Characterized by Low Diversity

Soyeon Ahn^(✉) and Haris Vikalo

The University of Texas at Austin, Austin, TX, USA
soyeon.ahn@utexas.edu, hvikalo@ece.utexas.edu

Abstract. RNA viruses replicate with high mutation rates, creating closely related viral populations. The heterogeneous virus populations, referred to as viral quasispecies, rapidly adapt to environmental changes thus adversely affecting efficiency of antiviral drugs and vaccines. Therefore, studying the underlying genetic heterogeneity of viral populations plays a significant role in the development of effective therapeutic treatments. Recent high-throughput sequencing technologies have provided invaluable opportunity for uncovering the structure of quasispecies populations. However, accurate reconstruction of viral quasispecies remains difficult due to limited read-lengths and presence of sequencing errors. The problem is particularly challenging when the strains in a population are highly similar, i.e., the sequences are characterized by low mutual genetic distances, and further exacerbated if some of those strains are relatively rare; this is the setting where state-of-the-art methods struggle. In this paper, we present a novel viral quasispecies reconstruction algorithm, aBayesQR, that employs a maximum-likelihood framework to infer individual sequences in a mixture from high-throughput sequencing data. The search for the most likely quasispecies is conducted on long contigs that our method constructs from the set of short reads via agglomerative hierarchical clustering; operating on contigs rather than short reads enables identification of close strains in a population and provides computational tractability of the Bayesian method. Results on both simulated and real HIV-1 data demonstrate that the proposed algorithm generally outperforms state-of-the-art methods; aBayesQR particularly stands out when reconstructing a set of closely related viral strains (e.g., quasispecies characterized by low diversity).

Keywords: Viral quasispecies reconstruction · Low diversity · Bayesian method

1 Introduction

A number of potentially life-threatening infectious diseases are caused by RNA viruses, including human immunodeficiency virus (HIV), hepatitis C virus (HCV), influenza and Ebola. RNA viruses have a relatively high mutation rate

due to both their error-prone replication process and the lack of sophisticated repair mechanisms [1]. Consequently, they rapidly evolve and exist as a set of non-identical but closely related genetic variants, known as a viral quasispecies. Viral populations can readily adapt to dynamic environments and develop resistance to antiviral drugs and vaccines, which makes the design of effective and long-lasting treatments for RNA viral diseases exceedingly difficult [2]. Determining the structure of viral populations helps the understanding of viral diseases and provides guidance in the development of effective medical therapeutics. Quasispecies spectrum reconstruction (QSR) aims to assemble individual haplotype sequences in a population and estimate their prevalence using sequencing reads generated from a sample containing a set of viral variants. High-throughput next-generation sequencing (NGS) technologies have enabled affordable acquisition of data needed to assemble quasispecies. However, relatively short length of the NGS reads and the presence of errors in sequencing data render the QSR problem difficult. The QSR problem is particularly challenging when the strains in a viral population are highly similar, i.e., the sequences are characterized by low mutual genetic distances, and further exacerbated if some of those strains are relatively rare [3].

Several software tools for solving the QSR problem by analyzing NGS data have been developed in recent years. ShoRAH [4], the earliest publicly available such software, was developed by combining a path cover based approach and probabilistic clustering in [5, 6], respectively, and applied to analysis of HIV data [7]. Read-graph approach was the basis for ViSpA [8], developed as a variant of the network flow method proposed in [9]. [10], proposed a combinatorial method for QSR and the resulting software, QuRe, was provided by [11]. An approach that resulted in the software package PredictHaplo [12] relied on a Dirichlet Process mixture model and was developed specifically targeting HIV population reconstruction; QuasiRecomb [13] is based on a hidden Markov model that explicitly models recombination events. In [14], a benchmarking study that compares the performance of several publicly available quasispecies reconstruction softwares was presented. The study demonstrated that none of the tested methods could reconstruct populations characterized by low pairwise distance between the haplotype sequences. Following this study other softwares, including HaploClique [15], based on max-clique enumeration of a read alignment graph, and VGA [16], a graph-coloring based heuristic method, were developed. Most recently, a reference-assisted *de novo* assembly pipeline, ViQuaS, was proposed in [17]. ViQuaS extends an existing algorithm, QuRe [10], and outperforms various other techniques on a wide range of dataset. However, performance of these more recent methods deteriorates dramatically in the scenarios where the genetic diversity of a population is low [3].

Both [3, 14] have pointed out that the existing methods for viral quasispecies reconstruction struggle in the scenarios where the populations are characterized by low diversity. This, in part, is due to the presence of relatively long genetic regions that are common to pairs of closely related viral sequences; clearly, this makes distinguishing different strains challenging. The problem becomes even

more difficult when the frequency of one (or more) of the close strains is low; in such settings small genetic distances may be confused for sequencing errors and hence remain undetected. Such failures to detect may have serious consequences in antiviral treatment studies since undetected strains cannot be properly targeted for drug and vaccine design. It has been shown that even the viral strains existing at low frequencies can cause a drug treatment failure due to their resistance to the drug [18, 19]. Therefore, complete recovery of the composition of viral populations is of critical importance for effective antiviral therapies.

In this paper, we propose a novel QSR algorithm, aBayesQR (combining agglomerative hierarchical clustering and Bayesian inference), that overcomes limitations of the existing methods and reliably reconstructs quasispecies characterized by low diversity. The algorithm performs reconstruction of a quasispecies from next-generation sequencing (NGS) data in two stages. In the first stage, conflict-free short reads are hierarchically merged and assembled into longer sequences (contigs) which we refer to as super-reads. In the second stage, likelihoods of the probable quasispecies are computed using the assembled super-reads (rather than using the original set of short reads), and the most likely set of viral strains is selected. Note that the super-reads synthesized in the first stage of aBayesQR allow us to distinguish between closely related strains which share long genetic regions as well as reduce the search space and enable computational tractability of the Bayesian inference conducted in the second stage. The second stage of aBayesQR involves sequential pruning of the solution space; in particular, the likely set of partial viral strains comprising n single nucleotide variants (SNVs) is generated by extending previously inferred partial viral strains having $n - 1$ SNVs. The number of sequences in a set (i.e., the size of a viral population) is dynamically updated at each step by evaluating quality of the set of partially reconstructed viral strains, and ultimately precisely inferred at the end of the search process. The relative frequencies of each strain are determined by counting the numbers of reads unambiguously associated with each of the reconstructed strains. Our tests on both simulated and experimental data demonstrate superior performance compared to state-of-the-art methods for quasispecies reconstruction. In particular, it is shown that unlike the competing methods, aBayesQR is capable of detecting and reliably reconstructing viral haplotypes having very small mutual genetic distances.

2 Proposed Method

Our algorithm for inferring spectrum of a viral population consists of the following two steps: (1) constructing super-reads by hierarchically clustering aligned paired-end reads, (2) inferring the most likely quasispecies from the set of super-reads and estimating the frequencies of the strains in the quasispecies.

2.1 Super-Reads Construction via Agglomerative Clustering

In the first stage of aBayesQR, paired-end reads uniquely mapped to a reference genome are grouped into super-reads via agglomerative hierarchical clustering.

This is facilitated by a weighted graph $G = (\mathcal{V}, \mathcal{E})$ which is constructed and recursively updated as the clustering proceeds. In particular, each vertex of G is associated with a cluster collecting reads that originated from a single strain in a quasispecies; we denote the set of reads in the i^{th} cluster (i.e., the cluster associated with the i^{th} vertex) as $V_i = \{v_i^j, j = 1, \dots, |V_i|\}$. Let sr_i denote a consensus sequence (i.e., a super-read) constructed from the reads in V_i . The i^{th} and j^{th} vertex of G are connected by an edge $e_{ij} \in \mathcal{E}$ if all the reads in V_i and V_j (or, equivalently, sr_i and sr_j) are conflict-free and an overlap criterion, specified later in this subsection, is satisfied. The weight w_{ij} of the edge e_{ij} is a measure of similarity between V_i and V_j at each step, the algorithm merges a pair of vertices connected by the edge having the largest weight to form a new vertex and agglomerates the corresponding clusters.

The alleles at homozygous sites, common to all the components of a quasi-species, are not utilized in the reconstruction procedure. Instead, we separate reads having originated from different strains by clustering them using heterogeneous sites with reliable SNV information. An SNV information is considered reliable if the relative abundance of the allele is above a pre-determined threshold, as in [20]; alleles whose abundance is below the threshold are treated as sequencing errors and disregarded in the process of clustering. For convenience, let us denote the set of pre-processed paired-end reads by $R = \{r_i, i = 1, \dots, |R|\}$. The agglomerative clustering is initialized with $|R|$ clusters, one for each read; in other words, we start with $V_1 = r_1, \dots, V_{|R|} = r_{|R|}$, implying that

$|\mathcal{V}| = \sum_{i=1}^{|\mathcal{V}|} |V_i| = |R|$, and proceed by sequentially merging judiciously chosen pairs of vertices (i.e., agglomerating the corresponding clusters). Intuitively, it is meaningful to reduce the number of vertices in the graph by merging those associated with conflict-free consensus sequences that have a large overlap. To formalize this, let $L_i = \{l_1, \dots, l_{|L_i|}\}$ denote an index set of the SNV positions covered by sr_i , let $L_{i \cap j} = \{l_1, \dots, l_{|L_{i \cap j}|}\}$ be the index set of SNV positions covered by both sr_i and sr_j , and let $L_{i \cup j} = \{l_1, \dots, l_{|L_{i \cup j}|}\}$ be the index set of SNV positions covered by either sr_i or sr_j . Then the pairs of vertices (i, j) that we consider as candidates for merging and thus connect by an edge are those satisfying either

$$|L_{i \cap j}| \geq \theta \cdot |L_{i \cup j}| \quad \text{or} \quad |L_{i \cap j}| = \min(|L_i|, |L_j|),$$

where the 2^{nd} condition promotes merger of short super-reads, and the choice of θ is discussed below. To quantify uncertainty inherent to a clustering solution due to existence of non-overlapping positions among the reads in each cluster, we define a position-specific confidence score

$$score_i[l] = \frac{cr_i[l] - cr[l]}{1 - cr[l]}$$

where l denotes the position, $cr[\cdot]$ is the overall coverage rate, and $cr_i[\cdot]$ denotes cluster-specific coverage rate for V_i (i.e., $cr_i[l]$ is the fraction of reads in $V_i = \{v_i^j, j = 1, \dots, |V_i|\}$ covering position l). On the one hand, this score is penalized

at a site where the fraction of cluster members (short reads) covering the site is low; the score is negative if the cluster-specific coverage rate is below the global coverage rate which implies uncertainty of the clustering decision. On the other hand, positive scores indicate high confidence in the decision to group the reads into the same cluster. Note that the highest possible score of 1 at position l is achieved when all the reads in a cluster cover the l^{th} position. Using the confidence scores, we define the weight w_{ij} assigned to an edge e_{ij} to quantify similarity between V_i and V_j as

$$w_{ij} = \frac{1}{|L_{i \cup j}|} \sum_{l \in L_{i \cup j}} \text{score}_{e_{ij}}[l].$$

Given the weights w_{ij} , we can now specify the clustering procedure. In each step, the pair of vertices connected by the edge with maximum weight is merged; the newly constructed vertex inherits edges from the merged vertices and the weights on those edges are re-evaluated. A new (longer) consensus sequence is constructed by combining the two super-reads associated with the merged vertices; recall that there are no conflicts between the super-reads being merged. If after such an update step no edges connect the new vertex with the rest of the graph (because no inherited edges satisfy the connectivity condition), θ is decreased and the above process is repeated. We initially set θ to 0.9 and gradually decrease it by 0.1 while $\theta > 0$. The above procedure is repeated until no pairs of vertices satisfy the connectivity condition. By that point, a set of long consensus sequences (the final super-reads) has been formed from the clusters of reads associated with the nodes of the final graph. While the complexity of agglomerative clustering is, in general, $O(N^3)$ where N denotes the input data size [21], it has been shown that its time complexity can be reduced to $O(N^2)$ with accuracy equal to that of the brute-force method by using the partial maximum array technique [22]. We exploit this to efficiently construct super-reads. The algorithm for super-read construction is formalized as Algorithm 1.

Algorithm 1. Agglomerative clustering for super-reads construction

Input: Set of reads aligned to the reference genome

Output: Set of super-reads and the corresponding confidence scores

for $\theta > 0$ **do**

Build a weighted graph $G = (\mathcal{V}, \mathcal{E})$

while $E \neq \emptyset$ **do**

Merge two clusters connected with the largest weight

Update $G = (\mathcal{V}, \mathcal{E})$ and weights using partial maximum array

end while

$\theta = \theta - 0.1$

end for

2.2 ML Reconstruction of Quasispecies from Super-Reads

Here we describe how to reconstruct the most likely set of strains in a viral quasispecies using super-reads from Sect. 2.1 and their confidence scores. While in principle the method outlined in this section could be applied directly to the short reads provided by a sequencing platform, such an approach would in general not only be computationally prohibitive due to a very large number of short reads but also limit the ability of the algorithm to distinguish strains with small mutual genetic distances due to having long conserved regions. Relying on a relatively small number of long super-reads constructed from short reads circumvents both of these problems and makes the reconstruction more accurate and practically feasible. Note that sequencing errors may undesirably prevent clusters of reads from being merged with other clusters due to a violation of conflict-free requirement; consequently, a set of short reads in a small cluster is likely to have a disproportionate amount of sequencing errors. For this reason, we ignore clusters with very small memberships (in particular, those containing fewer than $0.001 \cdot |R|$ reads), which limits the detection of strains to those constituting more than 0.1% of the quasispecies.

Let $\mathcal{C} = \{C_m, m = 1, \dots, M\}$ denote the collection of clusters that remain after deleting clusters having only few reads; moreover, for convenience let us re-label the reads in C_m as c_m^j , i.e., $C_m = \{c_m^j, j = 1, \dots, |C_m|\}$ where $c_m^j \in R$. We organize the super-reads obtained by Algorithm 1 in Sect. 2.1 into the rows of an $M \times N$ matrix $\mathbf{S} = \{s_{mn}, m = 1, \dots, M, n = 1, \dots, N\}$ with entries $s_{mn} \in \{A, C, G, T, -\}$ where $-$ denotes a site not covered by a super-read and N denotes the total number of SNV sites in the strains of a quasispecies. A nucleotides in the (m, n) position of \mathbf{S} is assigned confidence $score_m[n]$ defined in Sect. 2.1; the scores for the entire matrix are normalized so that they fall between 0 and 1 in order to use them in our Bayesian approach to assembly. Let ε_{mn} be the probability that s_{mn} was estimated erroneously due to either a sequencing error in reads on the n^{th} SNV position or the uncertainty induced by reads not covering the n^{th} SNV position. Note that negative scores indicates low confidence resulting from insufficient cluster-specific coverage rate while positive scores imply relatively confident information. In order to map $score_m[n] \in (-\infty, 1]$ to the set $[0, 1]$, we set $\varepsilon_{mn} = 1 - e^{score_m[n]}$ for $score_m[n] < \ln(1 - \epsilon)$, where ϵ denotes the error rate of a sequencing platform. Otherwise, we set $\varepsilon_{mn} = \epsilon$.

Let $Q = \{q_k, k = 1, \dots, K\}$ denote the set of K strains of a viral quasispecies. The goal in the second stage of our method is to determine Q from the super-reads matrix \mathbf{S} using a probabilistic framework. An exhaustive search over the entire solution space is computationally intractable even for small \mathbf{S} ; instead, we reconstruct the set of K viral strains sequentially, extending partially estimated strains one SNV position at each step. Since maintaining and extending all possible partial strains inevitably increases their number exponentially, unlikely sets of candidate strains are pruned in each step. Each step consists of three basic parts: (a) extension of the partially reconstructed strains, (b) selection of probable sets comprising K strains chosen among those generated in step (a), and (c) evaluation

of the quality of the selected sets of strains and an update of K . The sequential Bayesian inference procedure in step t is illustrated in Fig. S1 in Appendix A.

