

Haixu Tang (Ed.)

LNBI 13976

# Research in Computational Molecular Biology

27th Annual International Conference, RECOMB 2023  
Istanbul, Turkey, April 16–19, 2023  
Proceedings



 Springer

MOREMEDIA



Lecture Notes in Computer Science

## Lecture Notes in Bioinformatics

13976

### 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*

### Editorial Board Members

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*

The series Lecture Notes in Bioinformatics (LNBI) was established in 2003 as a topical subseries of LNCS devoted to bioinformatics and computational biology.

The series publishes state-of-the-art research results at a high level. As with the LNCS mother series, the mission of the series is to serve the international R & D community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings.

Haixu Tang (Ed.)

# Research in Computational Molecular Biology

27th Annual International Conference, RECOMB 2023  
Istanbul, Turkey, April 16–19, 2023  
Proceedings



*Editor*  
Haixu Tang  
Indiana University Bloomington  
Bloomington, IN, USA

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Bioinformatics  
ISBN 978-3-031-29118-0              ISBN 978-3-031-29119-7 (eBook)  
<https://doi.org/10.1007/978-3-031-29119-7>

LNCS Sublibrary: SL8 – Bioinformatics

© The Editor(s) (if applicable) and The Author(s), under exclusive license  
to Springer Nature Switzerland AG 2023

Chapters “VStrains: De Novo Reconstruction of Viral Strains via Iterative Path Extraction From Assembly Graphs”, “Spectrum Preserving Tilings Enable Sparse and Modular Reference Indexing”, “Statistically Consistent Rooting of Species Trees Under the Multispecies Coalescent Model” and “Percolate: An Exponential Family JIVE Model to Design DNA-Based Predictors of Drug Response” are licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>). For further details see license information in the chapters.

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, expressed 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.

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

# Preface

This volume contains 11 extended abstracts and 33 short abstracts representing a total of 44 proceedings papers presented at the 27th International Conference on Research in Computational Molecular Biology (RECOMB 2023), which was hosted by Bilkent University, and took place at İstanbul, Turkey on April 16–19, 2023. These 44 contributions were selected using a rigorous peer-review process from 188 submissions to the conference. Each of the submissions received reviews from at least three members of the Program Committee (PC) or their designated sub-reviewers. After the initial review process, all submissions were opened for discussion by their reviewers and the conference program chair through the EasyChair Conference Management System. Final decisions were made based on the reviewers' assessments with some adjustment to ensure a broad coverage of bioinformatics research topics in the conference program.

RECOMB 2023 offered the authors the option to publish full extended papers in the conference proceedings, or to publish short abstracts in the proceedings while pursuing the publication of the full paper through a different venue. The authors of a subset of accepted papers were invited to submit revised manuscripts to be considered for publication in two partner journals, *Cell Systems* and *Genome Research*. All papers that appear as extended abstracts in the proceedings were invited for submission to the RECOMB 2023 special issue in the *Journal of Computational Biology*.

RECOMB 2023 also featured highlight talks of computational biology papers published in journals 18 months prior to the conference. Of the 60 submissions to the highlights track, eight were selected for oral presentation at RECOMB.

In addition to the presentations of these contributed papers, RECOMB 2023 featured six invited keynote talks given by leading scientists:

- İvet Bahar (Stony Brook University, USA, the EMBO Keynote Lecture)
- Richard Durbin (Wellcome Trust Sanger Institute and University of Cambridge, UK)
- Tuuli Lappalainen (New York Genome Center, USA; KTH Royal Institute of Technology; and SciLifeLab, Sweden)
- Fabian Theis (Helmholtz Munich, Germany and Wellcome Trust Sanger Institute, UK)
- Ewan Birney (European Bioinformatics Institute, UK)
- Sohini Ramachandran (Brown University, USA)

In addition, four RECOMB satellite meetings took place in parallel directly preceding the main RECOMB meeting, including:

- RECOMB-Seq 2023: The 13th RECOMB Satellite Conference on Biological Sequence Analysis
- RECOMB-CCB 2023: The 15th RECOMB Satellite Workshop on Computational Cancer Biology

- RECOMB-CG 2023: The 20th RECOMB Satellite Conference on Comparative Genomics
- RECOMB-Genetics 2023: The 11th RECOMB Satellite Workshop on Computational Methods in Genetics

The organization of this conference would not have been possible without the hard work of many colleagues who donated their time, efforts, and expertise. I am especially grateful to the local organizing committee: the Conference Co-chairs Can Alkan (Bilkent University) and Attila Gürsoy (Koç University), A. Ercüment Çiçek (Bilkent University) for publicity, Tunca Doğan (Hacettepe University) for registration, Arzucan Özgür (Boğaziçi University) for satellite meetings, and Öznur Taştan (Sabanci University) for publications, as well as the Student Organization Committee who volunteered their time and efforts. I am also grateful to many others who helped the organization of the conference but whose names were not yet known to us at the time of this writing. I want to thank the Poster Co-chairs, Iman Hajirasouliha (Weill Cornell Medicine) and Oğuzhan Külekçi (Istanbul Technical University), Keynotes Chair, Gamze Gürsoy (Columbia University), Highlights Chair, Ferhat Ay (La Jolla Institute for Immunology Satellites), and Travel Fellowship Co-chairs, Gürkan Bebek (Case Western Reserve University) and Arif Harmancı (University of Texas Health Science Center at Houston). I also want to thank the chairs of the Satellite meetings, including Co-chairs of RECOMB-Seq, Broňa Brejová (Comenius University in Bratislava) and A. Ercüment Çiçek (Bilkent University), Co-chairs of RECOMB-CG, Katharina Jahn (Freie Universität Berlin) and Tomáš Vinař (Comenius University in Bratislava), Co-chairs of RECOMB-Genetics, Emilia Huerta-Sanchez (Brown University), Nick Mancuso (University of Southern California) and Sriram Sankararaman (University of California, Los Angeles), and Co-chairs of RECOMB-CCB, Giulio Caravagna (University of Trieste) and Gabriele Schweikert (University of Dundee), for their great efforts in ensuring a high-quality technical programs of these satellite meetings. I am grateful to all of those PC members and sub-reviewers who took time to review and discuss submissions under a very tight schedule. Finally, I want to thank the keynote speakers and the authors of the proceedings papers, the highlight talks and the posters for presenting their work at the conference.

April 2023

Haixu Tang

# Organization

## General Chairs

Can Alkan (Co-chair)

Bilkent University, Turkey

Attila Gürsoy (Co-chair)

Koç University, Turkey

## Program Chair

Haixu Tang

Indiana University, USA

## Steering Committee

Vineet Bafna

University of California San Diego, USA

Bonnie Berger (Chair)

Massachusetts Institute of Technology, USA

Eleazar Eskin

University of California, Los Angeles, USA

Teresa Przytycka

National Institutes of Health, USA

Cenk Sahinalp

National Institutes of Health, USA

Roded Sharan

Tel Aviv University, Israel

Martin Vingron

Max Planck Institute for Molecular Genetics,  
Germany

## Organizing Committee

A. Ercüment Çiçek (Publicity  
Chair)

Bilkent University, Turkey

Tunca Doğan (Registration Chair)

Hacettepe University, Turkey

Arzucan Özgör (Satellites Chair)

Boğaziçi University, Turkey

Oznur Tastan (Publication Chair)

Sabancı University, Turkey

Iman Hajirasouliha (Poster  
Co-chair)

Weill Cornell Medicine, USA

Oğuzhan Külekçi (Poster  
Co-chair)

İstanbul Technical University, Turkey

Gamze Gürsoy (Keynote Chair)

Columbia University, USA

Ferhat Ay (Highlights Chair)

La Jolla Institute for Immunology, USA

Gürkan Bebek (Travel Fellowship Co-chair)	Case Western Reserve University, USA
Arif Harmancı (Travel Fellowship Co-chair)	University of Texas Health Science Center at Houston, USA

## Program Committee

Derek Aguiar	University of Connecticut, USA
Tatsuya Akutsu	Kyoto University, Japan
Srinivas Aluru	Georgia Institute of Technology, USA
Mukul S. Bansal	University of Connecticut, USA
Gürkan Bebek	Case Western Reserve University, USA
Niko Beerenwinkel	ETH Zurich, Switzerland
Bonnie Berger	Massachusetts Institute of Technology, USA
Valentina Boeva	ETH Zurich, Switzerland
Karsten Borgwardt	ETH Zurich, Switzerland
Christina Boucher	University of Florida, USA
Dongbo Bu	Chinese Academy of Sciences, China
Sebastian Böcker	Friedrich Schiller University Jena, Germany
Mark Chaisson	University of Southern California, USA
Cedric Chauve	Simon Fraser University, Canada
Brian Chen	Lehigh University, USA
A. Ercüment Çiçek	Bilkent University, Turkey
Lenore Cowen	Tufts University, USA
Tunca Doğan	Hacettepe University, Turkey
Mohammed El-Kebir	University of Illinois at Urbana-Champaign, USA
Nadia El-Mabrouk	University of Montreal, Canada
Travis Gagie	Diego Portales University, Chile
Xin Gao	King Abdullah University of Science and Technology, Saudi Arabia
Anthony Gitter	University of Wisconsin-Madison, USA
Boying Gong	University of California, Berkeley, USA
Çiğdem Gündüz Demir	Koç University, Turkey
Gamze Gürsoy	Columbia University, USA
Faraz Hach	University of British Columbia, Canada
Iman Hajirasouliha	Cornell University, USA
Bjarni Halldorsson	deCODE Genetics and Reykjavik University, Iceland
Arif Harmancı	University of Texas Health Sciences Center, USA
Farhad Hormozdiari	Google Health, USA
Fereydoun Hormozdiari	University California, Davis, USA

Lei Huang	Microsoft, USA
Tao Jiang	University of California, Riverside, USA
Emre Karakoç	Wellcome Sanger Institute, UK
Sündüz Keleş	University of Wisconsin-Madison, USA
Aly Khan	Toyota Technological Institute at Chicago, USA
Daisuke Kihara	Purdue University, USA
Gunnar W. Klau	Heinrich Heine University Düsseldorf, Germany
Tal Koream	Columbia University, USA
Mehmet Koyutürk	Case Western Reserve University, USA
Gregory Kucherov	CNRS/LIGM, France
Jens Lagergren	SBC and CSC, KTH, Sweden
Benjamin Langmead	Johns Hopkins University, USA
Heewook Lee	Arizona State University, USA
Jingyi Jessica Li	University of California, Los Angeles, USA
Stefano Lonardi	University of California Riverside, USA
Wenxiu Ma	University of California Riverside, USA
Jian Ma	Carnegie Mellon University, USA
Salem Malikic	National Cancer Institute, USA
Guillaume Marçais	Carnegie Mellon University, USA
Paul Medvedev	Pennsylvania State University, USA
Onur Mutlu	ETH Zurich, Switzerland
Veli Mäkinen	University of Helsinki, Finland
William Stafford Noble	University of Washington, USA
Ibrahim Numanagic	University of Victoria, Canada
Layla Oesper	Carleton College, USA
Yaron Orenstein	Ben-Gurion University, Israel
Robert Patro	University of Maryland, USA
Itzik Pe’Er	Columbia University, USA
Qian Peng	Scripps Research Institute, USA
Nadia Pisanti	University of Pisa, Italy
Cinzia Pizzi	University of Padova, Italy
Teresa Przytycka	National Center of Biotechnology Information, USA
Ben Raphael	Princeton University, USA
Knut Reinert	FU Berlin, Germany
Mina Rho	Hanyang University, South Korea
Sushmita Roy	University of Wisconsin-Madison, USA
Cenk Şahinalp	National Cancer Institute, USA
Sriram Sankararaman	University of California, Los Angeles, USA
Michael Schatz	Johns Hopkins University, USA
Alexander Schoenhuth	Bielefeld University, Germany
Russell Schwartz	Carnegie Mellon University, USA