Extending Partially Reconstructed Strains. Let $F_{1:t-1} = \{f_{1:t-1}^i, i = 1, \dots, |F_{1:t-1}|\}$ be the collection of partially reconstructed strains covering the first $t - 1$ SNV sites and let $B_t = \{b_t^j, j = 1, \dots, |B_t|\}$ be the lists of distinct bases in the t^{th} column of \mathbf{S} , where $b_t^j \in \{\text{A, C, G, T}\}$ and $2 \leq |B_t| \leq 4$. Then, all the possible extensions of $f_{1:t-1}^i$ to the SNV site t can be enumerated as $\{[f_{1:t-1}^i, b_t^1], \dots, [f_{1:t-1}^i, b_t^{|B_t|}]\}$. Let $S_{1:t-1}^{i_{c'}} = \{s_{1:t-1}^{i_{c'}}, c' = 1, \dots, |S_{1:t-1}^{i_{c'}}|\}$ be the collection of super-reads covering some of the first t SNV sites which are consistent with $f_{1:t-1}^i$ (ignoring “-” in $s_{1:t-1}^{i_{c'}}$) where $\{i_{c'}\}$ denote indices of rows of \mathbf{S} that are placed in $S_{1:t-1}^{i_{c'}}$, and let $S_t^{i_c} = \{s_t^{i_c}, c = 1, \dots, |S_t^{i_c}|\}$ denote the collection of nucleotides ($s_t^{i_c} \in \{\text{A, C, G, T}\}$, not “-”) observed at the t^{th} SNV site of the super-reads in $S_{1:t-1}^{i_c}$ where $\{i_c\}$ denote the indices of rows in \mathbf{S} that contribute to $S_t^{i_c}$. Given $S_{1:t-1}^{i_c}, S_t^{i_c}$ and $f_{1:t-1}^i$, the probability of b_t^j being the true extension of $f_{1:t-1}^i$ is given by

$$P(S_t^{i_c} | b_t^j, S_{1:t-1}^{i_c}, f_{1:t-1}^i) = \prod_{c=1}^{|S_t^{i_c}|} P(s_t^{i_c} | b_t^j),$$

$$P(s_t^{i_c} | b_t^j) = \begin{cases} 1 - \varepsilon_{i_c t}, & \text{if } b_t^j = s_t^{i_c}, \\ \frac{\varepsilon_{i_c t}}{|B_t|}, & \text{otherwise.} \end{cases}$$

We extend $f_{1:t-1}^i$ to $[f_{1:t-1}^i, b_t^j] \in F_{1:t-1,t}$ by appending the $b_t^j \in B_t$ which satisfies

$$\frac{P(S_t^{i_c} | b_t^j, S_{1:t-1}^{i_c}, f_{1:t-1}^i) \frac{1}{|S_t^{i_c}|}}{\sum_{B_t} P(S_t^{i_c} | b_t^j, S_{1:t-1}^{i_c}, f_{1:t-1}^i) \frac{1}{|S_t^{i_c}|}} \geq \delta_0, \text{ where the exponent ensures proper normalization}$$

and is needed since the number of super-reads, $|S_{1:t-1}^{i_c}|$, varies for each $\{f_{1:t-1}^i, i = 1, \dots, |F_{1:t-1}|\}$. For $f_{1:t-1}^i$ which has no matched super-reads, i.e., $|S_{1:t-1}^{i_c}| = 0$, we keep all of $|B_t|$ possible extensions of $f_{1:t-1}^i$. By collecting probable extensions for each $f_{1:t-1}^i \in F_{1:t-1}$, we obtain the set of partial strains stretching over the first t SNV sites, $F_{1:t-1,t}$. This procedure is formalized as function *ExtendFrag* in Appendix A.

Inferring Likely Sets of K Partial Strains. Having generated the probable partial strains $F_{1:t-1,t}$, we denote the set of all its possible subsets of K strains (i.e., the quaspecies population candidates) as $\mathcal{Q}_{1:t-1,t} = \{Q_{1:t-1,t}^i, i = 1, \dots, \binom{|F_{1:t-1,t}|}{K}\}$ where $Q_{1:t-1,t}^i = \{q_{kn}^i, k = 1, \dots, K, n = 1, \dots, t\}$ and $q_{kn}^i \in F_{1:t-1,t}$. The log-likelihoods of $Q_{1:t-1,t}^i$ can be expressed as

$$\ln P(\mathbf{S} | Q_{1:t-1,t}^i) = \sum_{m=1}^M \ln P(s_{m\cdot} | Q_{1:t-1,t}^i),$$

$$P(s_{m\cdot} | Q_{1:t-1,t}^i) = \frac{1}{K} \left(\sum_{k=1}^K \left(\prod_{n=1}^t P(s_{mn} | q_{kn}^i) \right) \right),$$

where s_m denotes the m^{th} row vector of the matrix of super-reads \mathbf{S} and

$$P(s_{mn}|q_{kn}^i) = \begin{cases} 1 - \varepsilon_{mn}, & \text{if } q_{kn}^i = s_{mn}, \\ \frac{\varepsilon_{mn}}{|B_n|}, & \text{if } q_{kn}^i \neq s_{mn} \text{ for } s_{mn} \neq -. \end{cases}$$

Let $Q_{1:t}^{max} = \max_{Q_{1:t-1,t}^i \in \mathcal{Q}_{1:t-1,t}} P(\mathbf{S}|Q_{1:t-1,t}^i)$. Among the $\binom{|F_{1:t-1,t}|}{K}$ sets in $\mathcal{Q}_{1:t-1,t}$, we keep only those that satisfy $P(\mathbf{S}|Q_{1:t-1,t}^i) > \delta_1 \cdot Q_{1:t}^{max}$ while the others are discarded; let us denote the collection of candidate sets that pass this test as $\mathcal{Q}_{1:t}$. For practical feasibility of the scheme, the collection of partially reconstructed strains $F_{1:t-1,t}$ is trimmed by excluding from it all the strains that are not part of at least one of the sets in $\mathcal{Q}_{1:t}$; we denote the resulting collection of partial strains by $F_{1:t} \in F_{1:t-1,t}$ and use it when extending the strains onto the $t + 1$ SNV site. The described procedure is formalized as function *InferQuasi* in Appendix A.

Determining the Number of Strains K in a Quasispecies. In this step, we assess appropriateness of K used in the inference of $\mathcal{Q}_{1:t}$ and update it if necessary. To this end, we rely on the minimum error correction (MEC) score which has previously been broadly used as a criterion in the design of methods for haplotype assembly [23, 24]. In the context of polyploid haplotype assembly, the MEC score is defined as the smallest number of nucleotides that needs to be changed in data (i.e., in observed reads) so that the corrected reads are consistent with having originated from K haplotypes. Let $HD_t(\cdot, \cdot)$ denote the Hamming distance between two sequences counted over the observed nucleotides in the first t SNV positions.¹ Then the MEC score of the most likely set $Q_{1:t}^{max}$ of K viral strains evaluated on the first t SNVs is

$$MEC_t(K) = \sum_{m=1}^M \min_{k \in \{1, \dots, K\}} \sum_{j=1}^{|C_m|} HD_t(c_m^j, q_k^{max}),$$

where q_k^{max} is the k^{th} row vector of $Q_{1:t}^{max}$. Let N_t be the total number of nucleotides observed in the first t SNV positions of all the reads of the dataset. Note that the smaller the MEC scores, the higher the accuracy of a clustering. If $MEC_t(K)/N_t < 2\epsilon$, we use the same value K in the next step where the likely set of viral strains stretching over the first $t + 1$ SNV positions is inferred. Otherwise, we increase K by 1, repeat the estimation of $\mathcal{Q}_{1:t}$, and evaluate the improvement rate of MEC score as

$$MECimpr(K) = \frac{MEC_t(K) - MEC_t(K + 1)}{MEC_t(K)}.$$

The reason for selecting K based on the MEC improvement rate (*MECimpr*) is that the MEC score drops significantly once K matches the actual number of

¹ If either of the two sequences has a gap “-” in a position, that position is ignored in the computation of the aforementioned Hamming distance.

clusters; our scheme attempts to detect that change in order to infer population size. If $MECimpr(K) > \eta$, where η denotes a pre-specified threshold, the number of species is updated as $K \leftarrow \min\{K + n, |F_{1:t-1,t}|\}$ where n is the smallest integer number such that $MECimpr(K + n) < \eta$. If $MECimpr(K) < \eta$, we update the number of species as $K \leftarrow \max\{K - n, 2\}$ where n is the smallest integer such that $MECimpr(K - n) \geq \eta$. The choice of threshold η is discussed in the Appendix B. The updated value of K is used for the inference of $\mathcal{Q}_{1:t+1}$. Note that the probable set of viral strains, $\mathcal{Q}_{1:t}$, is stored for each K to avoid performing redundant $MECimpr(\cdot)$ calculations.

Once we obtain the most likely set of K viral sequences covering N SNVs, $Q_{1:N}^{max}$, the full-length K quasispecies strains are reconstructed by inserting the consensus nucleotides observed in R into the non-SNV sites. We estimate relative frequencies p_k , $1 \leq k \leq K$, of quasispecies strains based on the Hamming distance between super-reads and the reconstructed sequences. In particular, for each super-read sr_i we determine the nearest assembled strain q_j where $j = \arg \min_{k \in \{1, \dots, K\}} HD(sr_i, q_k)$ and the number of reads involved in constructing the super-read sr_i is counted towards p_j . The entire scheme proposed in this subsection is summarized as Algorithm 2.

Algorithm 2. Sequential Bayesian Inference for quasispecies reconstruction

Input: Set of super-reads and the corresponding confidence scores

Output: Set of K strains of a viral quasispecies

Initial $K \leftarrow 2$, $F_{1:1} \leftarrow B_1$

for $t \in \{2, \dots, N\}$ **do**

$F_{1:t-1,t} = \mathbf{ExtendFrag}(F_{1:t-1}, t, \delta_0)$

$\mathcal{Q}_{1:t} = \mathbf{InferQuasi}(F_{1:t-1,t}, K, \delta_1)$

$K^* \leftarrow K$, $\mathcal{Q}_{1:t}^* \leftarrow \mathcal{Q}_{1:t}$

if $MEC_t(K)/N_t \geq 2\epsilon$ and $K < |F_{1:t-1,t}|$ **do**

$\mathcal{Q}_{1:t} = \mathbf{InferQuasi}(F_{1:t-1,t}, K+1, \delta_1)$

if $MECimpr(K) < \eta$ **do**

while $MECimpr(K) < \eta$ and $K > 2$

$\mathcal{Q}_{1:t}^* \leftarrow \mathcal{Q}_{1:t}$, $K^* \leftarrow K$, $K \leftarrow K - 1$

$\mathcal{Q}_{1:t} = \mathbf{InferQuasi}(F_{1:t-1,t}, K, \delta_1)$

end while

else do

while $MECimpr(K) \geq \eta$ and $K < |F_{1:t-1,t}|$

$\mathcal{Q}_{1:t}^* \leftarrow \mathcal{Q}_{1:t}$, $K^* \leftarrow K$

$\mathcal{Q}_{1:t} = \mathbf{InferQuasi}(F_{1:t-1,t}, K+1, \delta_1)$

end while

end if

end if

$K \leftarrow K^*$, $\mathcal{Q}_{1:t} \leftarrow \mathcal{Q}_{1:t}^*$

Get $F_{1:t}$ by pruning $F_{1:t-1,t}$ based on $\mathcal{Q}_{1:t}$

end for

Reconstruct full-length quasispecies Q from $Q_{1:N}^{max} \in \mathcal{Q}_{1:t}$ and R

Estimate frequencies of each strain $q_k \in Q$ based on $HD(sr_i, q_k)$ and $|C_i|$

3 Results and Discussion

3.1 Performance Comparison on Simulated Data

To evaluate performance of the proposed method for quasispecies reconstruction, we use metrics *Recall*, *Precision*, *Predicted Proportion*, and *Reconstruction Rate*. *Recall* is defined as the ratio of the number of correctly reconstructed strains to the total number of true strains in the quasispecies, i.e., $Recall = \frac{TP}{TP+FN}$, while *Precision* is defined as the fraction of correctly reconstructed strains among all the assembled sequences, i.e., $Precision = \frac{TP}{TP+FP}$. Noting that *Precision* usually reports high scores when the number of strains is underestimated while penalizing overestimation of the population size, we also report the ratio of the number of reconstructed sequences to the true population size, *Predicted Proportion*. The closer *Predicted Proportion* to 1, the more accurate the number of reconstructed strains. Moreover, to assess the degree of reconstruction accuracy, we define $Reconstruction\ Rate = \frac{1}{K} \sum_{k=1}^K \left(1 - \frac{HD(q_k, \hat{q}_k)}{G} \right)$, where G is the length of a genome, K is the number of strains in a quasispecies and q_k and \hat{q}_k denote the k^{th} true strain and its nearest sequence among the K estimated ones, respectively. To assess the accuracy of estimated frequencies, we use Jensen-Shannon divergence (JSD) which quantifies similarity between two distributions. Given a true distribution P and its approximation Q , the Kullback-Leibler (KL) divergence $D(P||Q) = \sum_{i=1}^n P(i) \log \frac{P(i)}{Q(i)}$ is undefined when $Q(i) = 0$. JSD, a symmetrized and smoothed version of the KL divergence, circumvents this problem by defining similarity of P and Q as $JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M)$, where M is defined as $M = \frac{1}{2}(P + Q)$.

We compare our algorithm with publicly available ShoRAH [4], PredictHaplo [12], and ViQuaS [17]. Since ViQuaS is an extension of the algorithm in [10, 11], and was shown to have superior performance compared to its predecessor, we omit the comparison with the software QuRe in [10, 11]. It is worth pointing out that for the synthetic data sets we study, ShoRAH could not reconstruct strains in the regions where the simulated sequencing coverage is relatively low compared to the average, resulting in reconstruction of strains that are shorter than the true length G . To facilitate a fair comparison with ShoRAH, we aligned its reconstructed strains to the reference genome and completed missing sites with bases from the reference. ViQuaS, on the other hand, tends to reconstruct many more strains than actually present; thus we followed ViQuaS's authors recommendation and retained only those having frequencies greater than f_{min} when calculating *Precision*. Finally, not all of the synthetic data sets could be processed with PredictHaplo, preventing us from reporting its performance in some of the scenarios.

We generated synthetic datasets by emulating high-throughput sequencing of a viral population consisting of a number of closely related viral genomes having length of 1300bp; this particular length was chosen to coincide with the longest region of the HIV *pol* gene. Quasispecies sequences are generated

Table 1. Performance comparison of different methods for varied diversities (div) on simulated data. Performance comparison of aBayesQR, ShoRAH, ViQuaS and PredictHaplo in terms of *Recall*, *Precision*, *Predicted Proportion (PredProp)*, *Reconstruction Rate (ReconRate)* and *JSD* on the simulated data with $err = 0.1\%$ and $cov = 500 \times$ vs. div for a mixture of 5 and 10 viral strains. Averaged PredictHaplo results are reported if it provides answers for more than 50% of data sets. Boldface values indicate the best performance for each $div(\%)$.