Mingfu Shao	Penn State University, USA
Roded Sharan	Tel Aviv University, Israel
Ritambhara Singh	Brown University, USA
Rohit Singh	Massachusetts Institute of Technology, USA
Sagi Snir	University of Haifa, Israel
Jens Stoye	Bielefeld University, Germany
Fengzhu Sun	University of Southern California, USA
Wing-Kin Sung	National University of Singapore, Singapore
Krister Swenson	CNRS, Université de Montpellier, France
Haixu Tang	Indiana University Bloomington, USA
Öznur Taştan	Sabanci University, Turkey
Jonathan Terhorst	University of Michigan, USA
Tamir Tuller	Tel Aviv University, Israel
Fabio Vandin	University of Padova, Italy
Martin Vingron	Max Planck Institut fuer molekulare Genetik, Germany
Jerome Waldispuhl	McGill University, Canada
Wei Wang	University of California, Los Angeles, USA
Sheng Wang	University of Washington, USA
Yijie Wang	Indiana University, USA
Sebastian Will	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Yu-Wei Wu	Taipei Medical University, Taiwan
Min Xu	Carnegie Mellon University, USA
Berrin Yanikoglu	Sabanci University, Turkey
Yuzhen Ye	Indiana University Bloomington, USA
Yun William Yu	University of Toronto, Canada
Jianyang Zeng	Tsinghua University, China
Shaojie Zhang	University of Central Florida, USA
Louxin Zhang	National University of Singapore, Singapore
Chi Zhang	Indiana University, School of Medicine, USA
Xuhong Zhang	Indiana University, USA
Jie Zheng	ShanghaiTech University, China
Degui Zhi	University of Texas Health Science Center at Houston, USA

## Additional Reviewers

Jarno Alanko	Monica Dragan	Kassian Kobert
Mohammed Alser	David Dreifuss	Can Kockan
Bayarbaatar Amgalan	Yuxuan Du	Jack Lanchantin
Ulzee An	Dat Duong	Da-Inn Lee
Yoann Anselmetti	Arda Durmaz	Gang Li
Faisal Bin Ashraf	Gudmundur Einarsson	Mengzhen Li
Volkan Atalay	Prashant Emani	Xiang Li
Marzieh Ayati	Jason Fan	Xuan Li
Metin Balaban	Zhaoxin Fan	Yu Li
Sina Barazandeh	David Fernández-Baca	Chris Lin
Fritz Bayer	Can Firtina	Yen Yi Lin
J. White Bear	Michael Ford	Huiling Liu
Nicasia Beebe-Wang	Martin Frith	Jason Liu
Tim Beissbarth	Boyang Fu	Xinhao Liu
Philipp Benner	Julien Gagneur	Yangying Liu
Inanc Birol	Vianne Gao	Yuelin Liu
Richard Border	Yuan Gao	Mingyu Lu
Marilia Braga	Mathieu Gascon	Yang Lu
Matthew Brendel	Veronica Guerrini	Cong Ma
Mengfei Cao	Spencer Halberg	Christopher Magnano
Sakshar Chakravarty	Wenkai Han	Lauren Mak
Addie Chambers	Marteinn Hardarson	Frederick Matsen
Isaure Chauvot de Beauchene	Ananth Hari	Zachary McCaw
Ke Chen	Dongze He	Dmitrii Meleshko
Nae-Chyun Chen	Wenjia He	David Merrell
Tong Chen	Lance Hentges	Alan Min
Uthsav Chitra	Eran Hermush	Sourena Moghaddesi
Simone Ciccolella	Simon Ho	Amirsadra Mohseni
Matteo Comin	Ermin Hodzic	Ardalan Naseri
Pawel Czyn	Yuhui Hong	Rami Nasser
Raghuram D. R.	Borislav Hristov	Çerağ Oğuztüzün
Pengtao Dang	Jiawei Huang	Marco Oliva
Reut Danino	Linh Huynh	Carlos Oliver
Mitra Darvish	Pelin Burcak Icer	Baraa Orabi
Arun Das	Shani Jacobson	Alexey Orlov
James Degnan	Elham Jafari	Gunnar Pålsson
Mattéo Delabre	Anupama Jha	Fabio Pardi
Pinar Demetci	Suraj Joshi	Kwangmoon Park
Atul Deshpande	Zahra Zare Jousheghani	Tae Yoon Park
Diego Diaz	Gun Kaynar	Luca Parmigiani
Cansu Dincer	Parsoa Khorsand	Ali Pazokitoroudi
Jun Ding	Florian Klimm	Iftah Peretz
	Sara Knaack	Pierre Peterlongo



Giulio Ermanno Pibiri	Sean Simmons	Junyan Xu
Daniel Pirak	Gagandeep Singh	Xiaopeng Xu
Yuri Pirola	Noor Pratap Singh	Gurkan Yardimci
Wei Qiu	Pavel Skums	Tiantian Ye
Suraj Rajendran	Haris Smajlovic	Kaan Yorgancioglu
Srividya Ramakrishnan	Anna Spiro	Ronghui You
Vladimir Reinharz	Alexander Strzalkowski	Xiaofei Zang
Andreas Rempel	Justin Tam	Bin Zhang
Jie Ren	Tianqi Tang	Haowen Zhang
Hugues Richard	Sharma V. Thankachan	Martin Zhang
Ahmet Süreyya Rifaioğlu	Tingzhong Tian	Pengfei Zhang
Justin Sanders	Nguyen Khoa Tran	Qimin Zhang
Hirak Sarkar	Achim Tresch	Ran Zhang
Palash Sashittal	Jakub Truszkowski	Ruochi Zhang
Kengo Sato	Alena van Bömmel	Sai Zhang
Johannes Schlüter	Fangping Wan	Cuncong Zhong
Henri Schmidt	Jia Wang	Qinghui Zhou
Sven Schrinner	Xiao Wang	Xiang Zhou
Sara C. Schulte	Yang Wang	Xinyu Zhou
Tizian Schulz	Ziye Wang	Yaoqi Zhou
Saleh Sereshki	Yuhui Wei	Kaiyuan Zhu
Ron Sheinin	Roland Wittler	Haiqi Zhu
Yilun Sheng	Yufeng Wu	Sebastian Zoellner
Huwenbo Shi	Ziqian Xie	Wenxuan Zuo
Qian Shi	Chencheng Xu	
Alireza Fotuhi Siahpirani	Hanwen Xu	

# Contents

## Extended Abstracts

VStrains: De Novo Reconstruction of Viral Strains via Iterative Path Extraction from Assembly Graphs .....	3
<i>Runpeng Luo and Yu Lin</i>	
Spectrum Preserving Tilings Enable Sparse and Modular Reference Indexing .....	21
<i>Jason Fan, Jamshed Khan, Giulio Ermanno Pibiri, and Rob Patro</i>	
Statistically Consistent Rooting of Species Trees Under the Multispecies Coalescent Model .....	41
<i>Yasamin Tabatabaee, Sébastien Roch, and Tandy Warnow</i>	
Sequence to Graph Alignment Using Gap-Sensitive Co-linear Chaining .....	58
<i>Ghanshyam Chandra and Chirag Jain</i>	
DM-Net: A Dual-Model Network for Automated Biomedical Image Diagnosis .....	74
<i>Xiaogen Zhou, Zhiqiang Li, and Tong Tong</i>	
MTGL-ADMET: A Novel Multi-task Graph Learning Framework for ADMET Prediction Enhanced by Status-Theory and Maximum Flow .....	85
<i>Bing-Xue Du, Yi Xu, Siu-Ming Yiu, Hui Yu, and Jian-Yu Shi</i>	
CDGCN: Conditional de novo Drug Generative Model Using Graph Convolution Networks .....	104
<i>Shikha Mallick and Sahely Bhadra</i>	
Percolate: An Exponential Family JIVE Model to Design DNA-Based Predictors of Drug Response .....	120
<i>Soufiane M. C. Mourragui, Marco Loog, Mirrelijn van Nee, Mark A van de Wiel, Marcel J. T. Reinders, and Lodewyk F. A. Wessels</i>	
Translation Rate Prediction and Regulatory Motif Discovery with Multi-task Learning .....	139
<i>Weizhong Zheng, John H. C. Fong, Yuk Kei Wan, Athena H. Y. Chu, Yuanhua Huang, Alan S. L. Wong, and Joshua W. K. Ho</i>	

Computing Shortest Hyperpaths for Pathway Inference in Cellular  
Reaction Networks ..... 155  
*Spencer Krieger and John Kececioglu*

T-Cell Receptor Optimization with Reinforcement Learning and Mutation  
Policies for Precision Immunotherapy ..... 174  
*Ziqi Chen, Martin Renqiang Min, Hongyu Guo, Chao Cheng,  
Trevor Clancy, and Xia Ning*

**Short Papers**

TREE-QMC: Improving Quartet Graph Construction for Scalable  
and Accurate Species Tree Estimation from Gene Trees ..... 195  
*Yunheng Han and Erin K. Molloy*

mapquik: Efficient Low-Divergence Mapping of Long Reads  
in Minimizer Space ..... 197  
*Barış Ekim, Kristoffer Sahlin, Paul Medvedev, Bonnie Berger,  
and Rayan Chikhi*

Deriving Confidence Intervals for Mutation Rates Across a Wide Range  
of Evolutionary Distances Using FracMinHash ..... 200  
*Mahmudur Rahman Hera, N. Tessa Pierce-Ward, and David Koslicki*

Entropy Predicts Sensitivity of Pseudo-random Seeds ..... 203  
*Benjamin Dominik Maier and Kristoffer Sahlin*

Seed-Chain-Extend Alignment is Accurate and Runs in Close to  $O(m \log n)$   
Time for Similar Sequences: A Rigorous Average-Case Analysis ..... 205  
*Jim Shaw and Yun William Yu*

Extremely-Fast Construction and Querying of Compacted and Colored de  
Bruijn Graphs with GGCAT ..... 208  
*Andrea Cracco and Alexandru I. Tomescu*

PASTE2: Partial Alignment of Multi-slice Spatially Resolved  
Transcriptomics Data ..... 210  
*Xinhao Liu, Ron Zeira, and Benjamin J. Raphael*

FastRecomb: Fast Inference of Genetic Recombination Rates in Biobank  
Scale Data ..... 212  
*Ardalan Naseri, William Yue, Shaojie Zhang, and Degui Zhi*

Efficient Taxa Identification Using a Pangenome Index ..... 214  
*Omar Ahmed, Massimiliano Rossi, Christina Boucher,  
and Ben Langmead*

Vector-Clustering Multiple Sequence Alignment: Aligning  
into the Twilight Zone of Protein Sequence Similarity with Protein  
Language Models ..... 217  
*Claire McWhite and Mona Singh*

Single-Cell Methylation Sequencing Data Reveal Succinct Metastatic  
Migration Histories and Tumor Progression Models ..... 219  
*Yuelin Liu, Xuan Cindy Li, Farid Rashidi Mehrabadi,  
Alejandro A. Schäffer, Drew Pratt, David R. Crawford, Salem Malikić,  
Erin K. Molloy, Vishaka Gopalan, Stephen M. Mount, Eytan Ruppín,  
Kenneth Aldape, and S. Cenk Sahinalp*

Information-Theoretic Classification Accuracy: A Criterion That Guides  
Data-Driven Combination of Ambiguous Outcome Labels in Multi-class  
Classification ..... 222  
*Chihao Zhang, Yiling Elaine Chen, Shihua Zhang, and Jingyi Jessica Li*

Efficient Minimizer Orders for Large Values of k Using Minimum  
Decycling Sets ..... 224  
*David Pellow, Lianrong Pu, Baris Ekim, Lior Kotlar, Ron Shamir,  
and Yaron Orenstein*

Dashing 2: Genomic Sketching with Multiplicities and Locality-Sensitive  
Hashing ..... 227  
*Daniel N. Baker and Ben Langmead*

*Startle*: A Star Homoplasy Approach for CRISPR-Cas9 Lineage Tracing ..... 229  
*Palash Sashittal, Henri Schmidt, Michelle Chan,  
and Benjamin J. Raphael*

A Fast and Scalable Method for Inferring Phylogenetic Networks  
from Trees by Aligning Lineage Taxon Strings (Extended Abstract) ..... 230  
*Louxin Zhang, Niloufar Abhari, Caroline Colijn, and Yufeng Wu*

Aligning Distant Sequences to Graphs Using Long Seed Sketches ..... 233  
*Amir Joudaki, Alexandru Meterez, Harun Mustafa,  
Ragnar Groot Koerkamp, André Kahles, and Gunnar Rätsch*

MD-Cat: Phylogenetic Dating Under a Flexible Categorical Model Using Expectation-Maximization .....	236
<i>Uyen Mai, Eduardo Charvel, and Siavash Mirarab</i>	
Phenotypic Subtyping via Contrastive Learning .....	239
<i>Aditya Gorla, Sriram Sankararaman, Esteban Burchard, Jonathan Flint, Noah Zaitlen, and Elior Rahmani</i>	
HOGVAX: Exploiting Peptide Overlaps to Maximize Population Coverage in Vaccine Design with Application to SARS-CoV-2 .....	241
<i>Sara C. Schulte, Alexander T. Dilthey, and Gunnar W. Klau</i>	
Ultra-Fast Genome-Wide Inference of Pairwise Coalescence Times .....	244
<i>Regev Schweiger and Richard Durbin</i>	
Leveraging Family Data to Design Mendelian Randomization That is Provably Robust to Population Stratification .....	247
<i>Nathan LaPierre, Boyang Fu, Steven Turnbull, Eleazar Eskin, and Sriram Sankararaman</i>	
Minimal Positional Substring Cover: A Haplotype Threading Alternative to Li & Stephens Model .....	249
<i>Ahsan Sanaullah, Degui Zhi, and Shaojie Zhang</i>	
Cell Segmentation for High-Resolution Spatial Transcriptomics .....	251
<i>Hao Chen, Dongshunyi Li, and Ziv Bar-Joseph</i>	
Unsupervised Deep Peak Caller for ATAC-seq .....	254
<i>Yudi Zhang, Ha T. H. Vu, Geetu Tuteja, and Karin Dorman</i>	
Unraveling Causal Gene Regulation from the RNA Velocity Graph Using Velorama .....	257
<i>Rohit Singh, Alexander P. Wu, Anish Mudide, and Bonnie Berger</i>	
PIsToN: Evaluating Protein Binding Interfaces with Transformer Networks .....	259
<i>Vitalii Stebliankin, Azam Shirali, Prabin Baral, Prem Chapagain, and Giri Narasimhan</i>	
DebiasedDTA: A Framework for Improving the Generalizability of Drug-Target Affinity Prediction Models .....	262
<i>Rtza Özçelik, Alperen Bağ, Berk Atlı, Melih Barsbey, Arzucan Özgür, and Elif Ozkirimli</i>	

Drug Synergistic Combinations Predictions via Large-Scale Pre-training  
and Graph Structure Learning ..... 265  
*Zhihang Hu, Qinze Yu, Yucheng Guo, Taifeng Wang, Irwin King,  
Xin Gao, Le Song, and Yu Li*

Pisces: A Cross-Modal Contrastive Learning Approach to Synergistic  
Drug Combination Prediction ..... 268  
*Jiacheng Lin, Hanwen Xu, Addie Woicik, Jianzhu Ma, and Sheng Wang*

Modeling and Predicting Cancer Clonal Evolution with Reinforcement  
Learning ..... 271  
*Stefan Ivanovic and Mohammed El-Kebir*

Enabling Trade-Offs in Privacy and Utility in Genomic Data Beacons  
and Summary Statistics ..... 274  
*Rajagopal Venkatesaramani, Zhiyu Wan, Bradley A. Malin,  
and Yevgeniy Vorobeychik*

Accurate Evaluation of Transcriptomic Re-identification Risks Using  
Discriminative Sequence Models ..... 277  
*Shuvom Sadhuka, Daniel Fridman, Bonnie Berger, and Hyunghoon Cho*

**Author Index** ..... 281

## **Extended Abstracts**



# VStrains: De Novo Reconstruction of Viral Strains via Iterative Path Extraction from Assembly Graphs

Runpeng Luo<sup>1</sup> and Yu Lin<sup>1</sup>

School of Computing, Australian National University, Canberra, Australia  
{john.luo,yu.lin}@anu.edu.au

**Abstract.** With the high mutation rate in viruses, a mixture of closely related viral strains (called viral quasispecies) often co-infect an individual host. Reconstructing individual strains from viral quasispecies is a key step to characterizing the viral population, revealing strain-level genetic variability, and providing insights into biomedical and clinical studies. Reference-based approaches of reconstructing viral strains suffer from the lack of high-quality references due to high mutation rates and biased variant calling introduced by a selected reference. De novo methods require no references but face challenges due to errors in reads, the high similarity of quasispecies, and uneven abundance of strains.

In this paper, we propose VStrains, a de novo approach for reconstructing strains from viral quasispecies. VStrains incorporates contigs, paired-end reads, and coverage information to iteratively extract the strain-specific paths from assembly graphs. We benchmark VStrains against multiple state-of-the-art de novo and reference-based approaches on both simulated and real datasets. Experimental results demonstrate that VStrains achieves the best overall performance on both simulated and real datasets under a comprehensive set of metrics such as genome fraction, duplication ratio, NGA50, error rate, *etc.*

**Availability:** VStrains is freely available at <https://github.com/MetaGenTools/VStrains>.

**Keywords:** De Novo Assembly · Viral Quasispecies · Assembly Graph · Path Extraction

## 1 Introduction

Viruses are the most abundant biological entities on Earth and have high mutation rates, up to a million times higher than their hosts [11,26]. Variations in viral genetic sequences lead to the emergence of new viral strains during evolution and are also known to be associated with many diseases [31]. One challenge

---

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_1](https://doi.org/10.1007/978-3-031-29119-7_1).



in viral studies is to analyze a mixture of closely related viral strains, referred to as *viral quasispecies*. The problem of inferring individual viral strains from sequencing data of viral quasispecies is called *strain-aware assembly* or *viral haplotype reconstruction*. Viral haplotype reconstruction in individual patients provides a signature of genetic variability and thus informs us about disease susceptibilities and evolutionary patterns of distinct viral strains [10]. Sequencing data of viral quasispecies from next-generation sequencing (NGS) techniques are short and have a low error rate [33] ( $\leq 0.5\%$ ) while that from third-generation sequencing (TGS) techniques are long and have a high error rate [9] (1.6–2.7% for deletions, 1.2–2.2% for mismatches and 1.1–2.4% for insertions). Due to the low pairwise strain divergence in viral quasispecies, it is challenging to distinguish the sequencing error in TGS data and highly similar viral strains. Therefore, various approaches have been proposed to infer individual viral strains from NGS data and can mainly be classified into two categories [15], reference-based and de novo (or reference-free). Reference-based approaches (such as PredictHaplo [30] and NeurHap [36]) rely on the alignment between reads and references and thus suffer from the lack of high-quality references due to high mutation rates [8], and biased variant calling introduced by a selected reference [3, 34]. De novo approaches directly assemble viral strains from sequencing reads without references and have the potential to identify novel viral strains and provide deep insights for viral genetic novelty [31].

While de novo (meta)-genomic assemblers such as SPAdes-series [1, 5, 7, 24, 28] could be applied to assemble individual strains, they tend to produce fragmented contigs rather than complete viral strains, or collapsed contigs ignoring differences between strains, as they are not specifically designed to distinguish closely related viral strains. Specialized de novo assemblers such as SAVAGE [3], PEHaplo [8], viaDBG [12] and Haploflow [13] directly assemble reads into strains from viral quasispecies and have achieved promising results. More recently, VG-Flow [4] was proposed to extend pre-assembled contigs (produced by the specialized assembler SAVAGE [3]) into full-length viral strains using flow variation graphs and significantly outperformed other approaches on recovering viral strains from viral quasispecies. While VG-Flow [4] guarantees its runtime to be polynomial in the genome size, all the recovered viral strains must be selected from a set of candidate paths inferred by greedy path extraction strategies and thus some viral strains not covered by the candidate set are infeasible to be reconstructed.

Here we propose VStrains, a de novo approach for reconstructing strains from viral quasispecies. VStrains employs SPAdes [5] to build the assembly graph from paired-end reads and incorporates contigs and coverage information to iteratively extract distinct paths as reconstructed strains. We benchmark VStrains against multiple state-of-the-art de novo and reference-based approaches on both simulated and real datasets. Experimental results demonstrate that VStrains achieves the best overall performance on both simulated and real datasets under a comprehensive set of metrics such as genome fraction, duplication ratio, NGA50, error rate, etc. In particular, in more challenging real datasets, VStrains achieves remarkable improvements in recovering viral strains compare to other methods.

## 2 Methods

### 2.1 Preliminary

An assembly graph generated by SPAdes is a directed graph  $G = (V, E)$ . Each vertex  $v \in V$  represents a double-stranded DNA segment where its forward and reverse strands are denoted by  $seq(v^+)$  and  $seq(v^-)$ , respectively. A distinctive feature of assembly graphs built by SPAdes (with iterative  $k$ -mer sizes up to  $k_{max}$ ) is that each  $(k_{max}+1)$ -mer appears in at most one vertex and thus can be used to uniquely identify the corresponding vertex in the assembly graph. Two vertices  $u$  and  $v$  can be connected by an edge  $e = (u^{o_u}, v^{o_v}) \in E$ , where  $o_u, o_v \in \{+, -\}$  denote the strandedness of  $u$  and  $v$ , respectively. Note that the suffix of  $u^{o_u}$  overlaps  $k_{max}$  positions with the prefix of  $v^{o_v}$  under the de Bruijn graph model [29] behind SPAdes. Moreover, the assembly graphs built by SPAdes also contain contigs information, *i.e.*, a contig in  $G(V, E)$  is defined as a path of vertices together with their strandedness information. The *coverage* of a vertex  $v$  is estimated by the number of reads containing the DNA segment corresponding to  $v$  and denoted by  $cov(v)$ . The coverage of a contig  $c$  is estimated by the average coverage along all the vertices in  $c$  and denoted by  $cov(c)$ .

### 2.2 Algorithm Overview

VStrains takes paired-end reads from viral quaspecies as the input and aims to recover individual viral strains. During pre-processing, VStrains first employs SPAdes to construct an assembly graph and contigs from paired-end reads, then canonizes the strandedness of vertices and edges, and further complements the assembly graph with additional linkage information from paired-end reads. After pre-processing, VStrains makes use of the contigs, paired-end links, and coverage information to perform branch splitting and non-branching path contraction to disentangle the assembly graph. Finally, VStrains outputs strain-specific sequences from the assembly graph via iterative contig-based path extraction. Refer to Fig. 1 for an overview of our algorithm. Details of each step are explained in the following sections.

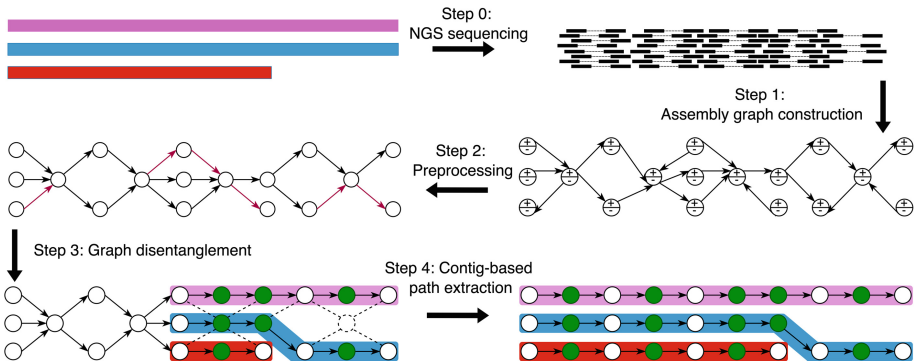


Fig. 1. The framework of VStrains

## 2.3 Preprocessing

### 2.3.1 Canonize Strandedness

Recall that each vertex in the assembly graph produced by SPAdes represents both the forward and reverse strands of a DNA segment, and thus contigs reported by SPAdes may refer to two different strands of the same DNA segment. Therefore, there is no obvious correspondence between a viral strain and a directed path in the assembly graph. To unravel this correspondence, VStrains performs the strandedness canonization to choose the strandedness  $o_v \in \{+, -\}$  for each vertex  $v \in V$  where all adjacent edges only use  $v^{o_v}$ . VStrains first chooses an arbitrary vertex  $s \in V$  as the *starting vertex* and fixes its strandedness to  $o_s$ . Let  $\bar{o}_s$  denotes the opposite strandedness of  $o_s$ , i.e.,  $\bar{o}_s = \{+, -\} \setminus \{o_s\}$ . VStrains flips its adjacent edges if necessary to ensure these edges only use  $s^{o_s}$ , e.g.,  $(u^{o_u}, s^{\bar{o}_s})$  and  $(s^{\bar{o}_s}, u^{o_u})$  will be flipped into  $(s^{o_s}, u^{o_u})$  and  $(u^{\bar{o}_u}, s^{o_s})$ , respectively. VStrains iteratively performs the above step until all the vertices have a fixed strandedness or one vertex has to use both strandedness. VStrains resolves the latter case by splitting this vertex into a pair of vertices, representing its forward and reverse strands, respectively. As a result,  $seq(v^{o_v})$  can be simplified to  $seq(v)$  where  $o_v$  is the chosen strandedness of  $v$ , and  $(u^{o_u}, v^{o_v}) \in E$  can be simplified to  $(u, v)$ , where  $o_u$  and  $o_v$  are the chosen strandedness of  $u$  and  $v$ , respectively. The vertices without any in-coming edges are defined as *source* vertices, whereas the vertices without any out-going edges are defined as *sink* vertices.