		5 strains					10 strains				
	$div(\%)$	1	2	3	4	5	1	2	3	4	5
Recall	aBayesQR	0.7080	0.7120	0.6840	0.6560	0.6320	0.5810	0.6380	0.6080	0.5860	0.5550
	ShoRAH	0.1920	0.1600	0.1300	0.1060	0.0780	0.0150	0.0380	0.0740	0.0640	0.0930
	ViQuaS	0.3700	0.5240	0.6040	0.6360	0.5960	0.0980	0.1700	0.3730	0.4720	0.5050
	PredictHaplo	-	-	-	0.6918	0.6808	-	-	0.1021	0.1550	0.2010
Precision	aBayesQR	0.7113	0.7130	0.6826	0.6447	0.6319	0.6210	0.6881	0.6610	0.6373	0.6140
	ShoRAH	0.1062	0.1418	0.1240	0.1078	0.0790	0.0050	0.0170	0.0498	0.0506	0.0824
	ViQuaS	0.1960	0.3206	0.4559	0.4982	0.5298	0.0485	0.1079	0.2973	0.4690	0.5596
	PredictHaplo	-	-	-	0.9373	0.8822	-	-	0.4509	0.6000	0.6833
PredProp	aBayesQR	1.0180	1.0120	1.0120	1.0360	1.0140	0.9680	0.9440	0.9240	0.9240	0.9100
	ShoRAH	1.9660	1.2200	1.0780	1.0000	1.0180	3.2000	2.9100	1.6710	1.3520	1.1860
	ViQuaS	2.1100	1.7220	1.4080	1.3340	1.2180	2.0860	1.8580	1.5450	1.2320	1.0730
	PredictHaplo	-	-	-	0.7388	0.7737	-	-	0.1947	0.2430	0.2890
ReconRate	aBayesQR	0.9990	0.9982	0.9971	0.9961	0.9953	0.9975	0.9967	0.9952	0.9942	0.9924
	ShoRAH	0.9948	0.9903	0.9891	0.9851	0.9827	0.9941	0.9900	0.9899	0.9897	0.9911
	ViQuaS	0.9963	0.9949	0.9917	0.9936	0.9897	0.9944	0.9910	0.9899	0.9881	0.9858
	PredictHaplo	-	-	-	0.9906	0.9896	-	-	0.9850	0.9797	0.9747
JSD	aBayesQR	0.0022	0.0008	0.0008	0.0014	0.0008	0.0043	0.0026	0.0023	0.0023	0.0025
	ShoRAH	0.0762	0.0174	0.0047	0.0009	0.0012	0.1390	0.1110	0.0422	0.0238	0.0109
	ViQuaS	0.0651	0.0255	0.0222	0.0097	0.0180	0.0993	0.0747	0.0495	0.0469	0.0454
	PredictHaplo	-	-	-	0.1020	0.1036	-	-	0.1971	0.1636	0.1312

by introducing independent mutations at uniformly random locations along the length of a randomly generated reference genome so as to obtain a predefined level of diversity ($div\%$), i.e., a predefined average Hamming distance between quasispecies strains. Simulating Illumina's MiSeq data, $2 \times 250\text{bp}$ -long paired-end reads are sampled uniformly from each viral strain with a mean coverage of $cov \times$ per strain. Inserts of the paired-end reads are on average 150bp long with standard deviation of 30. In our benchmarking tests, we focus on exploring the effects of diversity ($div\%$) on the accuracy of the quasispecies reconstruction. Two sets of viral populations are considered: (1) a mix of 5 viral strains with abundance levels 50%, 30%, 15%, 4% and 1%; and (2) a mix of 10 strains with abundance levels 36%, 24%, 16%, 8%, 5.5%, 4%, 3%, 2%, 1% and 0.5%. Note that the abundances are chosen to approximately follow geometric distribution and that the populations include low abundant strains. For each combination of the parameters, 100 data sets were generated and the reported results were obtained by averaging over those data instances. For PredictHaplo, which did not produce results in each instance, the averaged results are reported if more than 50 instances were successfully processed.

In all of the following experiments, potential SNVs are called if their abundance is higher than 1%, which is set relatively high to avoid false positives (FPs); FPs prevent reads to be merged with existing clusters in Sect. 2.1. We execute the function *ExtendFrag* with parameter $\delta_0 = 0.1$. Parameter δ_1 in function *InferQuasi* is initially set to 0.001, but adaptively increases if the number of combinations of partially reconstructed strains exceeds 10000; this is done to limit the number of likelihood calculations performed in each run of *InferQuasi*. The recommended value of η , a threshold used to determine population size K based on *MECimpr*(\cdot), is discussed in Appendix B.

We compare performances of aBayesQR, ShoRAH, ViQuaS and PredictHap when applied to the reconstruction of a quasispecies spectrum with diversity levels varying between 1% and 5% (i.e., $div \in \{1\%, 2\%, 3\%, 4\%, 5\%\}$). To test the ability of different methods to reconstruct quasispecies with low diversity, we assume low sequencing error rate of $err = 0.1\%$ (median mismatch error rates for 454 Life Sciences and Illumina platforms are 0.1% and 0.12%, respectively [25]). Coverage per strain $cov\times$ is set to $500\times$, implying total coverage of $2500\times$ and $5000\times$ for the 5-strain and 10-strain population, respectively; strains having frequencies 0.23% or higher in the 5-strain case and those with frequencies 0.46% or higher in the 10-strain case are covered with probability 0.99 [5].

Table 1 demonstrates that the proposed aBayesQR algorithm outperforms existing schemes. In terms of *Recall* and *Precision*, aBayesQR exhibits exceptionally good performance compared to competing methods when reconstructing quasispecies strains with diversity $div < 4\%$. The performance of ViQuaS deteriorates at low diversities in terms of most of the criteria (i.e., *Recall*, *Precision*, *Predicted Proportion* and *JSD*). PredictHaplo could not perform reconstruction in most of the low diversity instances yet it overall achieves the highest *Precision* because it typically underestimates the number of strains as shown by *Predicted Proportion* (e.g., estimating only 2–3 out of 10 strains), which is in agreement with the results reported by a previous study [14]. Among all methods, ShoRAH has the lowest performance in terms of *Recall* and *Precision*. As indicated by *Predicted Proportion*, aBayesQR is the most accurate method in terms of estimating the population size although it often misses a strain with the lowest frequency when applied to reconstruction of a quasispecies consisting of 10 strains. ViQuaS and ShoRAH typically overestimates the number of strains especially at low diversity levels. aBayesQR is the best method in terms of *Reconstruction Rate* at all levels of diversity. In terms of frequency estimation, aBayesQR overall outperforms all the other methods whereas PredictHaplo shows the highest *JSD* due to its drawback of underestimating the number of strains. Note that both ViQuaS and ShoRAH exhibit significantly increased (i.e., deteriorated) *JSD* at low diversity levels. This fact, along with the low *Recall* and *Precision* scores they have in low diversity settings, indicates that state-of-the-art methods experience major difficulties when attempting to reconstruct viral quasispecies in those settings, as also observed in [5, 14, 17].

We further study the effects that sequencing error rate ($err\%$) and coverage per strain ($cov\times$) have on the performance of the algorithms. Those results are

reported in Table S2 and S3 in the Appendix C, demonstrating superiority of aBayesQR as compared to the competing methods. The runtimes of the tested algorithms are shown in Table S4 in the Appendix C.

3.2 Performance Comparison on Real HIV Data

To further test the performance of our proposed method, we employ it for the analysis of the HIV 5-virus-mix dataset published in [20]. Specifically, we apply our algorithm to reconstruct an *in vitro* generated quasispecies population consisting of 5 known HIV-1 strains: HIV-1_{HXB2}, HIV-1_{89.6}, HIV-1_{JR-CSF}, HIV-1_{NL4-3} and HIV-1_{YU2}. Compared to the simulated data set, relative frequencies of the 5 HIV-1 strains are more evenly distributed (about 10–30%) and the pairwise distances between strains are higher (2.61–8.45%) [20]. We use the 2×250 bp-long paired-end reads provided by Illumina’s MiSeq Benchtop Sequencer. The reads are aligned to the HIV-1_{HXB2} reference genome; the reads shorter than 150nt and those having bases with quality scores less than a PHRED threshold of 60 are discarded. We compare the performance of our method applied to gene-wise quasispecies reconstruction of the above described HIV data with that of the competing techniques. Since the current version of ViQuaS software does not support specifying genomic regions, we could not use it in this experiment. When running aBayesQR, we set the parameter η to 0.09 (the setting recommended in Appendix B). Other parameters are set to the same values as the ones used in Sect. 3.1.

We evaluate and report the *Predicted Proportion* (i.e., the fraction of correctly estimated strains as defined in Sect. 3.1) and *Reconstruction Rate* in Table 2. On this real HIV-1 data set which (as pointed above) has different properties than the simulated data set in Sect. 3.1, aBayesQR is the most accurate among the considered methods in terms of *Predicted Proportion*. PredictHaplo underestimates the population size and reconstructs three or four strains in the 8 considered genes and ShoRAH greatly overestimates the population size for all 13 genes of the HIV-1 data set (e.g., it reconstructs 119 strains in gp120), which is consistent with our simulation results as well as with the results in [14]. aBayesQR and PredictHaplo are tied for the number of genes where all the strains are perfectly reconstructed (5 each); for the remaining genes, PredictHaplo provides a larger number of perfectly reconstructed strains. However, it is worth pointing out that PredictHaplo, designed for identification of HIV haplotypes, missed at least one strain in each of the remaining 8 genes while aBayesQR reconstructed most of the strains on all but two genes, gp120 and gp41. ShoRAH did not perfectly reconstruct any of the 13 genes, which is consistent with the simulation results. Moreover, overestimating the number of strains negatively affects the accuracy of ShoRAH’s frequency estimation; for instance, the sum of frequencies corresponding to the most abundant 5 strains does not exceed 50% in 9 out 13 genes (71% is the largest such sum, on *vpu*) (see Table S5 in the Appendix C).

To complement the gene-wise quasispecies reconstruction study with that of a global reconstruction, we consider the HIV-1 *gap-pol* region spanning 4307bp.

Table 2. Performance comparisons on a real HIV-1 5-virus-mix data set. *Predicted Proportion* (PredProp) and *Reconstruction Rate* (RR) for aBayesQR, ShoRAH and PredictHaplo applied to reconstruction of HIV-1_{HXB2}, HIV-1_{S9.6}, HIV-1_{JR-CSF}, HIV-1_{NL4-3} and HIV-1_{YU2} for all 13 genes of the HIV-1 dataset. (note: RR are expressed in percentages.) Boldface values indicate the genes where all the strains are perfectly reconstructed. The inferred frequencies are shown in Table S5 in Appendix C.

		p17	p24	p2-p6	PR	RT	RNase	int	vif	vpr	vpu	gp120	gp41	nef
aBayesQR	PredProp	1	1	1	1	1	1	1	1	1.2	1	0.8	0.8	1.2
	RR _{HXB2}	100	99.4	100	100	98.5	100	99.9	100	100	99.6	98	0	95.8
	RR _{S9.6}	100	98.7	100	100	98.6	100	100	100	100	92	96.5	98.9	95.5
	RR _{JR-CSF}	100	99.6	100	100	99	100	100	100	100	98.8	97.7	99.1	98.2
	RR _{NL4-3}	100	100	100	100	98.9	100	100	99.8	100	100	96.3	98.8	100
	RR _{YU2}	100	99.7	100	100	99.2	100	99.5	99.7	100	100	0	98.6	99.2
ShoRAH	PredProp	13	16.4	13.8	8.8	21.8	11.8	13.6	12.8	7.8	4	23.8	19.8	17.4
	RR _{HXB2}	100	96.7	100	100	98.2	100	97.5	100	100	100	97.7	98.4	98.2
	RR _{S9.6}	100	99.7	100	100	98.6	100	98.9	99.8	100	93.6	96.1	98.6	98.9
	RR _{JR-CSF}	100	100	100	100	99.8	96.4	99	100	100	98	96.9	96.3	94.7
	RR _{NL4-3}	100	99.1	97.3	100	98.9	99.2	99.3	99.3	100	100	96.1	98.5	98.6
	RR _{YU2}	94.2	99	100	98.3	98	94.5	98.6	95	93.2	90.8	97	95.4	97.9
PredictHaplo	PredProp	1	0.6	1	1	1	0.8	0.8	0.8	1	0.8	0.8	0.8	0.8
	RR _{HXB2}	100	0	100	100	100	98.9	100	100	100	93.17	0	0	0
	RR _{S9.6}	100	100	100	100	100	100	99.8	100	100	0	97.8	100	98.87
	RR _{JR-CSF}	100	100	100	100	100	100	100	100	100	100	99.7	100	100
	RR _{NL4-3}	100	99.1	100	100	100	100	100	100	100	100	100	100	100
	RR _{YU2}	100	0	100	100	100	0	0	0	100	100	98.6	100	100

To efficiently process 355241 paired-end reads that remain after applying a quality filter, we organize the region into a sequence of windows of length 400bp where the consecutive windows overlap by 150bp and run aBayesQR on those windows. The entire region is assembled by connecting strains in the consecutive windows while testing consistency in the overlapping intervals. The number of strains retrieved in the global reconstruction is decided by majority voting of the number of strains obtained in each window. The frequencies are estimated by counting reads nearest (in terms of Hamming distance) to each of the reconstructed strains. Following this procedure, both aBayesQR and PredictHaplo could reconstruct all 5 HIV-1 strains in the *gap-pol* region correctly, i.e., they both achieved *Reconstruction rate* of 100 for all 5 strains and *Predicted Proportion* of 1. The frequencies estimated by aBayesQR are 15.21%, 19.34%, 25.56%, 27.61% and 12.27% while those estimated by PredictHap are 13.21%, 13.56%, 25.67%, 19.69% and 27.86%. ShoRAH highly overestimated the number of strains and reported *Predicted Proportion* of 41.8; its five most abundant strains estimated are reported to have frequencies 8.51%, 5.04%, 3.41%, 3.24% and 3.09%.

4 Conclusions

In this paper, we presented a novel maximum-likelihood based approximate algorithm for reconstructing viral quasispecies from high-throughput sequencing data. aBayesQR assembles paired-end short reads into longer fragments based on similarity of the read overlaps and the uncertainty level of non-overlapping regions. The probable sets of partially reconstructed strains are inductively searched and a subset of those strains is extended to efficiently deduce the most likely set of strains in a quasispecies. Detection of the population size is embedded into the algorithm and is empirically shown to be very accurate; the number of strains is dynamically adjusted based on the reliability of the partially assembled quasispecies in each extension step. Performance of the developed method is tested on both synthetic datasets and a real HIV-1 dataset. In both settings, the new algorithm outperforms existing techniques in terms of accuracy of the quasispecies size estimation, perfect reconstruction of strains, proportion of correct bases in each reconstructed strain and the estimation of their abundance. A particularly high accuracy is observed in estimating the population size (i.e., the number of strains) and their relative abundance. Tests on synthetic datasets demonstrates that aBayesQR is capable of reconstructing quasispecies at low diversity, showing superior performance in those settings compared to state-of-the-art algorithms. Furthermore, the study on a real HIV-1 dataset demonstrates that our proposed algorithm outperforms or has performance comparable to that of the existing methods in the general setting of viral quasispecies reconstruction.

aBayesQR can be extended and applied to the problem of estimating the population size and the degree of variation among the constituent species in related fields such as immunogenetics. On a related note, bacterial populations are characterized by having relatively lower mutation rates than viral and thus typically have fewer segregating sites on the sequences in a population. The ability of our method to perform highly accurate reconstruction in such settings should be further investigated.

A software aBayesQR is available at <https://github.com/SoyeonA/aBayesQR>. An appendix can be found in a bioRxiv version of this paper which is available at <http://biorxiv.org/content/early/2017/01/27/103630>.

Acknowledgements. This work was funded by the National Science Foundation under grants CCF 1507998 and CCF 1618427.