### 2.3.2 Inferring Paired-End Links

While SPAdes uses paired-end reads to construct contigs as paths in the assembly graph, VStrains uses unique  $k$ -mers of vertices in the assembly graph to establish mappings between paired-end reads and pairs of vertices and thus infers paired-end links.

VStrains uses minimap2 [20] to find exact matches of  $(k_{max}+1)$ -mers between pairs of vertices in the assembly graph produced by SPAdes (with iterative  $k$ -mer sizes up to  $k_{max}$ ) and paired-end reads. Assume  $u$  and  $v$  are a pair of vertices in the assembly graph. A *PE* link is added between  $u$  and  $v$  if a paired-end read contains at least one  $(k_{max}+1)$ -mer in  $u$  and at least one  $(k_{max}+1)$ -mer in  $v$ . Note that the  $(k_{max}+1)$ -mer can uniquely identify the corresponding vertex and thus makes it extremely unlikely to produce false-positive *PE* links unless errors in reads coincide with rare variations between strains. Since  $(k_{max}+1)$ -mer is usually smaller than the read length, even erroneous paired-end reads may infer paired-end links as long as they still contain error-free  $(k_{max}+1)$ -mers.

Note that paired-end reads are commonly used to infer paired-end links between vertices in the assembly graph. For example, overlap-graph-based assemblers, such as SAVAGE [3] and PEHaplo [8], use pairwise alignments between paired-end reads to build overlap graphs [27], and thus face challenges to choose appropriate parameters (e.g., the overlap length cutoff along with allowed mismatches) to distinguish false-positive links between perfect reads from different strains and true positive links between erroneous reads from the same strain in

overlap graphs. De Bruijn graph-based assemblers, such as SPAdes and viaDBG, split paired-end reads into  $k$ -bimers [5] or bilabel [23], pairs of  $k$ -mers of exact (or near exact) distances, one from the forward read and the other from the reverse read. Although this split strategy helps adjust  $k$ -bimers/bilabel distances [5, 23] and efficiently construct de Bruijn graphs, it does not make full use of all available  $k$ -mer pairs (with varying distances) between forward and reverse reads to further simplify their assembly graphs. To overcome this limitation, VStrains uses all available  $k$ -mer pairs without any distance constraints between forward and reverse reads to create  $PE$  links between vertices in the assembly graph, which unravels its potential to further simplify the assembly graph produced by SPAdes.

## 2.4 Graph Disentanglement

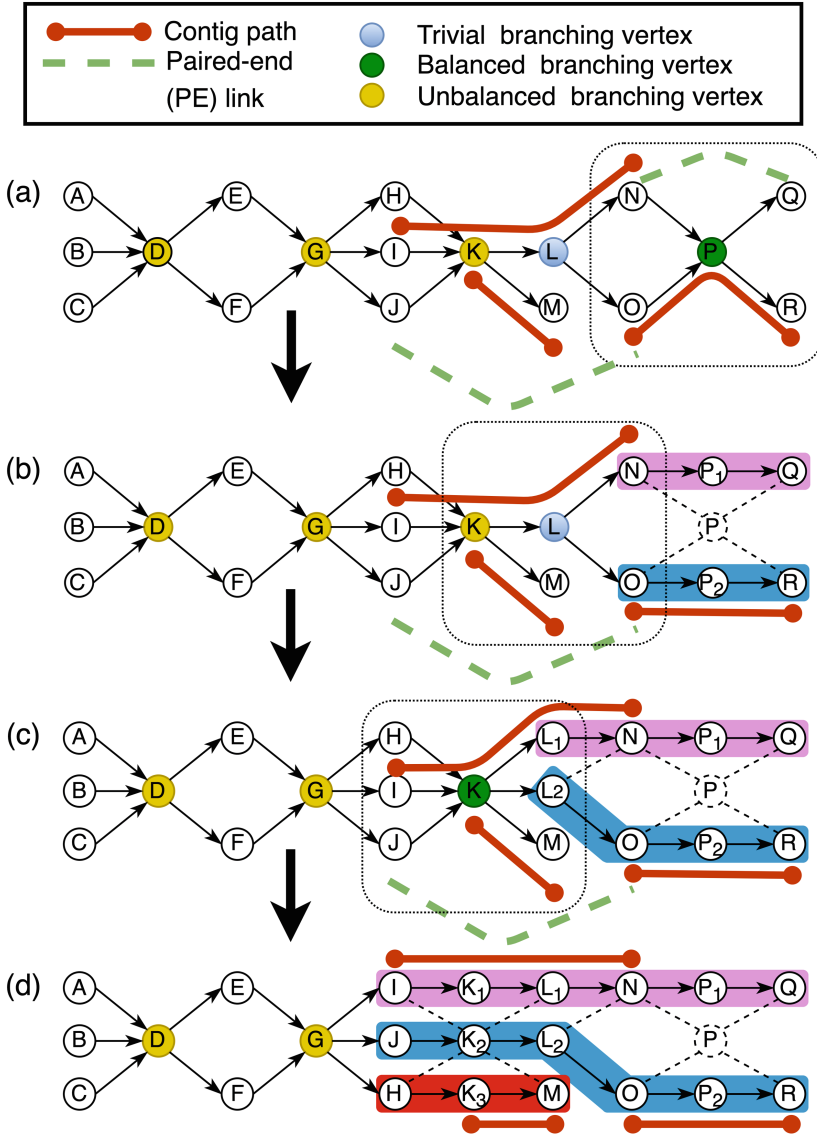
After pre-processing, paired-end link information has been incorporated, and viral strains are expected to correspond to directed paths in the assembly graph. However, strains may share vertices and edges, and thus result in an entangled assembly graph. In this section, the graph disentanglement iteratively splits branching vertices and contracts non-branching paths as follows.

### 2.4.1 Branching Vertex Splitting

A vertex  $v \in V$  is called a *branching vertex* if either the in-degree or out-degree of  $v$  is greater than 1. A branching vertex is *non-trivial* if both its in-degree and out-degree are greater than 1, and *trivial* otherwise. For example, vertex L is a trivial branching vertex while vertices D, G, K, and P are non-trivial branching vertices in Fig. 2(a).

Without loss of generality, assume a trivial branching vertex  $v$  has multiple in-coming edges  $\{(u_i, v) \in E \mid i = 1, \dots, n\}$ . In a trivial split, vertex  $v$  will be replaced by vertices  $\{v_1, \dots, v_n\}$ , and each in-coming edge  $(u_i, v)$  will be replaced by  $(u_i, v_i)$ , respectively. The coverage of  $v_i$  and the capacity of  $(u_i, v_i)$  are set to the capacity of  $(u_i, v)$ . If  $v$  has an out-going edge  $(v, w)$ ,  $(v, w)$  will be replaced by edges  $\{(v_i, w) \mid i = 1, \dots, n\}$  where the capacity of  $(v_i, w)$  is set to the coverage of  $v_i$ .

A non-trivial branching vertex  $v$  is called *balanced* if  $v$  has the same number of in-coming edges  $\{(u_i, v) \in E \mid i = 1, \dots, n\}$  and out-going edges  $\{(v, w_j) \in E \mid j = 1, \dots, n\}$ . For example, vertex P is a balanced branching vertex in Fig. 2(a). Let  $U = \{u_i \mid i = 1, \dots, n\}$  and  $W = \{w_j \mid j = 1, \dots, n\}$ . In a balanced split, the balanced branching vertex  $v$  will be replaced by vertices  $\{v_1, \dots, v_n\}$ , and an in-coming edge  $(u_i, v)$  and out-going edge  $(v, w_i)$  will be replaced by  $(u_i, v_i)$  and  $(v_i, w_i)$  if  $u_i$  and  $w_i$  are both contained in at least one contig or connected by a  $PE$  link. The above balanced split of  $v$  corresponds to a bijection between  $U$  and  $W$ , and most of these one-to-one mappings can be perfectly inferred by contigs and  $PE$  links between  $U$  and  $W$ . For example, a balanced split of P from Fig. 2(a) to (b) corresponds to two one-to-one mappings,  $O \leftrightarrow R$  and  $N \leftrightarrow Q$ , inferred by the contig and  $PE$  link information, respectively. In case  $U$  and  $W$  form a partial



**Fig. 2.** Graph disentanglement of VStrains. (a) P is a balanced branching vertex. (b) P is split into  $P_1$  and  $P_2$  in a balanced split corresponding to  $O \leftrightarrow R$  (contig path) and  $N \leftrightarrow Q$  (PE link). (c) A trivial branching vertex L in (b) is split into  $L_1$  and  $L_2$  in a trivial split, and thus previously unbalanced branching vertex K becomes a balanced branching vertex. (d) K is split into  $K_1$ ,  $K_2$  and  $K_3$  in a balanced split, corresponding to three one-to-one mappings,  $I \leftrightarrow L_1 N P_1 Q$  (contig path),  $J \leftrightarrow L_2 O P_2 R$  (PE link), and  $H \leftrightarrow M$  (coverage compatible pair). D and G are not split during graph disentanglement.

bijection using contigs and  $PE$  links information, VStrains further uses coverage information and aims to find more one-to-one mappings. For the  $u_i \in U$  and  $w_j \in W$  not in the current partial bijection, an one-to-one mapping is established between  $u_i$  and  $w_j$  if and only if  $u_i$  and  $w_j$  form a *coverage compatible pair*, *i.e.*,  $u_i = \arg \min_u |cap(v, w_j) - cap(u, v)|$  and  $w_j = \arg \min_w |cap(v, w) - cap(u_i, v)|$ . For example in Fig. 2(c) to (d), vertices H and M form a coverage compatible pair, together with two other one-to-one mappings  $I \leftrightarrow L_1 NP_1 Q$  (from contig path) and  $J \leftrightarrow L_2 OP_2 R$  (from  $PE$  link), lead to a balanced split on the balanced branching vertex K in Fig. 2(d).

Note that not all non-trivial branching vertices are balanced (*e.g.*, vertex K is an unbalanced branching vertex in Fig. 2(a)). For such unbalanced vertex  $v$ , VStrains performs a trivial split on its adjacent trivial branching vertices and aims to convert  $v$  into a balanced vertex. For example, after performing a trivial split on vertex L in Fig. 2(b) to (c), vertex K now becomes a balanced branching vertex in Fig. 2(c) and becomes a candidate for a balanced split. VStrains performs a balanced split on  $v$  if the above bijection can be established by contigs,  $PE$  links, and coverage information.

#### 2.4.2 Non-branching Path Contraction

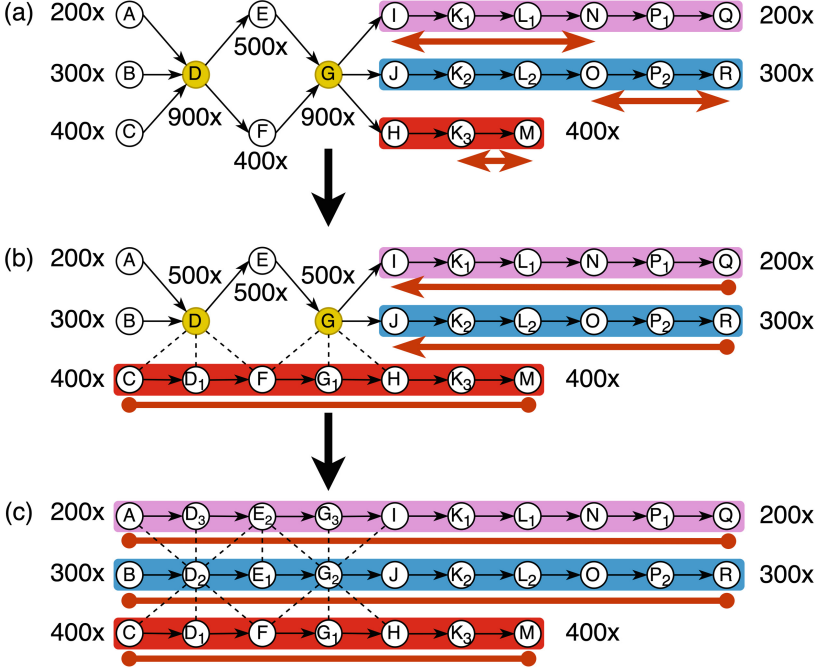
The above branch split operation in the assembly graph creates non-branching paths, *i.e.*, path  $p = (v_1, v_2, \dots, v_n), v_i \in V \forall i = 1, \dots, n$ , where the in-degree of  $v_2, \dots, v_n$  and the out-degree of  $v_1, \dots, v_{n-1}$  are all 1. Following the similar idea of graph simplification in SPAdes, VStrains contracts all the non-branching paths. For example, the above non-branching path  $p$  is contracted into one vertex  $v_p$ , and each in-coming edge  $(u, v_1)$  of  $v_1$  is replaced by  $(u, v_p)$  and each out-going edge  $(v_n, w)$  is replaced by  $(v_p, w)$  with the same capacity, the coverage of  $v_p$  is set to be the average coverage of  $v_1, \dots, v_n$ . Moreover,  $v_p$  inherits all the  $PE$  links of vertices in non-branching path  $p$ . For example in Fig. 2(d), three non-branching paths in three different colors on the right are contracted, respectively.

#### 2.5 Contig-Based Path Extraction

While VStrains effectively disentangles the assembly graph through branching vertex splitting and non-branching path contraction in the above step, there still exist branching vertices (*e.g.*, D and G in Fig. 2(d)) which may introduce ambiguity in distinguishing full paths that correspond to individual viral strains.

Recall SPAdes outputs contigs as paths of vertices on the assembly graph, which are usually sub-paths of viral strain induced paths on the assembly graph. The remaining problem is to extend these contigs (sub-paths) into corresponding viral strains (full paths) on the assembly graph. Note that a full path on the assembly graph starts from a source vertex and ends at a sink vertex or is a cyclic path (*i.e.*, its first and the last vertex coincide). Ideally, the strain-specific (not shared by multiple viral strains) contigs should be extended and extracted first. The longer a contig is, the more likely that this contig is strain-specific.

Therefore, VStrains iteratively selects the longest contig and extends its corresponding sub-path on both ends. Without loss of generality, consider the right



**Fig. 3.** The contig-based path extraction of VStrains. (a) Assembly graph after graph disentanglement with three contigs:  $I \rightarrow K_1 \rightarrow L_1 \rightarrow N$ ,  $O \rightarrow P_2 \rightarrow R$ ,  $K_3 \rightarrow M$ . (b) The longest contig  $I \rightarrow K_1 \rightarrow L_1 \rightarrow N$  is firstly extended to  $I \rightarrow K_1 \dots \rightarrow P_1 \rightarrow Q$  (terminated at I due to the lack of coverage compatible pair with respect to 200x). Afterwards, the second longest contig  $O \rightarrow P_2 \rightarrow R$  is extended to  $J \rightarrow K_2 \rightarrow \dots \rightarrow P_2 \rightarrow R$  (terminated at J due to the lack of coverage compatible pair with respect to 300x). At last, the shortest contig  $K_3 \rightarrow M$  is extended to  $C \rightarrow D_1 \rightarrow \dots \rightarrow K_3 \rightarrow M$  (thanks to the coverage compatible pairs with respect to 400x) (c) Contigs  $J \rightarrow \dots \rightarrow R$  and  $I \rightarrow \dots \rightarrow Q$  are extended to full paths (thanks to the path extraction and coverage/topology update in (b)).

extension of the current contig  $C = (v_1, v_2, \dots, v_n)$ . If  $v_n$  has only one outgoing edge  $(v_n, v_{n+1})$ , the current contig will be extended into  $(v_1, \dots, v_n, v_{n+1})$ . If  $v_n$  has multiple out-going edges  $\{(v_n, v_{n+1}^1), \dots, (v_n, v_{n+1}^m)\}$ , we follow the same strategy in Sect. 2.4.1 to look for one-to-one mapping between  $v_{n-1}$  and  $\{v_{n+1}^1, \dots, v_{n+1}^m\}$  using contigs, paired-end reads and coverage information. If  $v_{n-1}$  forms the one-to-one mapping with  $v_{n+1}^i$ , the current contig will be extended into  $(v_1, \dots, v_n, v_{n+1}^i)$ , otherwise, the extension to the right terminates. If  $v_n$  is a sink vertex (without any out-going edges), the extension to the right terminates. If  $v_n$  is a visited vertex during extension on the other end, the extension to the right terminates and a cyclic path is obtained by combining both left and right extensions.

If the currently selected contig can be extended into a full path, VStrains will include this extended path as one of the output strains. Otherwise, VStrains will



contract this extended path as a single vertex, and wait for confident extension in the future step. VStrains will also update the coverage information of the assembly graph. More specifically, VStrains estimates the path coverage as the median coverage of all the non-branching vertices along the path, and reduces the path coverage from all the traversed vertices. For example, VStrains extends the contig  $H \rightarrow K_3 \rightarrow M$  into the full path  $C \rightarrow D_1 \rightarrow \dots \rightarrow K_3 \rightarrow M$  (using coverage compatible pairs  $F \leftrightarrow H$  and  $C \leftrightarrow F$  described in Sect. 2.4.1) in Fig. 3(b).

By iteratively extending and extracting the contig from the assembly graph, VStrains obtains a set of distinct paths as the final viral strains. This iterative strategy contracts/extracts the most confident sub-paths/full-paths first, and updates topology and coverage information of the assembly graph on the fly, which in turn reveals more strain-specific paths in the updated graph and facilitates the subsequent path extractions. For example, after extracting the full path  $C \rightarrow D_1 \rightarrow \dots \rightarrow K_3 \rightarrow M$  in Fig. 3(b), the (sub)-paths  $I \rightarrow K_1 \rightarrow \dots \rightarrow P_1 \rightarrow Q$  and  $J \rightarrow K_2 \rightarrow \dots \rightarrow P_2 \rightarrow R$  are able to further extend to the left (using coverage compatible pairs  $A \leftrightarrow I$  and  $B \leftrightarrow J$ ) into two full paths  $A \rightarrow D_3 \rightarrow \dots \rightarrow P_1 \rightarrow Q$  and  $B \rightarrow D_2 \rightarrow \dots \rightarrow P_2 \rightarrow R$ .

Note that, unlike other greedy path finding strategies [4, 8, 13] deriving a set of candidate paths based on the original flow-variation graph, VStrain iteratively extracts the most confident path and updates the assembly graph on the fly, which results in more accurate reconstruction of viral strains. For example, VG-Flow employs three greedy strategies (maximum capacity, minimum capacity, shortest paths) to derive the set of candidate paths, from which the final output strains will be selected. However, such greedy strategies are directly applied on the original flow-variation graph, making it likely to find erroneous paths (e.g., the maximum-capacity path  $C \rightarrow D \rightarrow E \rightarrow G \rightarrow H \rightarrow K_3 \rightarrow M$  and the minimum-capacity path  $A \rightarrow D \rightarrow F \rightarrow G \rightarrow I \rightarrow \dots \rightarrow P_1 \rightarrow Q$  in Fig. 3(a) are both erroneous paths).

## 3 Experimental Setup

### 3.1 Experimental Datasets

#### 3.1.1 Simulated Datasets

To evaluate the performance and scalability of VStrains, we used three simulated viral quasispecies datasets from [3] consisting of 6 Poliovirus, 10 hepatitis C virus (HCV), and 15 Zika virus (ZIKV) mixed strains, respectively. These datasets were commonly used to benchmark strain-aware viral assemblers and simulated from known reference genomes using SimSeq [6] with the default error profile.

#### 3.1.2 Real Datasets

Two real datasets with different coverages are obtained from the NCBI database. The first 5-HIV-labmix (20,000x) dataset [14] is a lab mix of 5 known real human immunodeficiency virus (HIV) strains (NCBI accession number *SRR961514*). The second 2-SARS-COV-2 (4,000x) dataset [37] is a mixture of 2 real severe



acute respiratory syndrome coronavirus 2 (SARS-COV-2) strains (BA.1 and B.1.1). Two independently assembled strains, BA.1 (NCBI accession number *SRR18009684*) and B.1.1 (NCBI accession number *SRR18009686*), are used as the ground-truth for evaluation. Table 1 presents a summary of the simulated and real datasets.

**Table 1.** Quasispecies characteristics of the simulated and real benchmarking datasets.

Dataset	Virus type	Genome size	Strain abundance	Pairwise divergence	Data type	Sequencing coverage	Read length
6 Poliovirus	Poliovirus	7.5 kbp	1.6–51%	1.2–7%	Simulated	20,000x	2×250 bp
10 HCV	HCV-1a	9.3 kbp	5–19%	6–9%	Simulated	20,000x	2×250 bp
15 ZIKV	ZIKV	10.3 kbp	2–13%	1–10%	Simulate	20,000x	2×250 bp
5 HIV-labmix	HIV-1	9.6 kbp	10–30%	1–6%	Real	20,000x	2×250 bp
2 SARS-COV-2	SARS-COV-2	30.3 kbp	47.6–52.4%	0.28%	Real	4,000x	2×75 bp

All datasets consist of Illumina Miseq paired-end reads.

### 3.2 Baselines and Evaluation Metrics

VStrains was benchmarked against the state-of-the-art methods on assembling viral strains including SPAdes [5], SAVAGE [3], VG-Flow [4], PEHaplo [8], viaDBG [12], Haploflow [13], and PredictHaplo [30]. Three recent reference-based machine-learning approaches GASeq [17], CAECseq [16] and NeurHap [36] were not included for comparison because all of them have only been applied to a gene segment of viral strains in their experiments and failed to handle the above datasets on whole viral strains (*i.e.*, memory usage exceeding 300 GB). Note that PredictHaplo [30] is a reference-based approach and needs an accurate reference as the input. Therefore, we randomly select a ground-truth viral strain and provide it to PredictHaplo [30] for each dataset. All above tools under evaluation use default settings unless specified otherwise. Refer to Section S3 in the Supplementary Materials for a detailed description of baselines.

Similar to previous studies, we employ MetaQUAST [25] to evaluate all assembly results of viral strains. As viral quasispecies contains highly similar strains, we run MetaQUAST with the “--unique-mapping” option to minimize ambiguous false positive mapping. For each assembly, we report the genome fraction, duplication ratio, NGA50, error rate, and number of contigs. Genome fraction is defined as the total number of aligned bases in the reference, divided by the genome size. A base in the reference genome is counted as aligned if there is at least one contig with at least one alignment to this base. Duplication ratio is defined as the total number of aligned bases in the assembly, divided by the total number of aligned bases in the reference. NG50 is the contig length such that using longer or equal length contigs produce half of the bases of the reference genome, whereas NGA50 counts the lengths of aligned blocks instead of

contig lengths, such that the contig is broken into smaller pieces when it has a misassembly with respect to the reference genome. Error rate is defined as the sum of mismatch rate, indel rate, and N’s rate, which reflects the number of errors with respect to the reference genome size.

## 4 Experimental Results

In this section, we show the performance of VStrains and other baselines on both simulated and real datasets. In consistent with previous observations [4, 12], we found that SPAdes in general outperforms metaSPAdes [28] and other specialized versions (refer to Section S1 in the Supplementary Materials for detailed comparison among SPAdes-series assemblers) and thus is employed to build assembly graphs for VStrains.