References

1. Duarte, E., Novella, I., Weaver, S., Domingo, E., Wain-Hobson, S., Clarke, D., Moya, A., Elena, S., De La Torre, J., Holland, J.: RNA virus quasispecies: significance for viral disease and epidemiology. *Infect. Agents Dis.* **3**(4), 201–214 (1994)
2. Lauring, A.S., Andino, R.: Quasispecies theory and the behavior of RNA viruses. *PLoS Pathog.* **6**(7), e1001005 (2010)
3. Posada-Cespedes, S., Seifert, D., Beerwinkler, N.: Recent advances in inferring viral diversity from high-throughput sequencing data. *Virus Res.* (2016)

4. Zagordi, O., Bhattacharya, A., Eriksson, N., Beerenwinkel, N.: ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinform.* **12**(1), 119 (2011)
5. 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 Comput. Biol.* **4**(5), e1000074 (2008)
6. Zagordi, O., Geyrhofer, L., Roth, V., Beerenwinkel, N.: Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. *J. Comput. Biol.* **17**(3), 417–428 (2010)
7. Zagordi, O., Klein, R., Däumer, M., Beerenwinkel, N.: Error correction of next-generation sequencing data and reliable estimation of HIV quasispecies. *Nucleic Acids Res.* **38**(21), 7400–7409 (2010)
8. Astrovskaia, I., Tork, B., Mangul, S., Westbrooks, K., Măndoiu, I., Balfe, P., Zelikovsky, A.: Inferring viral quasispecies spectra from 454 pyrosequencing reads. *BMC Bioinform.* **12**(6), 1 (2011)
9. 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, vol. 4983, pp. 159–170. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-79450-9_15](https://doi.org/10.1007/978-3-540-79450-9_15)
10. Prospero, M.C., Prospero, L., Bruselles, A., Abbate, I., Rozera, G., Vincenti, D., Solmone, M.C., Capobianchi, M.R., Ulivi, G.: Combinatorial analysis and algorithms for quasispecies reconstruction using next-generation sequencing. *BMC Bioinform.* **12**(1), 1 (2011)
11. Prospero, M.C., Salemi, M.: QuRe: software for viral quasispecies reconstruction from next-generation sequencing data. *Bioinformatics* **28**(1), 132–133 (2012)
12. Prabhakaran, S., Rey, M., Zagordi, O., Beerenwinkel, N., Roth, V.: HIV haplotype inference using a propagating dirichlet process mixture model. *IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB)* **11**(1), 182–191 (2014)
13. Töpfer, A., Zagordi, O., Prabhakaran, S., Roth, V., Halperin, E., Beerenwinkel, N.: Probabilistic inference of viral quasispecies subject to recombination. *J. Comput. Biol.* **20**(2), 113–123 (2013)
14. Schirmer, M., Sloan, W.T., Quince, C.: Benchmarking of viral haplotype reconstruction programmes: an overview of the capacities and limitations of currently available programmes. *Briefings Bioinform.* **15**(3), 431–442 (2012)
15. Töpfer, A., Marschall, T., Bull, R.A., Luciani, F., Schönhuth, A., Beerenwinkel, N.: Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput. Biol.* **10**(3), e1003515 (2014)
16. Mangul, S., Wu, N.C., Mancuso, N., Zelikovsky, A., Sun, R., Eskin, E.: Accurate viral population assembly from ultra-deep sequencing data. *Bioinformatics* **30**(12), i329–i337 (2014)
17. Jayasundara, D., Saeed, I., Maheswararajah, S., Chang, B., Tang, S.L., Halgamuge, S.K.: ViQuaS: an improved reconstruction pipeline for viral quasispecies spectra generated by next-generation sequencing. *Bioinformatics* **31**(6), 886–896 (2014)
18. Le, T., Chiarella, J., Simen, B.B., Hanczaruk, B., Egholm, M., Landry, M.L., Dieckhaus, K., Rosen, M.I., Kozal, M.J.: Low-abundance HIV drug-resistant viral variants in treatment-experienced persons correlate with historical antiretroviral use. *PLoS ONE* **4**(6), e6079 (2009)

19. Simen, B.B., Simons, J.F., Hullsiek, K.H., Novak, R.M., MacArthur, R.D., Baxter, J.D., Huang, C., Lubeski, C., Trenchalk, G.S., Braverman, M.S., et al.: Low-abundance drug-resistant viral variants in chronically HIV-infected, antiretroviral treatment-naive patients significantly impact treatment outcomes. *J. Infect. Dis.* **199**(5), 693–701 (2009)
20. Di Giallonardo, F., Töpfer, A., Rey, M., Prabhakaran, S., Dupont, Y., Leemann, C., Schmutz, S., Campbell, N.K., Joos, B., Lecca, M.R., et al.: Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic Acids Res.* **42**(14), e115 (2014)
21. Sasirekha, K., Baby, P.: Agglomerative hierarchical clustering algorithm-a review. *Int. J. Sci. Res. Publ.* **3**(3) (2013)
22. Jung, S.Y., Kim, T.S.: An agglomerative hierarchical clustering using partial maximum array and incremental similarity computation method. In: *Proceedings IEEE International Conference on Data Mining, ICDM 2001*, pp. 265–272. IEEE (2001)
23. Lancia, G., Bafna, V., Istrail, S., Lippert, R., Schwartz, R.: SNPs problems, complexity, and algorithms. In: Heide, F.M. (ed.) *ESA 2001*. LNCS, vol. 2161, pp. 182–193. Springer, Heidelberg (2001). doi:[10.1007/3-540-44676-1_15](https://doi.org/10.1007/3-540-44676-1_15)
24. Lippert, R., Schwartz, R., Lancia, G., Istrail, S.: Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings Bioinform.* **3**(1), 23–31 (2002)
25. Archer, J., Baillie, G., Watson, S.J., Kellam, P., Rambaut, A., Robertson, D.L.: Analysis of high-depth sequence data for studying viral diversity: a comparison of next generation sequencing platforms using Segminator II. *BMC Bioinform.* **13**(1), 47 (2012)

BeWith: A Between-Within Method for Module Discovery in Cancer using Integrated Analysis of Mutual Exclusivity, Co-occurrence and Functional Interactions (Extended Abstract)

Phuong Dao¹, Yoo-Ah Kim¹, Sanna Madan², Roded Sharan³(✉),
and Teresa M. Przytycka¹(✉)

¹ National Center of Biotechnology Information, NLM, NIH, Bethesda, MD, USA
przytyck@ncbi.nlm.nih.gov

² Department of Computer Science, University of Maryland, College Park, MD, USA

³ Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel
roded@post.tau.ac.il

The analysis of the cancer mutational landscape has been instrumental in studying the disease and identifying its drivers and subtypes. In particular, mutual exclusivity of mutations in cancer drivers has recently attracted a lot of attention. These relationships can help identify cancer drivers, cancer-driving pathways, and subtypes [1–4]. The co-occurrence of mutations has also provided critical information about possible synergistic effects between gene pairs [5].

Importantly, both properties can arise due to several different reasons, making the interpretation challenging. Specifically, mutually exclusive mutations within a functionally interacting gene module may indicate that a mutation in either of the two genes dysregulates the same pathway. On the other hand, mutually exclusive mutations might reflect a situation where two genes drive different cancer types, which is more likely to occur between genes belonging to different pathways. We have previously observed that within cancer type mutual exclusivity is more enriched with physically interacting genes than between cancer types mutual exclusivity [2]. Thus, the interaction information between genes might provide hints toward the nature of the mutual exclusivity. In addition, mutual exclusivity is not necessarily limited to cancer drivers, and therefore a proper understanding of this property is critical for obtaining a better picture of cancer mutational landscape and for cancer driver prediction.

The co-occurrence of mutations might also emerge due to a number of different causes. Perhaps the most important case is when the co-inactivation of two genes simultaneously might be beneficial for cancer progression such as the co-occurrence of TP53 mutation and MYC amplification [5] or co-occurring mutations in PIK3CA and RAS/KRAS [6]. Alternatively, co-occurrence of somatic mutations might indicate the presence of a common mutagenic process.

Given the diversity of reasons for observing the mutational patterns, we hypothesised that jointly considering co-occurrence, mutual exclusivity and functional interaction relationships will yield a better understanding of the mutational

P. Dao and Y.-A. Kim—Equal contribution.

landscape of cancer. To address this challenge, we designed a general framework, named BeWith, for identifying modules with different combinations of mutation and interaction patterns. On a high level, BeWith tackles the following problem: given a set of genes and two types of edge scoring functions (within and between scores), find the clusters of genes such that genes within a cluster maximize the within scores while gene pairs in two different clusters maximize the between scores. We formulated the BeWith module identification problem as an Integer Linear Programming (ILP) and solved it to optimality.





In this work, we focused on three different settings of the BeWith framework: BeME-WithFun (mutual exclusivity between different modules and functional similarity of genes within modules), BeME-WithCo (mutual exclusivity between modules and co-occurrence within modules), and BeCo-WithMEFun (co-occurrence between modules while enforcing mutual exclusivity and functional interactions within modules). By utilizing different settings of within and between properties, BeWith revealed complex relations between mutual exclusivity, functional interactions, and co-occurrence. In particular, BeME-WithFun identified functionally coherent modules containing cancer associated genes. By looking for co-occurring mutations inside a module, the BeME-WithCo setting allowed us to investigate mutated modules in a novel way and help uncover synergetic gene pairs in breast cancer. Going beyond cancer driving mutations, the setting also provided insights into underlying mutagenic processes in cancer. Importantly, the BeWith formulation is very general and can be used to interrogate other aspects of the mutational landscape by exploring different combinations of within-between definitions and constraints with simple modifications.

Implementation is available at <https://www.ncbi.nlm.nih.gov/CBBresearch/Przytycka/software/bewith.html>.

References

1. Babur, Ö., Gönen, M., Aksoy, B.A., Schultz, N., Ciriello, G., Sander, C., Demir, E.: Systematic identification of cancer driving signaling pathways based on mutual exclusivity of genomic alterations. *Genome Biol.* **16**, 45 (2015)
2. Kim, Y.A., Cho, D.Y., Dao, P., Przytycka, T.M.: MEMCover: integrated analysis of mutual exclusivity and functional network reveals dysregulated pathways across multiple cancer types. *Bioinformatics* **31**(12), i284–92 (2015)
3. Leiserson, M.D.M., Blokh, D., Sharan, R., Raphael, B.J.: Simultaneous identification of multiple driver pathways in cancer. *PLoS Comput. Biol.* **9**(5), e1003054 (2013)
4. Leiserson, M.D.M., Hsin-Ta, W., Fabio, V., Raphael, B.J.: CoMEt: a statistical approach to identify combinations of mutually exclusive alterations in cancer. *Genome Biol.* **16**(1), 72 (2015)
5. Ulz, P., Heitzer, E., Speicher, M.R.: Co-occurrence of MYC amplification and TP53 mutations in human cancer. *Nat. Genet.* **48**(2), 104–106 (2016)
6. Cancer Genome Atlas Network: Comprehensive molecular portraits of human breast tumours. *Nature* **490**(7418), 61–70 (2012)

K-mer Set Memory (KSM) Motif Representation Enables Accurate Prediction of the Impact of Regulatory Variants

Yuchun Guo^{}, Kevin Tian^{}, Haoyang Zeng^{},
and David K. Gifford^{}

MIT, Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA, USA
gifford@mit.edu

Introduction

The discovery and representation of transcription factor (TF) DNA sequence binding specificities is critical for understanding regulatory networks and interpreting the impact of non-coding genetic variants. The position weight matrix (PWM) model does not represent binding specificities accurately because it assumes that base positions in the motif are independent. Recent studies have shown that DNA sequences proximal to a TF motif core may affect DNA shape and hence TF binding [1]. We hypothesized that a motif model that preserves base positional dependences and includes proximal flanking bases would improve upon existing motif models.

Approach

We introduce the K-mer Set Memory (KSM) model that represents TF binding specificity as a set of aligned gapped and ungapped k-mers that are over-represented at TF binding sites. KSM motif matching requires an exact match of one or more component k-mers, thus preserving inter-positional dependences. The k-mers matching the motif core and the flanking bases are combined non-additively to score the KSM motif matches. We have developed a *de novo* motif discovery method called KMAC to learn KSM and corresponding PWM motifs from TF ChIP-seq data.

Results

We compared KMAC with four state-of-the-art motif discovery methods, MEME, MEME-chip, Homer, and Weeder2, on discovering PWM motifs from the binding sites of 78 TFs in 209 ENCODE ChIP-seq experiments. KMAC identified previously published PWM motifs in more experiments than the other methods.

We then compared the performance of KSM versus other motif models in predicting *in vivo* TF binding by discriminating TF-bound sequences from unbound sequences. For the GAPB dataset, a KSM motif outperforms PWM motifs computed by KMAC, MEME, and Homer, as well as representations that model base inter-dependences such as TFFM [2] and Slim [3] (Fig. 1A). For sequences with identical PWM scores, the KSM scores of the positive sequences are generally higher than those of the negative sequences (Fig. 1B). This is because KSM motif matches in positive sequences often contain more KSM k-mers that cover motif flanking bases than those in negative sequences.

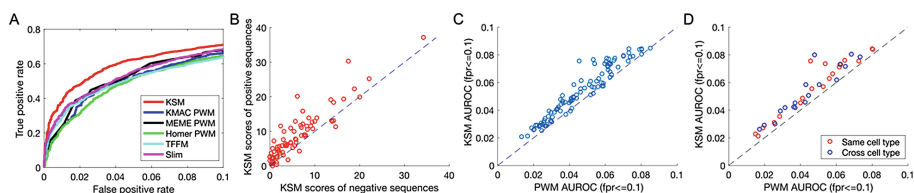


Fig. 1. KSM outperforms PWM in predicting *in vivo* TF binding in held-out data.

Out of 104 ChIP-seq datasets, KSMS outperform PWMs in 85 datasets; and PWMs do not outperform KSMS in any dataset (Fig. 1C). Overall, a KSM significantly outperforms a PWM ($p = 4.79e-18$, paired Wilcoxon signed rank test) and a TFFM in predicting TF binding ($p = 0.045$, paired Wilcoxon signed rank test). We also found that a KSM is able to generalize across cell types. For 19 unique TFs, KSMS significantly outperformed PWMs when a motif learned from one cell type (K562) is used to predict binding of the same TF in another cell type (GM12878 or H1-hESC) (Fig. 1D). The KSM predictions across the cell types perform similarly to the KSM predictions in the same cell type ($p = 0.091$, paired Wilcoxon signed rank test).

(A) The partial ROC (fpr ≤ 0.1) of KSM and other models for predicting ChIP-seq binding of GABP in K562 cells. (B) Comparison of the mean KSM scores of positive versus negative sequences that corresponds to the same PWM scores. Each point represents a set of sequences that have the same PWM score. (C) Comparison of the median AUROC (fpr ≤ 0.1) scores of KSM and PWM for 104 experiments with five cross-validation datasets. (D) Similar to (C), but comparing KSM and PWM in the same cell type (red) or cross cell type (blue) in 19 TFs.

In addition, we evaluated the ability of different sequence features to predict the regulatory activity of e-QTLs using a computational framework [4] that performed the best in the CAGI 4 “eQTL-causal SNPs” challenge. We found that KSM derived features (AUPRC = 0.461, AUROC = 0.683) outperformed Homer PWM derived features (AUPRC = 0.434, AUROC = 0.629), MEME PWM derived features (AUPRC = 0.408, AUROC = 0.619) and sequence features derived from DeepSEA [5], a deep learning model (AUPRC = 0.396, AUROC = 0.628), in predicting the differential regulatory activities of e-QTL alleles. The combined KSM and DeepSEA features achieved the best performance (AUPRC = 0.464, AUROC = 0.696).

Finally, we have created a public resource of KSM and PWM motifs from more than one thousand ENCODE TF ChIP-seq datasets.

Conclusion

We found that the K-mer Set Model (KSM) is a more powerful motif representation than the PWM and TFFM models for identifying held-out DNA sequences that are bound by a TF. We also found that KMAC more accurately discovers PWM motifs than other tested methods. Thus, the KSM and PWM models produced by the KMAC method improve the ability to model TF binding specificities, and enable more accurate characterization of non-coding genetic variants.