### 4.1 Performance on Simulated Datasets

Table 2 summarizes the performance of de novo and reference-based approaches on reconstructing viral strains in three simulated datasets. Note that specialized strain-aware assemblers such as PEHaplo, viaDBG, and SAVAGE typically outperform the general-purpose assembler SPAdes, especially in genome fraction. One possible reason is that SPAdes is not designed to distinguish highly similar viral strains. When two or more strains share long and identical sequences, SPAdes may result in fragmented assemblies (*i.e.*, low NGA50) and keep only one copy of such shared sequences (*i.e.*, low genome fraction). VG-Flow uses assembled contigs from SAVAGE (VG-Flow+SAVAGE) and SPAdes (VG-Flow+SPAdes) to build flow-variation graphs and effectively improves their genome fraction and NGA50. While the greedy strategies in building candidate paths make VG-Flow tractable, VG-Flow may be forced to use more (incorrect) paths to cover all strains (*i.e.* high duplication ratio and error rate) when not all the ground-truth paths have been included in its selected candidate set. VStrains uses contigs, paired-end reads, and coverage information to extract strain-specific paths iteratively from assembly graphs built by SPAdes (VStrains+SPAdes) and achieves the best overall performance on a comprehensive set of metrics (including genome fraction, duplication ratio, NGA50, error rate and number of contigs). It is worth noting that VStrains is able to adapt the general-purpose assembler SPAdes to assemble highly similar strains and even outperform existing specialized strain-aware assemblers.

**Table 2.** Performance of de novo and reference-based approaches on reconstructing viral strains in simulated datasets

6 Poliovirus Strains	Genome Fraction (GF)	Duplication Ratio	NGA50 (# ref strains >50% GF)	Error Rate (mis+indel+N's)	# Contigs (> 500 bp)
PredictHaplo	16.68%	<b>1.00</b>	7459(1)	0.871%	1
PEHaplo	82.68%	<b>1.00</b>	7393(5)	0.160%	5
Haploflow	61.40%	<b>1.00</b>	6671(3)	0.517%	5
viaDBG	68.80%	2.51	2535(6)	0.018%	48
SAVAGE	85.03%	1.70	2930(5)	<b>0.014%</b>	50
VG-Flow+SAVAGE	61.69%	1.27	5656(4)	0.020%	8
SPAdes	43.82%	1.01	5706(2)	0.228%	8
VG-Flow+SPAdes	–	–	–	–	–
VStrains+SPAdes	<b>89.67%</b>	<b>1.00</b>	<b>6682(6)</b>	0.087%	<b>6</b>
10 HCV Strains	Genome Fraction (GF)	Duplication Ratio	NGA50 (# ref strains >50% GF)	Error Rate (mis+indel+N's)	# Contigs (> 500 bp)
PredictHaplo	89.97%	<b>1.00</b>	9292(9)	0.325%	9
PEHaplo	95.95%	1.01	8859(10)	0.013%	12
Haploflow	62.09%	1.55	8893(6)	3.834%	22
viaDBG	97.66%	2.18	9033(10)	<b>0.002%</b>	24
SAVAGE	99.52%	1.07	9059(10)	<b>0.002%</b>	18
VG-Flow+SAVAGE	<b>99.67%</b>	<b>1.00</b>	<b>9264(10)</b>	0.003%	<b>10</b>
SPAdes	90.81%	<b>1.00</b>	8840(10)	0.006%	<b>10</b>
VG-Flow+SPAdes	94.11%	1.10	8687(10)	0.016%	12
VStrains+SPAdes	98.16%	<b>1.00</b>	9124(10)	0.046%	<b>10</b>
15 ZIKV Strains	Genome Fraction (GF)	Duplication Ratio	NGA50 (# ref strains >50% GF)	Error Rate (mis+indel+N's)	# Contigs (> 500 bp)
PredictHaplo	46.66%	<b>1.00</b>	10269(7)	0.427%	7
PEHaplo	84.24%	1.5	6256(15)	0.402%	46
Haploflow	21.82%	4.25	10198(3)	4.167%	26
viaDBG	93.04%	2.97	5136(15)	0.028%	276
SAVAGE	98.85%	1.47	4031(15)	<b>0.011%</b>	116
VG-Flow+SAVAGE	98.23%	1.20	10081(15)	0.077%	18
SPAdes	66.71%	<b>1.00</b>	6447(11)	0.037%	27
VG-Flow+SPAdes	–	–	–	–	–
VStrains+SPAdes	<b>98.87%</b>	<b>1.00</b>	<b>10130(15)</b>	0.068%	<b>16</b>

Three simulated datasets consist of 6-Poliovirus (20,000x), 10-HCV (20,000x), and 15-ZIKV (20,000x). ‘-’ indicates that the corresponding approach failed to complete on the given dataset or exceeded the peak memory limit (500 GB) or CPU time limit (800 h). Refer to Section S2 Table S2.1.x, S2.2.x and S2.3.x in the Supplementary Materials for the detailed strain-level results reported by MetaQUAST.

## 4.2 Performance on Real Datasets

Table 3 summarizes the performance of VStrains and other de novo and reference-based approaches on two real datasets, 5-HIV-labmix [14] and 2-SARS-COV-2 [37]. While viaDBG results in high genome fraction and low error rate, it comes at a cost of (extremely) high duplication ratio and an excessive number of contigs, making it infeasible to distinguish correct and incorrect contigs from its output. As a reference-based approach, PredictHaplo achieves a high genome fraction (99.22%) and high NGA50 (9604) for 5-HIV-labmix dataset because a

ground-truth viral strain was provided as its input. However, in reality, accurate reference genomes are usually not available and species are unknown in viral quaspecies, it is impossible to provide good reference genomes for PredictHaplo.

Similar to the performance on the simulated datasets, SPAdes results in very fragmented assemblies with low genome fraction (49.11%) and NGA50 (614) for 5-HIV-labmix dataset. SAVAGE as a specialized strain-aware assembler produces almost double the genome fraction (87.34%) and NGA50 (1397) compare to SPAdes. While VG-Flow+SAVAGE increases the NGA50 to 7235, it decreases the genome fraction to 77.7%, doubles the duplication ratio (from 1.56 to 3.12), and significantly increase the error rate from 0.115% to 1.038%, which indicates its limitation on handling real datasets. On the other hand, VStrains+SPAdes significantly improves the overall performance on SPAdes, *e.g.*, increase genome fraction from 49.11% to 86.42%, NGA50 from 614 to 7583, and decreases the error rate from 0.515% to 0.237%. Note that VG-Flow+SPAdes fails to achieve such an improvement as VG-Flow is mainly designed to couple with SAVAGE. However, SAVAGE typically assumes at least 10,000x total coverage of viral sequencing data (quote from its [GitHub site](#)), which limits the application of VG-Flow on datasets without ultra-high coverage (*e.g.*, on the 2-SARS-COV-2 dataset with only 4,000x coverage).

**Table 3.** Performance of de novo and reference-based approaches on reconstructing viral strains in real datasets

5 HIV-labmix Strains	Genome Fraction (GF)	Duplication Ratio	NGA50 (# ref strains >50% GF)	Error Rate (mis+indel+N's)	# Contigs (> 500 bp)
PredictHaplo	<b>99.22%</b>	<b>1.00</b>	<b>9604(5)</b>	1.259%	<b>5</b>
PEHaplo	84.19%	1.55	1915(5)	0.300%	41
Haploflow	56.80%	1.26	4832(4)	2.675%	18
viaDBG	92.80%	13.54	5942(5)	<b>0.071%</b>	228
SAVAGE	87.34%	1.56	1397(5)	0.115%	72
VG-Flow+SAVAGE	77.70%	3.12	7235(4)	1.038%	23
SPAdes	49.11%	1.07	614(3)	0.515%	33
VG-Flow+SPAdes	79.75%	2.27	1938(5)	1.137%	78
VStrains+SPAdes	86.42%	1.30	7583(5)	0.237%	12
2 SARS-COV-2 Strains	Genome Fraction (GF)	Duplication Ratio	NGA50 (# ref strains >50% GF)	Error Rate (mis+indel+N's)	# Contigs (> 500 bp)
PredictHaplo	50.02%	9.00	30347(1)	0.024%	9
PEHaplo	67.10%	1.24	21822(1)	0.073%	4
Haploflow	54.78%	1.12	30308(1)	0.059%	<b>3</b>
viaDBG	<b>80.96%</b>	1.52	5501(2)	<b>0.004%</b>	20
SAVAGE	–	–	–	–	–
VG-Flow+SAVAGE	–	–	–	–	–
SPAdes	48.44%	<b>1.00</b>	795(1)	0.014%	7
VG-Flow+SPAdes	–	–	–	–	–
VStrains+SPAdes	63.37%	<b>1.00</b>	<b>12272(2)</b>	0.013%	<b>3</b>

Two real datasets consist of 5-HIV-labmix (20,000x) and 2-SARS-COV-2 (4,000x). ‘–’ indicates that the corresponding approach failed to complete on the given dataset or exceeded the peak memory limit (500 GB) or CPU time limit (800 h). Refer to Section S2 Table S2.4.x and S2.5.x in the Supplementary Materials for the detailed strain-level results reported by MetaQUAST.

## 5 Software and Resource Usage

All the experiments were run under the National Computational Infrastructure (NCI) Gadi supercomputer by submitting jobs to the Gadi biodev queue with the default job dependencies. The allocated RAM size was limited to 500 GB and CPU time was limited to 800h. The peak memory (maximum resident set size) refers to the peak amount of memory throughout the program execution. Table 4 summarize the CPU time and peak memory for different approaches on all viral quasispecies benchmarks, respectively.

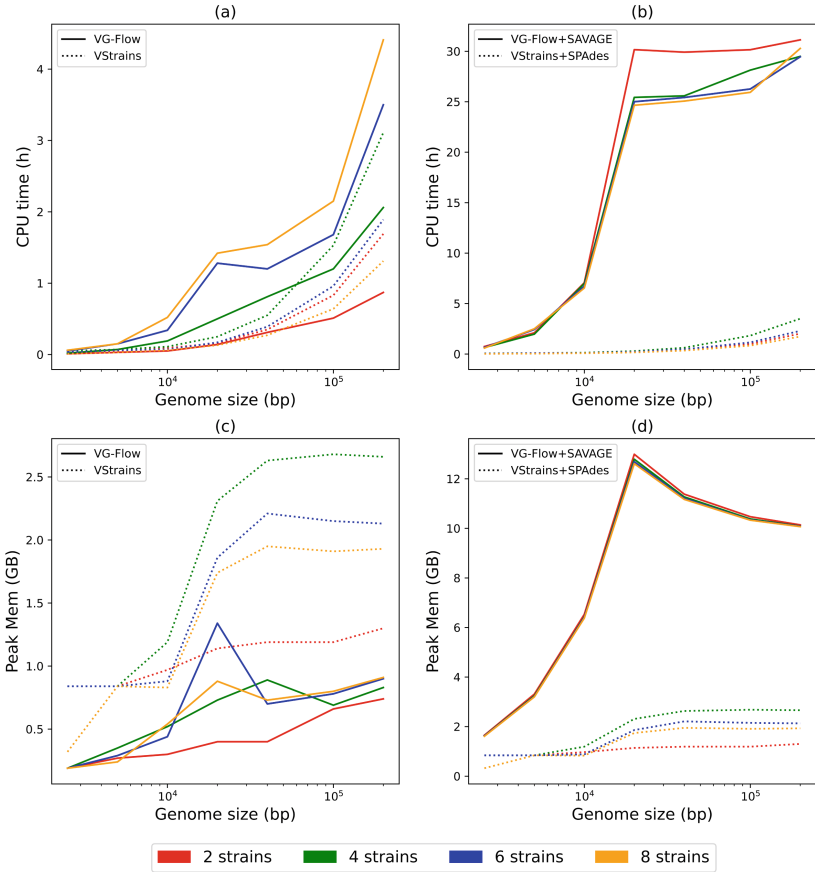
From Table 4, we observe that Haploflow is much more efficient than all other approaches in runtime and peak memory, but may not be a preferred tool due to its extremely high error rate and low genome fraction (as shown in Table 2 and 3). The general-purpose assembler SPAdes is more efficient than specialized strain-aware assemblers such as PEHaplo, Haploflow, viaDBG and SAVAGE in terms of runtime and peak memory. To reconstruct full-length viral strains from pre-assembled contigs, VG-Flow and VStrains cost comparable running time and memory usage. However, since VG-Flow employs SAVAGE to generate the pre-assembled contigs and the running time and memory usage of SAVAGE are almost the most expensive when compared to other tools, which results in high memory usage and running time of “SAVAGE+VG-Flow”. On the contrary, the running time and memory usage of “SPAdes+VStrains” are comparable with other specialized assemblers.