References

1. Slattery, M., Riley, T., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J., Mann, R.S.: Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. *Cell* **147**, 1270–1282 (2011)
2. Mathelier, A., Wasserman, W.W.: The next generation of transcription factor binding site prediction. *PLoS Comput. Biol.* **9**, e1003214 (2013)
3. Keilwagen, J., Grau, J.: Varying levels of complexity in transcription factor binding motifs. *Nucl. Acids Res.* **43**, e119–e119 (2015)
4. Zeng, H., Edwards, M.D., Guo, Y., Gifford, D.K.: Accurate eQTL prioritization with an ensemble-based framework. *bioRxiv*. 69757 (2016)
5. Zhou, J., Troyanskaya, O.G.: Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Meth.* **12**, 931–934 (2015)

Network-Based Coverage of Mutational Profiles Reveals Cancer Genes

Borislav H. Hristov and Mona Singh^(✉)

Department of Computer Science and Lewis-Sigler Institute for Integrative Genomics,
Princeton University, Princeton, NJ, USA
mona@cs.princeton.edu

Summary. A central goal in cancer genomics is to identify the somatic alterations that underpin tumor initiation and progression. This task is challenging as the mutational profiles of cancer genomes exhibit vast heterogeneity, with many alterations observed within each individual, few shared somatically mutated genes across individuals, and important roles in cancer for both frequently and infrequently mutated genes. While commonly mutated cancer genes are readily identifiable, those that are rarely mutated across samples are difficult to distinguish from the large numbers of other infrequently mutated genes. Here, we introduce a method that considers per-individual mutational profiles within the context of protein-protein interaction networks in order to identify small connected subnetworks of genes that, while not individually frequently mutated, comprise pathways that are perturbed across (i.e., “cover”) a large fraction of the individuals. We devise a simple yet intuitive objective function that balances identifying a small subset of genes with covering a large fraction of individuals. We show how to solve this problem optimally using integer linear programming and also give a fast heuristic algorithm that works well in practice. We perform a large-scale evaluation of our resulting method, **nCOP**, on 6,038 TCGA tumor samples across 24 different cancer types. We demonstrate that our approach is more effective in identifying cancer genes than both methods that do not utilize any network information as well as state-of-the-art network-based methods that aggregate mutational information across individuals. Overall, our work demonstrates the power of combining per-individual mutational information with interaction networks in order to uncover genes functionally relevant in cancers, and in particular those genes that are less frequently mutated.

Methods. We model the biological network as an undirected graph G where each vertex represents a gene, and there is an edge between two vertices if an interaction has been found between the corresponding proteins. We annotate each node in the network with the IDs of the individuals having one or more mutations in the corresponding gene. Our goal is to find a relatively small connected component G' such that most patients have mutations in one of the genes within it. A small subgraph is more likely to consist of functionally related genes and is less likely to be the result of overfitting to the set of individuals whose diseases we are analyzing. However, we would also like our model to have the greatest possible explanatory power—that is, to account for, or cover, as many patients as possible by including genes that are mutated within their cancers.

We formulate our problem to balance these two competing objectives with a parameter α that controls the trade-off between keeping the subgraph small and covering more patients as to minimize $\alpha X + (1 - \alpha)Size(G')$, where X is the fraction of patients that do not have an alteration in a gene included in G' (i.e., they are uncovered) and $Size(G')$ is the size of the subgraph.

For a fixed value of α , we have developed two approaches to solve the underlying optimization problem: one based on linear programming and the other a fast greedy heuristic. To select an appropriate α for a set of cancer samples, we devise a simple but effective data-driven cross-validation technique. In particular, we split our samples into training, validation and test sets. A test set of (10%) of the patients is completely withheld. While varying α in small increments in the interval $(0; 1)$, the remaining data is repeatedly split (100 times for each value of α) into training (80%) and validation (20%) sets. For each split, the algorithm is run on the training set to find G' . The fractions of patients covered (by the selected G') in the training and validation sets are compared. The parameter α is selected where performance on the validation sets deviates as compared to the training sets. Once α is chosen for a set of cancer samples, we repeatedly (1000 times) run the algorithm on this set, each time withholding a fraction (15%) of the patients in order to introduce some randomness in the process. Genes are then ranked by the number of times they appear in G' .

Results. We run nCOP on somatic point mutation data from 24 different TCGA cancer types. We show that nCOP effectively uses network information to uncover known cancer genes by considering how well it recapitulates known cancer genes (CGCs) in comparison to network-agnostic methods. We find that nCOP outperforms MutSigCV 2.0 (Lawrence et al. 2013), a state-of-the-art frequency-based approach, on 21 of the 24 cancer types, and a basic set cover approach on all 24 types. We also show that nCOP is more effective in uncovering known cancer genes than Muffinn (Cho et al. 2016), a recent network-based method that considers mutations found in interacting genes. Finally, we examine the non-CGC genes which are highly ranked by nCOP and observe that they tend to be less frequently mutated. Our results are consistent across the three different networks we used (HPRD, HINT, and Biogrid), showing the robustness of the method with respect to the underlying network. Further, we demonstrate that our training-validation-test set framework is a highly effective approach for choosing an α that balances patient coverage with subnetwork size.

In summary, we present nCOP, a method that incorporates individual mutational profiles with protein-protein interaction networks, and show it is a powerful approach for uncovering cancer genes. Researchers can use our framework to rapidly and easily prioritize cancer genes, as nCOP requires only straightforward inputs and runs on a desktop machine. Indeed, nCOP's efficiency, robustness, and ease of use make it an excellent choice to investigate cancer as well as possibly other complex diseases. nCOP can be freely downloaded at: <http://compbio.cs.princeton.edu/ncop/>.

Ultra-Accurate Complex Disorder Prediction: Case Study of Neurodevelopmental Disorders

Linh Huynh¹ and Fereydoon Hormozdiari^{1,2,3}(✉)

¹ Genome Center, UC Davis, Davis, USA
fhormozd@ucdavis.edu

² MIND institute, UC Davis, Davis, USA

³ Biochemistry and Molecular Medicine, UC Davis, Davis, USA

Motivation and Problem Definition. Early prediction of complex disorders (e.g. autism, intellectual disability or schizophrenia) is one of the main goals of personalized genomics and precision medicine. Considering the high genetic heritability of neurodevelopmental disorders ($h^2 > 0.5$ for autism [1]) we are proposing a novel problem and framework for accurate prediction of autism and related disorders based on rare and *de novo* genetic variants [2]. However, a positive diagnosis/prediction of a complex disorder (e.g., autism or intellectual disability) can have a severe negative psychological and economical impact on affected individuals and their family. Thus, one of the primary practical constraints in developing models and methods for prediction of a severe complex disorder *is to guarantee a false positive prediction/discovery rate (FDR) of virtually zero*. Hence, we are introducing a novel problem for prediction of complex disorders for a subset of affected cases with very low false positive prediction. We denote this problem as Ultra-Accurate Disorder Prediction (UADP) problem.

Methods. We have proposed framework for solving the UADP problem denoted as Odin (**O**racle for **D**isorder predictio**N**). Odin will intuitively predict an input/test sample to be an affected case if and only if it satisfies two conditions:

1. The input sample is “far” from any unaffected control sample
2. The input sample is “close” to many affected case samples

For satisfying the first condition, we simply use the nearest neighbor (NN) approach using a distance function (e.g., Euclidean distance). For satisfying the second condition, we first develop a novel algorithm that finds a cluster (together with a dimension reduction) that contains a significant number of affected cases and does not contain any unaffected controls. This cluster is denoted as *unicolor cluster*, as it only includes the affected cases. We denote the problem of finding such a cluster as Unicolor Clustering with Dimensionality Reduction (UCDR) problem. An input sample passes the second condition if it falls inside of this unicolor cluster. A weighted version of UCDR, where we can assign weights to each dimension is denoted as Weighted Unicolor Clustering with Dimensionality Reduction (WUCDR) problem. We have shown that the decision version of an UCDR instance is NP-complete using reduction from equal-subset sum problem [3]. We propose an iterative approach with two steps to solve the WUCDR problem. In the first step, given weights for each dimension, we find the cluster to

cover a maximum number of affected cases. In the second step, given the cluster from the first step, we find the new set of weights for each dimension by using a linear programming (LP) formulation.

Results. We used the leave-one-out (LOO) cross validation technique to compare the prediction power of Odin and the of k-NN and SVM classifiers. As our stated goal is to keep the false positive prediction of unaffected samples as cases close to zero, we will only consider the most conservative results for each method (i.e., where false discovery rate (FDR) < 0.01). For the same FDR threshold, Odin's true positive rate for predicting autism is at least twice higher than the best k-NN result (for various values of k) and significantly higher than SVM. Our experimental results indicate the ability of our approach in ultra-accurate prediction of autism spectrum disorder (ASD) in additional 8% of cases which do not have a severe mutation in *recurrently* mutated genes, with less than 0.5% of false positive prediction rate for unaffected controls.

Odin is publicly available at <https://github.com/HormozdiariLab/Odin>.

References

1. Sandin, S., Lichtenstein, P., Kuja-Halkola, R., Larsson, H., Hultman, C.M., Reichenberg, A.: The familial risk of autism. *JAMA* **311**(17), 1770–1777 (2014)
2. Iossifov, I., O’Roak, B.J., Sanders, S.J., Ronemus, M., Krumm, N., Levy, D., Stessman, H.A., Witherspoon, K.T., Vives, L., Patterson, K.E., et al.: The contribution of de novo coding mutations to autism spectrum disorder. *Nature* **515**(7526), 216–221 (2014)
3. Woeginger, G.J., Yu, Z.: On the equal-subset-sum problem. *Inf. Process. Lett.* **42**(6), 299–302 (1992)

Inference of the Human Polyadenylation Code

Michael K.K. Leung^{1,2(✉)}, Andrew Delong^{1,2},
and Brendan J. Frey^{1,2,3}

¹ Deep Genomics, MaRS Centre, Heritage Building, Suite 320,
Toronto, ON M5G 1L7, Canada
mleung@psi.toronto.edu

² Department of Electrical and Computer Engineering, University of Toronto,
Toronto M5S 3G4, Canada

³ Banting and Best Department of Medical Research, University of Toronto,
Toronto M5S 3E1, Canada

Abstract. Processing of transcripts at the 3'-end is a two-step procedure that involves cleavage at a polyadenylation site followed by the addition of a poly (A)-tail. By selecting which polyadenylation site is cleaved in transcripts with multiple sites, alternative polyadenylation enables genes to produce transcript isoforms with different 3'-ends. To facilitate the identification and treatment of disease-causing mutations that affect polyadenylation and to understand the underlying regulatory processes, a computational model that can accurately predict polyadenylation patterns based on genomic features is desirable. Previous works have focused on identifying candidate polyadenylation sites, as well as classifying sites which may be tissue-specific. However, what is lacking is a predictive model of the underlying mechanism of site selection, competition, and processing efficiency in a tissue-specific manner. We develop a deep learning model that trains on 3'-end sequencing data and predicts tissue-specific site selection among competing polyadenylation sites in the 3' untranslated region of the human genome.

Two neural network architectures are evaluated: one built on hand-engineered features, and another that directly learns from the genomic sequence. The hand-engineered features include polyadenylation signals, cis-regulatory elements, n-mer counts, nucleosome occupancy, and RNA-binding protein motifs. The direct-from-sequence model is inferred without prior knowledge on polyadenylation, based on a convolutional neural network trained with genomic sequences surrounding each polyadenylation site as input. Both models are trained using the Tensor Flow library.

The proposed polyadenylation code can predict functional site selection among competing polyadenylation sites across all tissues. Importantly, it does so without relying on evolutionary conservation. The model can directly distinguish pathogenic from benign variants that appear near annotated polyadenylation sites, achieving a classification AUC of 0.98 ($p < 1 \times 10^{-8}$) on ClinVar. We further demonstrate the potential use of the same model to predict the effects of antisense oligonucleotides to redirect polyadenylation and to scan the genome to find candidate polyadenylation sites.

Folding Membrane Proteins by Deep Transfer Learning

Zhen Li^{1,3}, Sheng Wang^{1,2}, Yizhou Yu³, and Jinbo Xu¹(✉)

¹ Toyota Technological Institute at Chicago, Chicago, USA
jinboxu@gmail.com

² Department of Human Genetics, University of Chicago, Chicago, USA

³ Department of Computer Science, University of Hong Kong,
Hong Kong, China

Membrane proteins (MPs) are important for drug design and have been targeted by approximately half of current therapeutic drugs. In many genomes 20–40% of genes encode MPs. In particular, Human genome has >5,000 reviewed MPs and more than 3000 of them are non-redundant at 25% sequence identity. Experimental determination of MP structures is challenging as they are often too large for NMR experiments and very difficult to crystallize. As of October 2016, there are ~510 non-redundant MPs with solved structures, and a majority number of MPs have no solved structures.

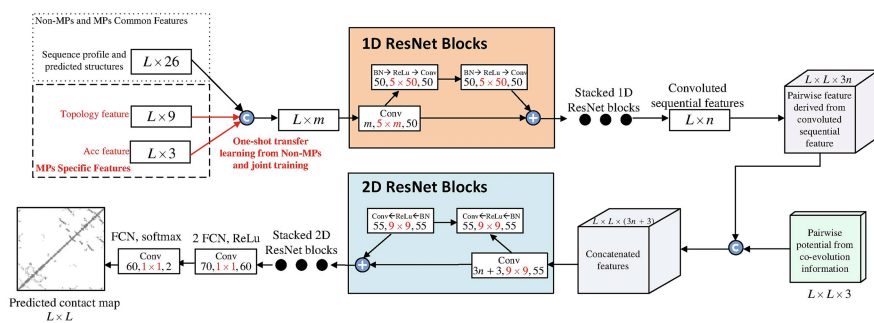


Fig. 1. Overview of our deep learning model for MP contact prediction where L is the sequence length of one MP under prediction.

Developing computational methods for MP structure prediction is challenging partially due to lack of MPs with solved structures for homology modeling or for parameter estimation of ab initio folding. Recently contact-assisted ab initio folding has made good progress [1]. This technique first predicts the contacts of a protein and then use predicted contacts as restraints to guide folding simulation. Contact-assisted folding heavily depends on accurate prediction of protein contacts. Co-evolution analysis can predict contacts accurately for some proteins with a large number of sequence homologs. However, protein families without good templates in PDB on average have many fewer sequences homologs than those with good templates.

Zhen Li and Sheng Wang – These authors contributed equally to the work as first authors.

We have developed a deep transfer learning method that can significantly improve MP contact prediction by learning contact occurrence patterns from thousands of non-membrane proteins (non-MPs). We treat a contact map as an image and formulate contact prediction similarly as pixel-level image labeling. As shown in Fig. 1, our deep network is composed of two concatenated deep residual neural networks. Each network consists of some residual blocks and each block has 2 convolution and ReLU layers. The first residual network conducts 1-dimensional (1D) convolutional transformations of sequential features. Its output is converted to a 2-dimensional (2D) matrix by an operation called outer concatenation and fed into the 2nd residual network together with the pairwise features. The 2nd residual network conducts 2D convolutional transformations of its input and feeds its output into logistic regression, which predicts the probability of any two residues in a contact. We predict all the contacts of a protein simultaneously to capture contact occurrence patterns and improve prediction accuracy. We use two types of protein features: sequential features and pairwise features. The sequential features include protein sequence profile, secondary structure and solvent accessibility predicted by RaptorX-Property [2]. The pairwise features include co-evolutionary strength generated by CCMpred [3], mutual information and pairwise contact potential. Some MP-specific features are also tested.

We studied three training strategies: NonMP (only non-MPs used as training proteins), MP-only (only MPs used as training proteins) and Mixed (both non-MPs and MPs used as training proteins). Tested on 510 non-redundant MPs, our deep models trained by NonMP only, MP-only and Mixed have top L/10 long-range prediction accuracy 0.69, 0.63 and 0.72, respectively, all much better than CCMpred (0.47) and the CASP11 winner MetaPSICOV (0.55). When only contacts in transmembrane regions are evaluated, our models have top L/10 long-range accuracy 0.57, 0.53, and 0.62, respectively, again much better than MetaPSICOV (0.45) and CCMpred (0.40). These results suggest that sequence-structure relationship learned by our deep model from non-MPs generalizes well to MP contact prediction and that non-MPs and MPs share common contact occurrence patterns.

Improved contact prediction also leads to better contact-assisted folding. We build 3D structure models for a MP by feeding its top predicted contacts to the CNS package, and evaluate model quality by TMscore, which ranges from 0 to 1, indicating the worst and best models, respectively. A model with TMscore >0.5 (0.6) is (very) likely to have a correct fold. The average TMscore (RMSD in Å) of the 3D models built by our three models MP-only, NonMP-only and Mixed are 0.45 (14.9), 0.49 (13.2), and 0.52 (10.8), respectively. By contrast, the average TMscore (RMSD in Å) of the 3D models built from MetaPSICOV and CCMpred-predicted contacts are 0.39 (16.7) and 0.36 (17.0), respectively. When the best of top 5 models are considered and TMscore = 0.6 is used as cutoff, our three models can predict correct folds for 110, 160, and 200 of 510 MPs, respectively, while MetaPSICOV and CCMpred can do so for only 77 and 56 of them, respectively. Homology modeling can correctly fold 41 MPs when MPs and non-MPs are used as templates and 3 MPs when only non-MPs are used as templates. When TMscore = 0.5 is cutoff, our Mixed method, MetaPSICOV, and CCMpred can predict correct folds for 283, 147, and 122 MPs, respectively.

References

1. Wang, S., Sun, S., Li, Z., Zhang, R., Xu, J.: Accurate De Novo prediction of protein contact map by ultra-deep learning model. *PLoS Comput. Biol.* **13**, e1005324 (2017)
2. Wang, S., Li, W., Liu, S., Xu, J.: RaptorX-Property: a web server for protein structure property prediction. *Nucleic Acids Res.* gkw306 (2016)
3. Seemayer, S., Gruber, M., Söding, J.: CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics* **30**, 3128–3130 (2014)

A Network Integration Approach for Drug-Target Interaction Prediction and Computational Drug Repositioning from Heterogeneous Information

Yunan Luo^{1,3}, Xinbin Zhao², Jingtian Zhou², Jinling Yang¹, Yanqing Zhang¹,
Wenhua Kuang², Jian Peng³(✉), Ligong Chen²(✉), and Jianyang Zeng¹(✉)

¹ Institute for Interdisciplinary Information Sciences, Tsinghua University,
Beijing, China

zengjy321@tsinghua.edu.cn

² School of Pharmaceutical Sciences, Tsinghua University, Beijing, China

ligongchen@biomed.tsinghua.edu.cn

³ Department of Computer Science, University of Illinois at Urbana-Champaign,
Champaign, USA

jianpeng@illinois.edu

The emergence of large-scale genomic, chemical and pharmacological data provides new opportunities for drug discovery and repositioning. Systematic integration of these heterogeneous data not only serves as a promising tool for identifying new drug-target interactions (DTIs), which is an important step in drug development, but also provides a more complete understanding of the molecular mechanisms of drug action. In this work, we integrate diverse drug-related information, including drugs, proteins, diseases and side-effects, together with their interactions, associations or similarities, to construct a heterogeneous network with 12,015 nodes and 1,895,445 edges. We then develop a new computational pipeline, called DTINet, to predict novel drug-target interactions from the constructed heterogeneous network. Specifically, DTINet focuses on learning a low-dimensional vector representation of features for each node, which accurately explains the topological properties of individual nodes in the heterogeneous network, and then predicts the likelihood of a new DTI based on these representations via a vector space projection scheme. DTINet achieves substantial performance improvement over other state-of-the-art methods for DTI prediction. Moreover, we have experimentally validated the novel interactions between three drugs and the cyclooxygenase (COX) protein family predicted by DTINet, and demonstrated the new potential applications of these identified COX inhibitors in preventing inflammatory diseases. These results indicate that DTINet can provide a practically useful tool for integrating heterogeneous information to predict new drug-target interactions and repurpose existing drugs.

The full paper of DTINet is available at [1]. The source code of DTINet and the input heterogeneous network data can be downloaded from <https://github.com/luoyunan/DTINet>.

Y. Luo et al.—These authors contributed equally to this work.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 383–384, 2017.

DOI: 10.1007/978-3-319-56970-3

Acknowledgments. This work was supported in part by the National Basic Research Program of China (Grant 2011CBA00300 and 2011CBA00301), the National Natural Science Foundation of China (Grant 61033001, 61361136003, 61472205 and 81470839), the China's Youth 1000-Talent Program, the Beijing Advanced Innovation Center for Structural Biology, and the Tsinghua University Initiative Scientific Research Program (Grant. 20161080086). J.P. received support as an Alfred P. Sloan Research Fellow. We acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

Reference

1. Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Peng, J., Chen, L., Zeng, J.: A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information (2017). bioRxiv. doi:<https://doi.org/10.1101/100305>

Epistasis in Genomic and Survival Data of Cancer Patients

Dariusz Matlak and Ewa Szczurek^(✉)

Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Warsaw, Poland
szczurek@mimuw.edu.pl

Extended Abstract

Fitness is a measure of replicative and survival success of an individual, relative to competitors in the same population. Epistasis is an interaction between genes, and refers to departure from independence of effects that their genomic alterations have on fitness. Beerenwinkel et al. [2] defined epistatic interactions not only among two, but also more genes. Here, we consider epistasis of genes in their contribution to fitness of tumors in cancer patients.

Current state of the art cancer therapies have limited efficacy due to toxicity and rapid development of drug resistance. Recently, therapies exploiting synthetic lethal interactions between genes were proposed to overcome these difficulties [8]. Synthetic lethality occurs when the co-inactivation of two genes results in cellular death, while inactivation of each individual gene is viable. In cancer, one gene inactivation can already occur via the endogenous mutation in the tumor cells, and not in the normal cells of the body. Thus, applying a drug that targets the synthetic lethal partner of that gene will selectively kill cancer cells, leaving the rest viable. A famous example is the interaction between *BRCA1* and *PARP1*. In *BRCA1* deficient cells, treatment with a PARP inhibitor, such as Olaparib [4], is expected to result in selective tumor cell death.

Synthetic lethality is, however, context dependent. For example, compared to the dramatic effect that PARP1 inhibition has on *BRCA1*-deficient cell lines, the efficacy of Olaparib therapy on patients was low, since a positive response was observed in less than 50% of BRCA-mutated cancers [3]. This raises the crucial issue of therapeutic biomarkers. For *BRCA1* and *PARP1*, mutation of *TP53BP1* in addition to *BRCA1* was observed to alleviate the synthetic lethal effect [1]. Thus, in *TP53BP1* deficient tumors, administrating Olaparib is not justified, and unaltered *TP53BP1* is a biomarker of this therapy. Such dependence of pairwise interaction on the mutational status of a third gene is represented by conditional epistasis, a type of the triple epistatic interactions [2].

Experimental approaches to identification of synthetic lethality in human cancer are overwhelmed by the number of, and thus test only small subsets of all possible interactions [7]. The effort and money required for these experiments calls for a pre-selection of synthetic lethal partners based on the computational analysis of existing data. Previous methods [5, 9] aimed at deciphering synthetic lethality from somatic alteration, expression or survival data of cancer patients.

Here, we introduce SurvLRT, an approach for identification of epistatic gene pairs and triplets in human cancer. We propose a statistical model based on Lehman alternatives [6], which allows to estimate fitness of tumors with a given genotype from survival of carrier patients. We assume that a decrease of fitness of tumors due to a particular genotype is exhibited by a proportional increase of survival of the patients. Based on these assumptions, we introduce a likelihood ratio test for the significance of a given pairwise or triple epistatic interaction. In the test, the null model assumes that there is no epistasis and the gene alterations are independent, while the alternative assumes otherwise. The approach can detect both positive and negative interactions. Compared to our previous approach [9], SurvLRT offers a more natural interpretation of the notion of fitness, as well as a direct statistical test for the significance of epistasis. We analyze the sensitivity and power of SurvLRT in a controlled setting of simulated data. Next, we show that, compared to previous methods, our method performs favorably in predicting known pairwise synthetic lethal interactions. Finally, we apply SurvLRT to detect therapeutic biomarkers, first by recapitulating *TP53BP1*, the known biomarker for therapies based on the *BRCA1*, *PARP1* interaction, and second by identifying a genomic region deleted in tumors as a new and even more significant biomarker.

Acknowledgement. This work was partially supported by the Polish National Science Center grant numbers 2015/16/W/NZ2/00314 and 2015/19/P/NZ2/03780.

References

1. Aly, A., Ganesan, S.: BRCA1, PARP, and 53BP1: conditional synthetic lethality and synthetic viability. *J. Mol. Cell Biol.* **3**(1), 66–74 (2011)
2. Beerenwinkel, N., Pachter, L., Sturmfels, B.: Epistasis and shapes of fitness landscapes. *Stat. Sinica* **17**, 1317–1342 (2007)
3. Chan, S.L., Mok, T.: PARP inhibition in BRCA-mutated breast and ovarian cancers. *Lancet* **376**(9737), 211–213 (2010)
4. Hutchinson, L.: Targeted therapies: PARP inhibitor olaparib is safe and effective in patients with BRCA1 and BRCA2 mutations. *Nat. Rev. Clin. Oncol.* **7**(10), 549 (2010)
5. Jerby-Arnon, L., Pfetzer, N., Waldman, Y.Y., McGarry, L., James, D., Shanks, E., Seashore-Ludlow, B., Weinstock, A., Geiger, T., Clemons, P.A., Gottlieb, E., Ruppin, E.: Predicting cancer-specific vulnerability via data-driven detection of synthetic lethality. *Cell* **158**(5), 1199–1209 (2014)
6. Lehman, E.L.: The power of rank tests. *Ann. Math. Statist.* **24**(1), 23–43 (1953)
7. Lord, C.J., McDonald, S., Swift, S., Turner, N.C., Ashworth, A.: A high-throughput RNA interference screen for DNA repair determinants of PARP inhibitor sensitivity. *DNA Repair (Amst.)* **7**(12), 2010–2019 (2008)
8. Porcelli, L., Quatrala, A.E., Mantuano, P., Silvestris, N., Brunetti, A.E., Calvert, H., Paradiso, A., Azzariti, A.: Synthetic lethality to overcome cancer drug resistance. *Curr. Med. Chem.* **19**(23), 3858–3873 (2012)
9. Szczurek, E., Misra, N., Vingron, M.: Synthetic sickness or lethality points at candidate combination therapy targets in glioblastoma. *Int. J. Cancer* **133**(9), 2123–2132 (2013)

Ultra-Fast Identity by Descent Detection in Biobank-Scale Cohorts Using Positional Burrows-Wheeler Transform

Ardalan Naseri¹, Xiaoming Liu², Shaojie Zhang¹(✉), and Degui Zhi³

¹ Department of Computer Science, University of Central Florida,
Orlando, FL 32816, USA
shzhang@cs.ucf.edu

² Department of Epidemiology, Human Genetics and Environmental Science,
University of Texas Health Science Center at Houston, Houston, TX 77030, USA

³ School of Biomedical Informatics,
University of Texas Health Science Center at Houston, Houston, TX 77030, USA
degui.zhi@uth.tmc.edu

Recent advancements in genome-wide SNP array and whole genome sequencing technologies have led to the generation of enormous amounts of population genotype data. Understanding the genetic relationships based on individuals' genotypes will shed light on better insight into precision medicine or population genetics. A basic measure of genetic relationship is Identity by Descent (IBD). IBD is defined as chromosomal segments shared between two individual chromosomes which have been inherited from a common ancestor.

Most of the existing methods for IBD detection, such as IBDseq [1], PLINK [4], and PARENTE [5], can handle both genotype and haplotype data, but they rely on pairwise comparison of all individuals and therefore are not scalable for large number of individuals. GERMLINE [3] avoids pairwise comparison by using hash table on haplotype sequences. Under the assumption that the number of seed matches between any individual and others is constant, the complexity of GERMLINE will grow linearly with the number of samples. However, the individuals in a population share different levels of common substructures, therefore the seed matches may deviate from its idealized linear behavior in a sample with a large number of individuals. As a result, it will not be fast enough for hundreds of thousands of individuals and millions of variant sites.

In this work, we present an efficient computational method, named RaPID (Random Projection for IBD Detection), to find IBD segments larger than a given length in haplotype data. We use an efficient population genotype index, Positional Burrows-Wheeler Transform (PBWT) [2], that scales up linearly with the sample size. PBWT algorithm can compute all haplotype matches that exceed a given length in $O(\max(MN, I))$, where M denotes the number of individuals, N the number of variant sites and I the number of matches. The key idea behind PBWT is to sort the sequences by their reversed prefix at each position. The PBWT algorithm sweeps through the list of variant sites and keeps the starting position of each match between neighboring prefixes. PBWT searches for exact matches and cannot tolerate mismatches that might be due to genotyping

or phasing errors. In order to account for genotyping or phasing errors, we build PBWT over random projections of the original sequences. We divide the panel into non-overlapping windows with the same length. The length is defined in terms of consecutive variant sites. For each window, we select a variant site at random and find all exact matches that exceed a minimum length using PBWT. We repeat the random projection PBWT multiple times to increase the detection power. Since the error rate is presumably low, a true IBD segment will have a high probability to be identified in some of the multiple runs. On the other hand, a non-IBD segment will have a lower probability to be selected in multiple runs. We model these probabilities as binomial distributions. A matched segment between any two individuals is considered to be an IBD if it was selected more than a certain number of times among a total number of PBWT runs.

To evaluate the performance of RaPID, we have computed the accuracy and power in a simulated population and compared the results with GERMLINE and IBDseq. Accuracy is defined as the percentage of the correctly detected IBD segments which overlap at least 50% with a true IBD. Power is defined as the average of detected proportions of true IBDs. RaPID maintains comparable accuracy and power to GERMLINE and IBDseq while being orders of magnitudes faster than GERMLINE. On our simulated data, the running time of RaPID was more than 100 times faster than GERMLINE when searching for IBDs with a minimum length of 3 cM. Therefore, RaPID would be an appropriate tool for IBD detection in very large Biobank-scale genotyped cohorts. To demonstrate the utility of RaPID for real data, we have applied it on the 1000 Genome Project data. The results show that RaPID can detect population events at different time scales. Implementation is available at <https://github.com/ZhiGroup/RaPID>.

References

1. Browning, B.L., Browning, S.R.: Detecting identity by descent and estimating genotype error rates in sequence data. *Am. J. Hum. Genet.* **93**(5), 840–851 (2013)
2. Durbin, R.: Efficient haplotype matching and storage using the positional Burrows-Wheeler transform (PBWT). *Bioinformatics* **30**(9), 1266–1272 (2014)
3. Gusev, A., Lowe, J.K., Stoffel, M., Daly, M.J., Altshuler, D., Breslow, J.L., Friedman, J.M., Pe'er, I.: Whole population, genome-wide mapping of hidden relatedness. *Genome Res.* **19**(2), 318–326 (2009)
4. Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M.A., Bender, D., Maller, J., Sklar, P., de Bakker, P.I., Daly, M.J., Sham, P.C.: PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am. J. Hum. Genet.* **81**(3), 559–575 (2007)
5. Rodriguez, J.M., Bercovici, S., Huang, L., Frostig, R., Batzoglu, S.: Parente2: a fast and accurate method for detecting identity by descent. *Genome Res.* **25**(2), 280–289 (2015)

Joker de Bruijn: Sequence Libraries to Cover All k -mers Using Joker Characters

Yaron Orenstein¹, Ryan Kim², Polly Fordyce³, and Bonnie Berger¹

¹ Massachusetts Institute of Technology, Cambridge, MA 02139, USA
bab@mit.edu

² Research Science Institute, McLean, VA 22207, USA

³ Stanford University, Stanford, CA 94305, USA

1 Introduction

Protein-DNA, -RNA and -peptide interactions drive nearly all cellular processes. Due to their high importance, high-throughput technologies using sequence libraries that cover all k -mers (i.e. words of length k) have been developed to measure them in a universal and unbiased manner [1]. These techniques all face a similar challenge: the space on the experimental device is limited, restricting the total sequence space that can be probed in a single experiment. While de Bruijn sequences cover all k -mers in the most compact manner, they remain $|\Sigma|^k$ characters long (where Σ is the alphabet, e.g. $\{A,C,G,T\}$). Here, we introduce a novel idea and algorithm for sequence design to cover all possible k -mers with a significantly smaller experimental sequence library by using *joker* characters, which represent all characters in the alphabet. Experimentally, such joker characters can be easily incorporated during oligonucleotide or peptide synthesis by using degenerate mixtures of nucleotides or amino acids, at no extra cost. However, joker characters introduce degeneracy which could potentially lower the statistical robustness of the measurements (as a measurement of a single oligonucleotide is now assigned to multiple sequences instead of just one). To address this challenge, we limit the use of joker characters to either one or two joker characters per k -mer, enabling the coverage of $(k+2)$ -mers at the same cost and space of k -mers — a savings of a factor of $|\Sigma|^2$ in sequence length (16 and 400 for DNA and amino acid alphabets, respectively). We validate that the library remains capable of *de novo* identification of high-affinity k -mers by testing it on known DNA-protein binding data for hundreds of proteins. The implementation of our algorithm is freely available at jokercake.csail.mit.edu.