**Table 4.** CPU Time and peak memory usage of de novo and reference-based approaches on viral haplotype reconstruction

	CPU time (hours)					Peak memory (GB)				
	Poliovirus	HCV	ZIKV	HIV	SARS	Poliovirus	HCV	ZIKV	HIV	SARS
PredictHaplo	0.96	2.03	1.99	1.22	19.10	0.77	1.12	1.11	0.91	1.33
PEHaplo	<b>571.93</b>	0.79	<b>127.48</b>	1.20	1.40	5.52	8.98	7.17	4.31	2.36
Haploflow	0.02	0.03	0.03	0.03	0.01	0.45	1.11	1.25	0.43	0.22
viaDBG	0.15	0.24	0.30	0.25	0.18	12.24	15.77	<b>15.85</b>	13.81	<b>8.78</b>
SAVAGE	33.15	<b>18.80</b>	14.84	<b>71.05</b>	–	<b>51.34</b>	<b>26.42</b>	13.39	<b>52.19</b>	–
VG-Flow	1.92	5.93	22.32	10.87	–	0.81	6.93	1.07	1.02	–
SPAdes	0.27	0.41	0.43	0.50	0.10	0.59	0.61	0.60	0.57	0.58
VStrains	3.20	3.76	5.22	6.50	0.23	1.72	1.52	1.78	1.62	0.87

To test the scalability of VStrains, we further compared its runtime and peak memory usage to VG-Flow on simulation datasets with increasing genome size and a variable number of strains (Fig. 4). VG-Flow was selected to compare as it is the most state-of-the-art viral quasispecies assembly post-processing tool. The runtime and memory usage of VStrains do not demonstrate significant correlations with respect to the increase in the number of strains while the runtime of VG-Flow increases with the number of strains. When increasing the genome size,

the runtime and memory usage of both VStrain and VG-Flow increased (Fig. 4(a) and (c)). It is worth noting that, when including the runtime and memory usage of the pre-assemblers (SAVAGE and SPAdes), SAVAGE+VG-Flow is much more sensitive to the genome size than VStrains+SPAdes (Fig. 4(b) and (d)). Thus, we concluded that VStrains+SPAdes is more efficient than VG-Flow+SAVAGE in the analysis of large-scale datasets. Taking into account the good performance of VStrains+SPAdes at Table 2 and Table 3, it is more cost-effective to employ VStrains as a postprocessing tool for SPAdes in terms of both performance and program efficiency compare to VG-Flow.



**Fig. 4.** CPU time and peak memory for VG-Flow, VStrains, VG-Flow+SAVAGE, and VStrains+SPAdes on simulated datasets consist of 2, 4, 6, 8 strains with increasing genome size (bp) (2500, 5000, 10,000, 20,000, 40,000, 100,000, 200,000). The x-axis is plotted on a logarithmic scale. The CPU time for VStrains+SPAdes (VG-Flow+SAVAGE) is the addition of VStrains (VG-Flow) and SPAdes (SAVAGE), and the peak memory for VStrains+SPAdes (VG-Flow+SAVAGE) is the maximum between VStrains (VG-Flow) and SPAdes (SAVAGE).

## 6 Conclusion and Discussion

In VStrains, we first introduce a strategy to canonize the strandedness of assembly graphs from SPAdes, which reduces the strain reconstruction problem into the path extraction problem. Secondly, we use all available  $k$ -mer pairs in paired-end reads to infer  $PE$  links in the assembly graph produced by SPAdes. Thirdly, we propose an effective way to incorporate  $PE$  links together with contigs and coverage information to disentangle the assembly graphs. Finally, we demonstrate how to extract confident strain-specific paths via iterative contig-based path extraction. Experimental results on both simulated and real datasets show that VStrains achieves the best overall performance among the state-of-the-art approaches.

Currently, VStrains relies on both assembly graphs and contigs from SPAdes and thus cannot couple with assemblers which does not explicitly output assembly graphs such as SAVAGE. The current implementation of VStrains requires additional alignments of paired-end reads to vertices in assembly graphs to infer  $PE$  links, which dominates the total runtime and peak memory usage. It is worth exploring how to make VStrains more flexible and efficient.

With the advance of third-generation sequencing (TGS), multiple approaches (including Strainline [22], Strainberry [35], VirStrain [21], viralFlye [2], *etc.*) have been proposed for strain-aware assembly using TGS data. VStrains has the potential to be extended to handle TGS data by taking advantages of assembly graphs built from Canu [19], Flye [18], wtdbg [32] and others.

**Acknowledgements.** We want to thank the anonymous reviewers for providing valuable and detailed feedback. We thank Hansheng Xue and Vijini Mallawaarachchi for testing the machine learning-based method (GAEseq, CAECSeq, and NeurHap) and SPAdes-series assembler (MetaSPAdes and MetaviralSPAdes). We thank Lianrong Pu for polishing the paper and providing valuable advice to us. This research was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

## References

1. Antipov, D., Raiko, M., Lapidus, A., Pevzner, P.A.: METAVIRAL SPADES: assembly of viruses from metagenomic data. *Bioinformatics* **36**(14), 4126–4129 (2020)
2. Antipov, D., Rayko, M., Kolmogorov, M., Pevzner, P.A.: viralFlye: assembling viruses and identifying their hosts from long-read metagenomics data. *Genome Biol.* **23**(1), 1–21 (2022)
3. Baaijens, J.A., El Aabidine, A.Z., Rivals, E., Schönhuth, A.: De novo assembly of viral quasiespecies using overlap graphs. *Genome Res.* **27**(5), 835–848 (2017)
4. Baaijens, J.A., Stougie, L., Schönhuth, A.: Strain-aware assembly of genomes from mixed samples using flow variation graphs. In: Schwartz, R. (ed.) RECOMB 2020. LNCS, vol. 12074, pp. 221–222. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45257-5\\_14](https://doi.org/10.1007/978-3-030-45257-5_14)
5. Bankevich, A., et al.: SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.* **19**(5), 455–477 (2012)

6. Benidt, S., Nettleton, D.: SimSeq: a nonparametric approach to simulation of RNA-sequence datasets. *Bioinformatics* **31**(13), 2131–2140 (2015)
7. Bushmanova, E., Antipov, D., Lapidus, A., Prjibelski, A.D.: rnaSPAdes: a *de novo* transcriptome assembler and its application to RNA-Seq data. *GigaScience* **8**(9), giz100 (2019)
8. Chen, J., Zhao, Y., Sun, Y.: *De novo* haplotype reconstruction in viral quasispecies using paired-end read guided path finding. *Bioinformatics* **34**(17), 2927–2935 (2018)
9. Delahaye, C., Nicolas, J.: Sequencing DNA with nanopores: troubles and biases. *PLoS ONE* **16**(10), e0257521 (2021)
10. Domingo, E., Sheldon, J., Perales, C.: Viral quasispecies evolution. *Microbiol. Mol. Biol. Rev.* **76**(2), 159–216 (2012)
11. Duffy, S.: Why are RNA virus mutation rates so damn high? *PLoS Biol.* **16**(8), e3000003 (2018)
12. Freire, B., Ladra, S., Paramá, J.R., Salmela, L.: Inference of viral quasispecies with a paired de Bruijn graph. *Bioinformatics* **37**(4), 473–481 (2021)
13. Fritz, A.: Haploflow: strain-resolved de novo assembly of viral genomes. *Genome Biol.* **22**(1), 1–19 (2021). <https://doi.org/10.1186/s13059-021-02426-8>
14. Giallonardo, F.D., et al.: Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic Acids Res.* **42**(14), e115 (2014)
15. Jablonski, K.P., Beerenwinkel, N.: Computational methods for viral quasispecies assembly. In: *Virus Bioinformatics*, pp. 51–64. Chapman and Hall/CRC (2021)
16. Ke, Z., Vikalo, H.: A convolutional auto-encoder for haplotype assembly and viral quasispecies reconstruction. In: *Advances in Neural Information Processing Systems* (NeurIPS), vol. 33, pp. 13493–13503 (2020)
17. Ke, Z., Vikalo, H.: A graph auto-encoder for haplotype assembly and viral quasispecies reconstruction. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 719–726 (2020)
18. Kolmogorov, M., Yuan, J., Lin, Y., Pevzner, P.A.: Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.* **37**(5), 540–546 (2019)
19. Koren, S., Walenz, B.P., Berlin, K., Miller, J.R., Bergman, N.H., Phillippy, A.M.: Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* **27**(5), 722–736 (2017)
20. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**(18), 3094–3100 (2018)
21. Liao, H., Cai, D., Sun, Y.: VirStrain: a strain identification tool for RNA viruses. *Genome Biol.* **23**(1), 1–28 (2022)
22. Luo, X., Kang, X., Schönhuth, A.: Strainline: full-length de novo viral haplotype reconstruction from noisy long reads. *Genome Biol.* **23**(1), 1–27 (2022)
23. Medvedev, P., Pham, S., Chaisson, M., Tesler, G., Pevzner, P.: Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. *J. Comput. Biol.* **18**(11), 1625–1634 (2011)
24. Meleshko, D., Hajirasouliha, I., Korobeynikov, A.: coronaSPAdes: from biosynthetic gene clusters to RNA viral assemblies. *Bioinformatics* **38**(1), 1–8 (2021)
25. Mikheenko, A., Saveliev, V., Gurevich, A.: MetaQUAST: evaluation of metagenome assemblies. *Bioinformatics* **32**(7), 1088–1090 (2016)
26. Moelling, K., Broecker, F.: Viruses and evolution—viruses first? A personal perspective. *Front. Microbiol.* **10**, 523 (2019)
27. Myers, E.W.: Toward simplifying and accurately formulating fragment assembly. *J. Comput. Biol.* **2**(2), 275–290 (1995)



28. Nurk, S., Meleshko, D., Korobeynikov, A., Pevzner, P.A.: metaSPAdes: a new versatile metagenomic assembler. *Genome Res.* **27**(5), 824–834 (2017)
29. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.* **98**(17), 9748–9753 (2001)
30. 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. Bioinf.* **11**(1), 182–191 (2013)
31. Pybus, O.G., Rambaut, A.: Evolutionary analysis of the dynamics of viral infectious disease. *Nat. Rev. Genet.* **10**(8), 540–550 (2009)
32. Ruan, J., Li, H.: Fast and accurate long-read assembly with wtdbg2. *Nat. Methods* **17**(2), 155–158 (2020)
33. Stoler, N., Nekrutenko, A.: Sequencing error profiles of Illumina sequencing instruments. *NAR Genomics Bioinform.* **3**(1), lqab019 (2021)
34. 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)
35. Vicedomini, R., Quince, C., Darling, A.E., Chikhi, R.: Strainberry: automated strain separation in low-complexity metagenomes using long reads. *Nat. Commun.* **12**(1), 1–14 (2021)
36. Xue, H., Rajan, V., Lin, Y.: Graph coloring via neural networks for haplotype assembly and viral quasispecies reconstruction. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022, to appear)
37. Yamasoba, D., et al.: Virological characteristics of the SARS-CoV-2 Omicron BA.2 spike. *Cell* **185**(12), 2103–2115 (2022)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Spectrum Preserving Tilings Enable Sparse and Modular Reference Indexing

Jason Fan<sup>1</sup>, Jamshed Khan<sup>1</sup>, Giulio Ermanno Pibiri<sup>2,3</sup>, and Rob Patro<sup>1</sup>(✉)

<sup>1</sup> University of Maryland, College Park, MD 20440, USA  
jasonfan@umd.edu, {jamshed,rob}@cs.umd.edu

<sup>2</sup> Ca' Foscari University of Venice, Venice, Italy  
giulioermanno.pibiri@unive.it

<sup>3</sup> ISTI-CNR, Pisa, Italy

**Abstract.** The reference indexing problem for  $k$ -mers is to pre-process a collection of reference genomic sequences  $\mathcal{R}$  so that the position of all occurrences of any queried  $k$ -mer can be rapidly identified. An efficient and scalable solution to this problem is fundamental for many tasks in bioinformatics.