2 Methods

We propose a novel solution to the problem of generating a short sequence covering all k -mers using joker characters. The solution is based on two steps: (i) a greedy heuristic; (ii) and an ILP formulation. The greedy heuristic examines at each step an addition of $k - 1$ characters from Σ followed by a joker character. The addition that covers the most k -mers that are yet to be covered p times

is chosen and added to the current sequence. The algorithm terminates when all k -mers have been covered at least p times. The ILP formulation minimizes the number of k -mers in the sequence under two sets of constraints. The first requires that all k -mers occur at least p times. The second guarantees that the k -mer occurrences can form a sequence. The ILP is solved using Gurobi ILP solver version 6.5.2 [2], where it is given the greedy solution as a starting solution.

3 Results

To test the performance of our algorithm, we ran it on different parameter combinations. We ran the greedy heuristic on $5 \leq k \leq 8$ for a DNA alphabet and $3 \leq k \leq 4$ for an amino acid alphabet, with $p = 1$. We then ran the ILP solver, starting from the greedy solution, with a time limit of 4 weeks. Results show that the greedy algorithm produces a sequence that is much smaller than the original de Bruijn sequence; i.e., less than 40% and 8% of the original for DNA and amino acid alphabets, respectively. Following the ILP solver, sequence length drops even further to less than 33% and 8% of the original, respectively, where the theoretical lower bounds are 25% and 5%, respectively. To test the performance of our algorithm in covering k -mers multiple times, we ran the greedy heuristic on $k = 6$, a DNA alphabet, and $1 \leq p \leq 16$. Here, we see that the greedy algorithm is producing a near-optimal sequence, less than 27% of the size of the original de Bruijn sequence for $p \geq 4$.

To demonstrate the utility of these libraries, we validated our performance when tested against a standard experimental 10-mer library of nearly 42,000 DNA sequences for which the binding affinities of hundreds of transcription factors is known [3]. Remarkably, our library correctly recovers the high-affinity target sites, despite a nearly 4-fold reduction in library size. We were able to handle 10-mer libraries due to a 100-fold speedup in implementation over a naive one for our joker library design.

4 Conclusion

We presented a new library design that covers all k -mers with a library of size that is almost $1/|\Sigma|$ (and possibly $1/|\Sigma|^2$) smaller than current libraries, making it possible to measure interactions of significantly longer k -mers while reducing both experimental footprint and cost. We have made the implementation and library designs freely available to others.

References

1. Fordyce, P.M., Gerber, D., Tran, D., Zheng, J., Li, H., DeRisi, J.L., Quake, S.R.: De novo identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. *Nat. Biotechnol.* **28**(9), 970–975 (2010)
2. Gurobi Optimization, I.: Gurobi Optimizer Reference Manual (2015). <http://www.gurobi.com>
3. Hume, M.A., Barrera, L.A., Gisselbrecht, S.S., Bulyk, M.L.: UniPROBE, update 2015: new tools and content for the online database of protein-binding microarray data on protein-DNA interactions. *Nucleic Acids Res.* gku1045 (2014)

GATTACA: Lightweight Metagenomic Binning Using Kmer Counting

Victoria Popic¹, Volodymyr Kuleshov¹, Michael Snyder²,
and Serafim Batzoglou¹(✉)

¹ Department of Computer Science, Stanford University, Stanford, CA, USA
{viq,kuleshov,serafim}@stanford.edu

² Department of Genetics, Stanford University, Stanford, CA, USA
mpsnyder@stanford.edu

Extended Abstract

Despite their important role, microbes constitute the dark matter of the biological universe. The main limitation hindering their study is sequencing technology. The short read lengths of modern instruments – combined with various inherent difficulties associated with complex bacterial environments – make it very difficult to perform simple tasks such as accurately identifying bacterial strains, recovering their genomic sequences, and assessing their abundance. Many approaches have been proposed to address these shortcomings. Specialized library preparation techniques such as Hi-C or synthetic long reads are often very accurate, but also prohibitively complex. As a result, approaches based on contig binning are more popular in practice.

Metagenomic binning refers to the problem of grouping together partially assembled sequence fragments (or contigs) that belong to the same species. The most successful recent approaches [1, 3, 5] perform unsupervised clustering based on contig sequence composition and coverage profiles across multiple metagenomic samples. In brief, these techniques assemble de-novo bacterial contigs and estimate the coverage of each contig within each sample of a large metagenomic cohort using read mapping. This approach is accurate but has two main limitations: it requires a large cohort of samples, as well as sizable compute resources for read alignment.

In this work we present GATTACA, a lightweight framework for metagenomic binning, which (1) avoids read alignment without loss of accuracy and (2) enables efficient stand-alone analysis of single metagenomic samples. Both results are based on the finding that we can approximate contig coverages using kmer counts while still achieving the same binning accuracy as leading alignment-based methods. In addition to offering a significant speedup in coverage estimation, using kmer counts, as opposed to alignment, provides us with the exciting ability to index *offline* any publicly-available metagenomic sample. This allows us to efficiently pull in data from large growing repositories, such as the Human Microbiome Project (HMP) [4] or the EBI Metagenomics archive [2] into any metagenomic study at almost no cost. For example, our kmer count index for a typical HMP sample only requires 100MB on average. We achieve the small

space requirement by leveraging memory-efficient hashing with minimal perfect hash functions (MPHF) and the probabilistic Bloom filter data structure. In contrast, using these datasets with read alignment would require massive downloads and expensive subsequent handling to map the reads. In terms of speedup, we found our coverage estimation time to be at least an order of magnitude faster (approximately $20\times$) when the index is computed offline (e.g. for recyclable public reference samples) and about $6\times$ when the kmers are counted on-the-fly (e.g. for private samples used only once), when compared to read mapping.

While using small indices allows us to incorporate a large number of publicly-available samples into a given study, not all existing samples will improve the binning accuracy. Therefore, we propose the following two metrics for sample selection: (1) relevance and (2) diversity. More specifically, we would like to select a panel of samples which share content with the sample being analyzed (our query) but that also differ in the content that is shared. We use locality sensitive hashing and the MinHash technique, to compare the samples efficiently. At a high level, we create and index small MinHash fingerprints for each sample in the database (offline), and then extract the appropriate samples according to the fingerprint of the query.

We evaluate GATTACA for clustering contigs assembled across multiple samples (co-assemblies) and from individual samples, using both synthetic and real datasets. We compare our method to several leading alignment-based methods for metagenomic binning (such as CONCOCT [1], MetaBAT [3], and MaxBin [5]), using standardized cluster evaluation metrics and benchmarks. GATTACA was implemented in C++ and Python and is freely available at <http://viq854.github.com/gattaca>.

References

1. Alneberg, J., Bjarnason, B.S., de Bruijn, I., Schirmer, M., Quick, J., Ijaz, U.Z., Lahti, L., Loman, N.J., Andersson, A.F., Quince, C.: Binning metagenomic contigs by coverage and composition. *Nat. Methods* **11**(11), 1144–1146 (2014)
2. Hunter, S., Corbett, M., Denise, H., Fraser, M., Gonzalez-Beltran, A., Hunter, C., Jones, P., Leinonen, R., McAnulla, C., Maguire, E., et al.: Ebi metagenomics—a new resource for the analysis and archiving of metagenomic data. *Nucleic Acids Res.* **42**(D1), D600–D606 (2014)
3. Kang, D.D., Froula, J., Egan, R., Wang, Z.: Metabat, an efficient tool for accurately reconstructing single genomes from complex microbial communities. *PeerJ* **3**, e1165 (2015)
4. Turnbaugh, P.J., Ley, R.E., Hamady, M., Fraser-Liggett, C., Knight, R., Gordon, J.I.: The human microbiome project: exploring the microbial part of ourselves in a changing world. *Nature* **449**(7164), 804 (2007)
5. Wu, Y.W., Tang, Y.H., Tringe, S.G., Simmons, B.A., Singer, S.W.: Maxbin: an automated binning method to recover individual genomes from metagenomes using an expectation-maximization algorithm. *Microbiome* **2**(1), 26 (2014)

Species Tree Estimation Using ASTRAL: How Many Genes Are Enough?

Shubhanshu Shekhar¹, Sebastien Roch², and Siavash Mirarab¹(✉)

¹ University of California, La Jolla, San Diego, CA 92093, USA
smirarab@ucsd.edu

² University of Wisconsin-Madison, Madison, WI 53715, USA

Abstract. ASTRAL is a widely used method for reconstructing species trees from unrooted gene tree data. In this paper, we derive bounds on the number of gene trees needed by ASTRAL for reconstructing the true species tree with high probability. We also present some simulation results which show trends consistent with our theoretical bounds.

Keywords: Species tree estimation · Sample complexity · ASTRAL

1 Introduction

Evolutionary histories of genes and species can be discordant due to various biological processes such as incomplete lineage sorting (ILS) [1, 2]. One way to account for the discordance is to first estimate a phylogenetic tree for each gene (a gene tree) and then to summarize them to get a species tree.

ASTRAL [3] is a widely-used summary method for species tree reconstruction, and is statistically consistent under the multi-species coalescent (MSC) model [2] of ILS. ASTRAL uses dynamic programming to maximize the number of induced quartet trees shared between the species tree and the set of input gene trees, and has exact and heuristic versions. In this paper, we study ASTRAL's theoretical data requirements for successful species tree reconstruction with high probability under the MSC model and provide matching simulations results.

2 Main Results

In these results, ASTRAL* refers to the exact version of ASTRAL, f is the length of the shortest branch in the species tree in coalescent units [2], n denotes the number of leaves in the species tree and m is the number of input gene trees.

Our first result says that for small values of f , $m = \Omega(f^{-2} \log n)$ gene trees are sufficient for the correct species tree reconstruction by ASTRAL*.

The rights of this work are transferred to the extent transferable according to title 17 § 105 U.S.C.

Theorem 1. Consider a model species tree with minimum branch length f . Then, in the limit of small f and for any $\epsilon > 0$, *ASTRAL** returns the true species tree with probability at least $1 - \epsilon$ if the number of input error-free gene trees satisfies

$$m > 20 \log \left(\frac{n}{6\epsilon} \right) \frac{1}{f^2}. \tag{1}$$

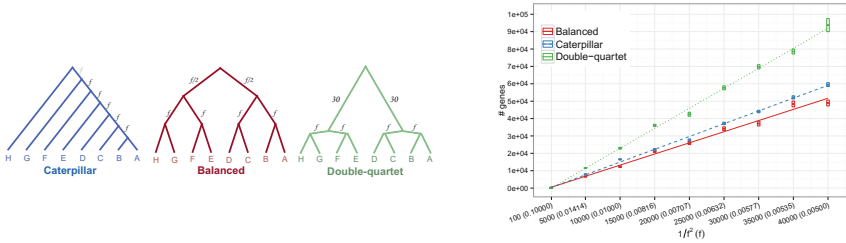


Fig. 1. Data requirement of *ASTRAL-II* and in simulations with $\epsilon = 0.1$. For each of the three different species tree shapes (left) and values of f (right panel; x axis), 401 replicate datasets are simulated using the MSC model, each with up to 10^5 gene trees. A binary search is used to find an approximate range for the smallest number of genes with which *ASTRAL* recovers the correct tree in at least 90% of the 401 replicates. Boxes show these ranges and a line is fitted to midpoints.

Our next result establishes that there exist species trees with lower bounds of error that are asymptotically similar to our upper bounds.

Theorem 2. For any $\rho \in (0, 1)$ and $a \in (0, 1)$, there exist constants f_0 and n_0 such that the following holds. For all $n \geq n_0$ and $f \leq f_0$, there exists a species tree with n leaves and shortest branch length f such that when *ASTRAL** is used with $m \leq \frac{a \log n}{5 f^2}$ gene trees, the event E that *ASTRAL** reconstructs the wrong tree has probability

$$P(E) \geq 1 - \rho. \tag{2}$$

These two results imply that *ASTRAL** requires $\Omega(f^{-2} \log n)$ gene trees to universally guarantee correct species tree reconstruction with high probability.

Simulation results: As expected by our theoretical results, the gene tree requirement of *ASTRAL* increases linearly with $1/f^2$ in a simulation study (Fig. 1).

Acknowledgment. The work was supported by National Science Foundation (NSF) grant IIS-1565862 to SM and SS and NSF grant DMS-1149312 (CAREER) to SR.

References

1. Maddison, W.P.: Gene trees in species trees. *Syst. Biol.* **46**(3), 523–536 (1997)
2. Degnan, J.H., Rosenberg, N.A.: Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends Ecol. Evol.* **24**(6), 332–340 (2009)
3. Mirarab, S., Warnow, T.: ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics* **31**(12), i44–i52 (2015)

Reconstructing Antibody Repertoires from Error-Prone Immunosequencing Datasets

Alexander Shlemov¹, Sergey Bankevich¹, Andrey Bzikadze¹,
Yana Safonova^{1(✉)}, and Pavel A. Pevzner^{1,2}

¹ Center for Algorithmic Biotechnology, Institute of Translational Biomedicine,
St. Petersburg State University, Saint Petersburg, Russia

safonova.yana@gmail.com

² Dept. of Computer Science and Engineering, University of California,
San Diego, La Jolla, CA, USA

1 Introduction

Recent progress in sequencing technologies enabled generation of high-throughput full-length antibody sequences using read-pairs formed by overlapping reads within read-pairs. However, transforming error-prone Rep-seq datasets into accurate *antibody repertoires* is a challenging bioinformatics problem [1, 2, 4] that is a prerequisite for a multitude of downstream studies of adaptive immune system.

Until 2013, there were few attempts to develop algorithms for full-length antibody repertoire reconstruction since it was unclear how to derive accurate repertoires from error-prone reads produced by the low-throughput 454 sequencing technology. However, in the last three years, immunology laboratories developed various Rep-seq protocols aimed at generating high-throughput Rep-seq datasets and constructing repertoires based on more accurate Illumina MiSeq reads.

Recently, several tools for constructing full-length antibody repertoires were developed, including MIGEC [3], PRESTO [4], MIXCR [1], and IGREP-TOIRECONSTRUCTOR [2]. Some of these tools also are able to utilize information about molecular barcodes. However, quality assessment of the constructed antibody repertoire and, thus, benchmarking of various repertoire construction algorithms are still poorly addressed problems.

In this paper, we use barcoded Rep-seq datasets and simulated antibody repertoires to benchmark various repertoire construction algorithms. Our novel toolkit includes IGREC, a tool for antibody repertoire construction from both barcoded and non-barcoded immunosequencing data, and IGQUAST, a tool for quality assessment of antibody repertoires. IGREC package is freely available at <http://yana-safonova.github.io/ig-repertoire-constructor>.

Alexander Shlemov and Sergey Bankevich—These authors contributed equally.

2 Discussion

Our benchmarking on non-barcoded data revealed that there is still no single repertoire construction tool that works better than others across the diverse types of Rep-seq datasets. However, IGREC is currently a tool of choice for analyzing hypermutated repertoires.

We also compared IGREC in a blind mode against PRESTO and MiGEC that utilize information about molecular barcodes. Benchmarking on simulated barcoded datasets revealed that while all tools result in high sensitivity, their precision varies and becomes rather low in the case of high PCR error rates. Surprisingly, repertoires reported by IGREC tool (in the blind mode) improved on the repertoires constructed by the specialized tools that use barcoding information.

Acknowledgements. We are indebted to Dmitry Chudakov, Dmitry Bolotin, Mikhail Shugay, Jason Vander Heiden, and Steven Kleinstein for productive discussions and assistance in benchmarking MiXCR, MiGEC, and pRESTO tools.

Funding. This project is supported by Russian Science Foundation (grant No 14-50-00069).

References

1. Bolotin, D.A., Poslavsky, S., Mitrophanov, I., Shugay, M., Mamedov, I.Z., Putintseva, E.V., Chudakov, D.M.: MiXCR: software for comprehensive adaptive immunity proling. *Nat. Methods* **12**(5), 3801 (2015)
2. Safonova, Y., Bonissone, S., Kurpilyansky, E., Starostina, E., Lapidus, A., Stinson, J., DePalatis, L., Sandoval, W., Lill, J., Pevzner, P.A.: IgRepertoireConstructor: a novel algorithm for antibody repertoire construction and immunoproteogenomics analysis. *Bioinformatics* **31**(12), i53–61 (2015)
3. Shugay, M., Britanova, O., Merzlyak, E., Turchaninova, M., Mamedov, I., Tuganbaev, T., Bolotin, D., Staroverov, D., Putintseva, E., Plevova, K., Linnemann, C., Shagin, D., Pospisilova, S., Lukyanov, S., Schumacher, T., Chudakov, D.M.: Towards error-free proling of immune repertoires. *Nat Methods* **11**, 6535 (2014)
4. Vander Heiden, J.A., Yaari, G., Uduman, M., Stern, J.N., O'Connor, K.C., Haer, D.A., Vigneault, F., Kleinstein, S.H.: pRESTO: a toolkit for processing high-throughput sequencing raw reads of lymphocyte receptor repertoires. *Bioinformatics* **30**(13), 1930–1932 (2014)

NetREX: Network Rewiring Using EXpression - Towards Context Specific Regulatory Networks

Yijie Wang¹, Dong-Yeon Cho¹, Hangnoh Lee², Brian Oliver²,
and Teresa M. Przytycka¹(✉)

¹ National Center of Biotechnology Information, National Library of Medicine, NIH,
Bethesda, MD 20894, USA
przytyck@ncbi.nlm.nih.gov

² Laboratory of Cellular and Developmental Biology, National Institute of Diabetes
and Digestive and Kidney Diseases, 50 South Drive,
Bethesda, MD 20892, USA

Extended Abstract

Understanding gene regulation is a fundamental step towards understanding of how cells function and respond to environmental cues and perturbations. An important step in this direction is the ability to infer the transcription factor (TF)-gene regulatory network (GRN). However gene regulatory networks are typically constructed disregarding the fact that regulatory programs are conditioned on tissue type, developmental stage, sex, and other factors. Due to lack of the biological context specificity, these context-agnostic networks may not provide insight for revealing the precise actions of genes for a specific biological system under concern. Collecting multitude of features required for a reliable construction of GRNs such as physical features (TF binding, chromatin accessibility) and functional features (correlation of expression or chromatin patterns) for every context of interest is costly. Therefore we need methods that are able to utilize the knowledge about a context-agnostic network (or a network constructed in a related context) for construction of a context specific regulatory network.

To address this challenge we developed a computational approach that utilizes expression data obtained in a specific biological context and a GRN constructed in a different but related context to construct a context specific GRN. Our method, NetREX, is inspired by network component analysis that estimates TF activities and their influences on target genes given predetermined topology of a TF-gene regulatory network. To predict a network under a different condition, NetREX removes the restriction that the topology of the TF-gene regulatory network is fixed and allows for adding and removing edges to that network. Mathematically, we use ℓ_0 norm to directly handle the number of removed and newly added edges as well as induce sparse solutions in our formulation.

Yijie Wang and Dong-Yeon Cho — Equal contribution.

The rights of this work are transferred to the extent transferable according to title 17 §105 U.S.C.

Unlike the widely used strategy, which is replacing the non-convex ℓ_0 norm by its convex relaxation ℓ_1 norm, we focus on the harder problem involving ℓ_0 norm and provide a number of rigorous derivations and results allowing us to adopt the recently proposed Proximal Alternative Linearized Maximization (PALM) algorithm. In addition, we also proved the convergence of the NetREX algorithm.

We tested our NetREX on simulated data and found that NetREX is able to dramatically improve the accuracy of the regulatory networks as long as the prior network and the gene expression are not very noisy. Subsequently, we applied NetREX for constructing regulatory networks for adult female flies. We used the network constructed in a recent study as the prior network, which was build by integrating diverse data sets including TF binding, evolutionarily conserved sequence motifs and so on. Starting with this network, we utilized a new expression data set that we collected for adult female flies where perturbations in expression were achieved by genetic deletions. We accessed the biological relevance of the predicted networks by using Gene Ontology annotations and physical protein-protein interactions. The networks predicted by NetREX showed higher biological consistency than alternative approaches. In addition, we used the list of recently identified targets of the Doublesex (DSX) transcription factor to demonstrate the predictive power of our method.

E Pluribus Unum: United States of Single Cells

Joshua D. Welch¹(✉), Alexander Hartemink², and Jan F. Prins¹

¹ Department of Computer Science, The University of North Carolina,
Chapel Hill, USA

{jwelch,prins}@cs.unc.edu

² Department of Computer Science, Duke University, Durham, USA

Extended Abstract

Single cell genomic techniques promise to yield key insights into the dynamic interplay between gene expression and epigenetic modification. However, the experimental difficulty of performing multiple measurements on the same cell currently limits efforts to combine multiple genomic data sets into a united picture of single cell variation [1, 2]. The current understanding of epigenetic regulation suggests that any large changes in gene expression, such as those that occur during differentiation, are accompanied by epigenetic changes. This means that if cells undergoing a common process are sequenced using multiple genomic techniques, examining any of the genomic quantities should reveal the same underlying biological process. For example, the main difference among cells undergoing differentiation will be the extent of their differentiation progress, whether you look at the gene expression profiles or the chromatin accessibility profiles of the cells.

We reasoned that this property of single cell data could be used to infer correspondence between different types of genomic data. To infer single cell correspondences, we use a technique called manifold alignment. Intuitively, manifold alignment constructs a low-dimensional representation (manifold) for each of the observed data types, then projects these representations into a common space (alignment) in which measurements of different types are directly comparable [3, 4]. To the best of our knowledge, manifold alignment has never been used in genomics. However, other application areas recognize the technique as a powerful tool for multimodal data fusion, such as retrieving images based on a text description, and multilingual search without direct translation [4].

We show for the first time that it is possible to construct cell trajectories, reflecting the changes that occur in a sequential biological process, from single cell epigenetic data. In addition, we present an approach called MATCHER that computationally circumvents the experimental difficulties of performing multiple genomic measurements on a single cell by inferring correspondence between single cell transcriptomic and epigenetic measurements performed on different cells of the same type. MATCHER works by first learning a separate manifold for the trajectory of each kind of genomic data, then aligning the manifolds to infer a shared trajectory in which cells measured using different techniques are directly comparable. Because there is, in general, no actual cell-to-cell correspondence

between datasets measured with different experimental techniques, MATCHER *generates* corresponding measurements by predicting what each type of measurement *would* look like at a given point in the process. Using scM&T-seq data, we confirm that MATCHER accurately predicts true single cell correlations between DNA methylation and gene expression without using known cell correspondence information.

We also downloaded publicly available single cell genomic data from a total of 4,974 single mouse embryonic stem cells grown in serum. Each cell in this dataset was individually assayed using one of four experimental techniques: RNA-seq, scM&T-seq, ATAC-seq, or ChIP-seq. We used MATCHER to infer correlations among these four measurements. This analysis gave novel insights into the changes that cells undergo as they transition from pluripotency to a differentiation primed state.

We found three main results. First, chromatin accessibility and histone modification changes largely fall into two anti-correlated categories: silencing of pluripotency factor binding sites and repression of lineage-specific genes by chromatin remodeling factors. Second, the action of pluripotency transcription factors is gradually removed by both transcriptional silencing of the genes and epigenetic silencing of the binding sites for these factors. In contrast, regulation of chromatin remodeling factor activity occurs primarily at the epigenetic level, largely unaccompanied by changes in the expression of the chromatin remodeling factors. Third, DNA methylation changes are strongly coupled to gene expression changes early in the process of differentiation priming, but the degree of coupling drops sharply later in the process.

Our work is a first step toward a united picture of heterogeneous transcriptional and epigenetic states in single cells. MATCHER promises to be a powerful tool as single cell genomic approaches continue to generate revolutionary discoveries in fields ranging from cancer biology and regenerative medicine to developmental biology and neuroscience.

References

1. Bock, C., Farlik, M., Sheffield, N.C.: Multi-omics of single cells: strategies and applications. *Trends Biotechnol.* **34**(8), 605-608 (2016)
2. Macaulay, I.C., Ponting, C.P., Voet, T.: Single-cell multiomics: multiple measurements from single cells. *Trends Genet.* **33**(2), 155-168 (2017)
3. Ham, J., Lee, D.D., Saul, L.K.: Semisupervised alignment of manifolds. In: AIS-TATS, p. 120127 (2005)
4. Wang, C., Mahadevan, S.: A general framework for manifold alignment. In: AAAI (2009)

ROSE: A Deep Learning Based Framework for Predicting Ribosome Stalling

Sai Zhang¹, Hailin Hu², Jingtian Zhou², Xuan He¹, Tao Jiang^{3,4,5},
and Jianyang Zeng¹(✉)

¹ Institute for Interdisciplinary Information Sciences, Tsinghua University,
Beijing, China

zengjy321@tsinghua.edu.cn

² School of Medicine, Tsinghua University, Beijing, China

³ Department of Computer Science and Engineering, University of California,
Riverside, CA, USA

⁴ MOE Key Lab of Bioinformatics and Bioinformatics Division,
TNLIST/Department of Computer Science and Technology,
Tsinghua University, Beijing, China

⁵ Institute of Integrative Genome Biology, University of California,
Riverside, CA, USA

Abstract. Translation elongation plays a crucial role in multiple aspects of protein biogenesis, e.g., differential expression, cotranslational folding and secretion. However, our current understanding on the regulatory mechanisms underlying translation elongation dynamics and the functional roles of ribosome stalling in protein synthesis still remains largely limited. Here, we present a deep learning based framework, called ROSE, to effectively predict ribosome stalling events in translation elongation from coding sequences. Our validation results on both human and yeast datasets demonstrate superior performance of ROSE over conventional prediction models. With high prediction accuracy and robustness across different datasets, ROSE shall provide an effective index to estimate the translational pause tendency at codon resolution. We also show that the ribosome stalling score (RSS) output by ROSE correlates with diverse putative regulatory factors of ribosome stalling, e.g., codon usage bias, codon cooccurrence bias, proline codons and N⁶-methyladenosine (m⁶A) modification, which validates the physiological relevance of our approach. In addition, our comprehensive genome-wide *in silico* studies of ribosome stalling based on ROSE recover several notable functional interplays between elongation dynamics and cotranslational events in protein biogenesis, including protein targeting by the signal recognition particle (SRP) and protein secondary structure formation. Furthermore, our intergenic analysis suggests that the enriched ribosome stalling events at the 5' ends of coding sequences may be involved in the modulation of translation efficiency. These findings indicate that ROSE can provide a useful index to estimate the probability of ribosome stalling and offer a powerful tool to analyze the large-scale ribosome profiling data, which will further expand our understanding on translation elongation dynamics. The full version of this work can be found as a preprint at <https://doi.org/10.1101/067108>.

S. Zhang, H. Hu, J. Zhou — These authors contributed equally to this work.

© Springer International Publishing AG 2017

S.C. Sahinalp (Ed.): RECOMB 2017, LNBI 10229, pp. 402–403, 2017.

DOI: 10.1007/978-3-319-56970-3

Acknowledgements. This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300 and 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003 and 61472205, the US National Science Foundation Grant DBI-1262107 and IIS-1646333, the China's Youth 1000-Talent Program, and the Beijing Advanced Innovation Center for Structural Biology.

Author Index

- Ahn, Soyeon 353
Allen, Andrew S. 336
Aluru, Srinivas 66
- Bankevich, Sergey 396
Batzoglou, Serafim 391
Berger, Bonnie 389
Bzikadze, Andrey 396
- Chaisson, Mark J. 117
Chen, Ligong 383
Chikhi, Rayan 272
Cho, Dong-Yeon 398
- Dao, Phuong 370
DeBlasio, Dan 1
DeLong, Andrew 379
Dilthey, Alexander 66
Donald, Bruce R. 157
- Ehrhardt, Marcel 190
Eichler, Evan E. 117
El-Kebir, Mohammed 318
Eskin, Eleazar 207, 303
- Fisher, Eyal 241
Fordyce, Polly 389
Fowler, Vance G. 157
Frey, Brendan J. 379
- Garrison, Erik 173
Gifford, David K. 372
Gordán, Raluca 336
Guo, Yuchun 372
- Hach, Faraz 50
Halperin, Eran 207, 241
Han, Wontack 18
Harris, Robert S. 272
Hartemink, Alexander 400
Haussler, David 34
He, Xuan 402
Hickey, Glenn 173
Holley, Guillaume 50
Hormozdiari, Farhad 303
- Hormozdiari, Fereydoun 377
Hristov, Borislav H. 375
Huang, Heng 287
Hu, Hailin 402
Huynh, Linh 377
- Jain, Chirag 66
Jansson, Jesper 82
Jiang, Tao 402
Joo, Jong Wha J. 303
Jou, Jonathan D. 157
- Kannan, Sreeram 117
Kececioglu, John 1
Keich, Uri 99
Kim, Ryan 389
Kim, Sungeun 287
Kim, Yoo-Ah 370
Kingsford, Carl 257
Klau, Gunnar W. 318
Koren, Sergey 66
Kuang, Wenhua 383
Kuleshov, Volodymyr 391
- Lee, Hangnoh 398
Leung, Michael K.K. 379
Li, Dongshunyi 336
Li, Zhen 380
Lingas, Andrzej 82
Liu, Xiaoming 387
Luo, Yunan 383
- Ma, Jian 224
Madan, Sanna 370
Mallick, Parag 134
Matlak, Dariusz 385
Medvedev, Paul 272
Miagkov, Dmitrii 34
Mirarab, Siavash 393
Mukherjee, Sudipto 117
- Naseri, Ardalan 387
Ness, Robert Osazuwa 134
Nho, Kwangsik 287

- Nikitin, Sergei 34
Noble, William Stafford 99
Novak, Adam M. 34, 173
- Ojewole, Adegoke A. 157
Oliver, Brian 398
Orenstein, Yaron 389
- Paten, Benedict 34, 173
Peng, Jian 383
Pevzner, Pavel A. 396
Phillippy, Adam M. 66
Pockrandt, Christopher 190
Popic, Victoria 391
Prins, Jan F. 400
Przytycka, Teresa M. 370, 398
- Rahmani, Elior 207, 241
Rajaby, Ramesh 82
Rajaraman, Ashok 224
Raphael, Benjamin J. 318
Reinert, Knut 190
Risacher, Shannon L. 287
Roch, Sebastien 393
Rosset, Saharon 241
- Sachs, Karen 134
Safonova, Yana 396
Saykin, Andrew J. 287
Schweiger, Regev 207, 241
Seo, Jungkyun 336
Sharan, Roded 370
Shekhar, Shubhanshu 393
Shen, Li 287
Shenhav, Liat 207, 241
Shlemov, Alexander 396
Singh, Mona 375
Smuga-Otto, Maciej 34
Snyder, Michael 391
- Solomon, Brad 257
Stoye, Jens 50
Sun, Chen 272
Sung, Wing-Kin 82
Szczurek, Ewa 385
- Tian, Kevin 372
- Vikalo, Haris 353
Vitek, Olga 134
- Wang, Mingjie 18
Wang, Sheng 380
Wang, Xiaoqian 287
Wang, Yijie 398
Welch, Joshua D. 400
Wittler, Roland 50
Wu, Yue 303
- Xu, Jinbo 380
- Yan, Jingwen 287
Yang, Jinling 383
Yao, Xiaohui 287
Ye, Yuzhen 18
Yu, Yizhou 380
- Zaccaria, Simone 318
Zeng, Haoyang 372
Zeng, Jianyang 383, 402
Zhao, Jingkan 336
Zhao, Xinbin 383
Zhang, Sai 402
Zhang, Shaojie 387
Zhang, Yanqing 383
Zhi, Degui 387
Zhou, Jingtian 383, 402
Zueva, Maria 34