In this work, we introduce the *spectrum preserving tiling* (SPT), a general representation of  $\mathcal{R}$  that specifies how a set of *tiles* repeatedly occur to spell out the constituent reference sequences in  $\mathcal{R}$ . By encoding the order and positions where *tiles* occur, SPTs enable the implementation and analysis of a general class of modular indexes. An index over an SPT decomposes the reference indexing problem for  $k$ -mers into: (1) a  $k$ -mer-to-tile mapping; and (2) a tile-to-occurrence mapping. Recently introduced work to construct and compactly index  $k$ -mer sets can be used to efficiently implement the  $k$ -mer-to-tile mapping. However, implementing the tile-to-occurrence mapping remains prohibitively costly in terms of space. As reference collections become large, the space requirements of the tile-to-occurrence mapping dominates that of the  $k$ -mer-to-tile mapping since the former depends on the amount of total sequence while the latter depends on the number of unique  $k$ -mers in  $\mathcal{R}$ .

To address this, we introduce a class of sampling schemes for SPTs that trade off speed to reduce the size of the tile-to-reference mapping. We implement a practical index with these sampling schemes in the tool `pufferfish2`. When indexing over 30,000 bacterial genomes, `pufferfish2` reduces the size of the tile-to-occurrence mapping from 86.3 GB to 34.6 GB while incurring only a  $3.6\times$  slowdown when querying  $k$ -mers from a sequenced readset.

**Availability:** `pufferfish2` is implemented in Rust and available at <https://github.com/COMBINE-lab/pufferfish2>.

---

**Supplementary materials:** S.1 to S.8 available online at <https://doi.org/10.5281/zenodo.7504717>.

---

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_2](https://doi.org/10.1007/978-3-031-29119-7_2).

**Keywords:** Reference Indexing · Spectrum Preserving Tilings · Minimal Perfect Hashing · Pufferfish2

## 1 Introduction

Indexing of genomic sequences is an important problem in modern computational genomics, as it enables the atomic queries required for analysis of sequencing data—particularly *reference guided* analyses where observed sequencing data is compared to known *reference* sequences. Fundamentally, analyses need to first rapidly locate short exact matches to reference sequences before performing other operations downstream. For example, for guided assembly of genomes, variant calling, and structural variant identification, seed sequences are matched to known references before novel sequences are arranged according to the seeds [1]. For RNA-seq, statistics for groups of related  $k$ -mers mapping to known transcripts or genes allow algorithms to infer the activity of genes in single-cell and bulk gene-expression analyses [2–4].

Recently, researchers have been interested in indexing collections of genomes for metagenomic and pan-genomic analyses. There have been two main types of approaches: full-text indexes, and hashing based approaches that typically index the *de Bruijn graph* (DBG). With respect to full-text indexes, researchers have developed tools that use the *r-index* [5] to compute matching statistics and locate maximal exact matches for large reference collections [6, 7]. For highly repetitive collections, such as many genomes from the same species, r-index based approaches are especially space efficient since they scale linearly to the number of runs in the *Burrows-Wheeler Transform* (BWT) [8] and not the length of the reference text. With respect to hashing based approaches, tools restrict queries to fixed length  $k$ -mers [1, 9] and index the DBG. These tools achieve faster exact queries but typically trade off space. In other related work, graph-based indexes that compactly represent genomic variations as paths on graphs have also been developed [10, 11]. However, these indexes require additional work to project queries landing on graph-based coordinates to linear coordinates on reference sequences.

Many tools have been developed to efficiently build and represent the DBG [12, 13]. Recently, Khan et al. introduced a pair of methods to construct the compacted DBG from both assembled references [14] and read sets [15]. Ekim et al. [16] introduced the minimizer-space DBG—a highly effective lossy compression scheme that uses minimizers as representative sequences for nodes in the DBG. Karasikov et al. developed the Counting DBG [17] that stores differences between adjacent nodes in the DBG to compress metadata associated with nodes (and sequences) in a DBG. Encouragingly, much recent work on *Spectrum Preserving String Sets* (SPSS) that compactly index the set-membership of  $k$ -mers in reference texts has been introduced [15, 18–23]. Although these approaches do not tackle the *locate* queries directly, they do suggest that even more efficient solutions for reference indexing are possible.

In this work, we extend these recent ideas and introduce the concept of a *Spectrum Preserving Tiling* (SPT) which encodes how and where  $k$ -mers in

an SPSS occur in a reference text. In introducing the SPT, this work makes two key observations. First, a hashing based solution to the reference indexing problem for  $k$ -mers does not necessitate a de Bruijn graph but instead requires a *tiling* over the input reference collection—the SPT formalizes this. Second, the reference indexing problem for  $k$ -mers queries can be cleanly decomposed into a  *$k$ -mer-to-tile* query and a *tile-to-occurrence* query. Crucially, SPTs enable the implementation and analysis of a general class of modular indexes that can exploit efficient implementations introduced in prior work.

**Contributions.** We focus our work on considering how indexes can, *in practice*, efficiently support the two composable queries—the  *$k$ -mer-to-tile* query and the *tile-to-occurrence* query. We highlight this work’s key contributions below. We introduce:

1. The *spectrum preserving tiling* (SPT). An SPT is a general representation that explicitly encodes how shared sequences—*tiles*—repeatedly occur in a reference collection. The SPT enables an entire *class* of sparse and modular indexes that support exact locate queries for  $k$ -mers.
2. An algorithm for sampling and compressing an indexed SPT built from unitigs that *samples* unitig-occurrences. For some small constant “sampling rate”,  $s$ , our algorithm stores the positions of only  $\approx 1/s$  occurrences and encodes all remaining occurrences using a small *constant* number of bits.
3. **Pufferfish2**: a practical index and implementation of the introduced sampling scheme. We highlight the critical engineering considerations that make **pufferfish2** effective in practice.

## 2 Problem Definition and Preliminaries

**The Mapped Reference Position (MRP) Query.** In this work we consider the *reference indexing problem for  $k$ -mers*. Given a collection of references  $\mathcal{R} = \{R_1, \dots, R_N\}$ , where each reference is a string over the DNA alphabet  $\{\mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G}\}$ , we seek an index that can efficiently compute the *mapped reference position* (MRP) query for a fixed  $k$ -mer size  $k$ . Given any  $k$ -mer  $x$ , the MRP query enumerates the positions of all occurrences of  $x$  in  $\mathcal{R}$ . Precisely, each returned occurrence is a tuple  $(n, p)$  that specifies that  $k$ -mer,  $x$ , occurs in reference  $n$  at position  $p$  where  $R_n[p : p + k] = x$ . If a  $k$ -mer does not occur in some  $R_n \in \mathcal{R}$ , the MRP query returns an empty list.

**Basic Notation.** Strings and lists are zero-indexed. The length of a sequence  $S$  is denoted  $|S|$ . The  $i$ -th character of a string  $S$  is  $S[i]$ . A  $k$ -mer is a string of length  $k$ . A sub-string of length  $\ell$  in the string  $S$  starting at position  $i$  is notated  $S[i : i + \ell]$ . The prefix and suffix of length  $i$  is denoted  $S[: i]$  and  $S[|S| - i :]$ , respectively. The concatenation of strings  $A$  and  $B$  is denoted  $A \circ B$ .

We define the *glue* operation,  $A \oplus_k B$ , to be valid for any pair of strings  $A$  and  $B$  that overlap by  $(k - 1)$  characters. If the  $(k - 1)$ -length suffix of  $A$  is equal to the  $(k - 1)$ -length prefix of  $B$ , then  $A \oplus_k B := A \circ B[(k - 1) :]$ . When  $k$  clear from context, we write  $A \oplus B$  in place of  $A \oplus_k B$ .

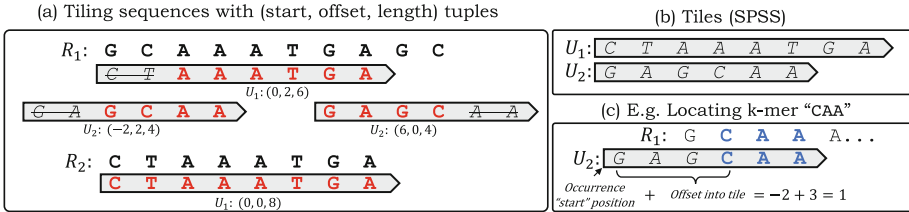
**Rank and Select Queries over Sequences.** Given a sequence  $S$ , the *rank* query given a character  $\alpha$  and position  $i$ , written  $\text{rank}_\alpha(S, i)$ , is the number of occurrences of  $\alpha$  in  $S[:i]$ . The *select* query  $\text{select}_\alpha(S, r)$  returns the position of the  $r$ -th occurrence of symbol  $\alpha$  in  $S$ . The *access* query  $\text{access}(S, i)$  returns  $S[i]$ . For a sequence of length  $n$  over an alphabet of size  $\sigma$ , these can be computed in  $O(\lg \sigma)$  time using a *wavelet matrix* that requires  $n \lg \sigma + o(n \lg \sigma)$  bits [24].

### 3 Spectrum Preserving Tilings

In this section, we introduce the *spectrum preserving tiling*, a representation of a given reference collection  $\mathcal{R}$  that specifies how a set of *tiles* containing  $k$ -mers repeatedly occur to spell out the constituent reference sequences in  $\mathcal{R}$ . This alternative representation enables a modular solution to the reference indexing problem, based on the interplay between two mappings—a  $k$ -mer-to-tile mapping and a tile-to-occurrence mapping.

#### 3.1 Definition

Given a  $k$ -mer length  $k$  and an input reference collection of genomic sequences  $\mathcal{R} = \{R_1, \dots, R_N\}$ , a spectrum preserving tiling (SPT) for  $\mathcal{R}$  is a five-tuple  $\Gamma := (\mathcal{U}, \mathcal{T}, \mathcal{S}, \mathcal{W}, \mathcal{L})$ :



**Fig. 1.** (a) A spectrum preserving tiling (SPT) with  $k = 3$ , (b) with tiles (an SPSS) that contain all  $k$ -mers in references. (c) The SPT explicitly encodes where each  $k$ -mer occurs.

- **Tiles:**  $\mathcal{U} = \{U_1, \dots, U_F\}$ . The set of *tiles* is a spectrum preserving string set, i.e., a set of strings such that each  $k$ -mer in  $\mathcal{R}$  occurs in some  $U_i \in \mathcal{U}$ . Each string  $U_i \in \mathcal{U}$  is called a *tile*.
- **Tiling sequences:**  $\mathcal{T} = \{T_1, \dots, T_N\}$  where each  $T_n$  corresponds to each reference  $R_n \in \mathcal{R}$ . Each tiling sequence is an ordered sequence of tiles  $T_n = [T_{n,1}, \dots, T_{n,M_n}]$ , of length  $M_n$ , with each  $T_{n,m} = U_i \in \mathcal{U}$ . We term each  $T_{n,m}$  a *tile-occurrence*.
- **Tile-occurrence lengths:**  $\mathcal{L} = \{L_1, \dots, L_N\}$ , where each  $L_n = [l_{n,1}, \dots, l_{n,M_n}]$  is a sequence of lengths.
- **Tile-occurrence offsets:**  $\mathcal{W} = \{W_1, \dots, W_N\}$ , where each  $W_n = [w_{n,1}, \dots, w_{n,M_n}]$  is an integer-sequence.

- **Tile-occurrence start positions:**  $\mathcal{S} = \{S_1, \dots, S_N\}$ , where each  $S_n = [s_{n,1}, \dots, s_{n,M_n}]$  is an integer-sequence.

A valid SPT must satisfy the *spectrum preserving tiling property*, that every reference sequence  $R_n$  can be reconstructed by gluing together *substrings of tiles* at offsets  $W_n$  with lengths  $L_n$ :

$$R_n = T_{n,1}[w_{n,1} : w_{n,1} + l_{n,1}] \oplus \dots \oplus T_{n,M_n}[w_{n,M_n} : w_{n,M_n} + l_{n,M_n}].$$

Specifically, the SPT encodes how redundant sequences—*tiles*—repeatedly occur in the reference collection  $\mathcal{R}$ . We illustrate how an ordered sequence of start-positions, offsets, and lengths explicitly specify how redundant sequences tile a pair of references in Fig. 1. More succinctly, each tile-occurrence  $T_{n,m}$  with length  $l_{n,m}$  tiles the reference sequence  $R_n$  as:

$$R_n[s_{n,m} + w_{n,m} : s_{n,m} + w_{n,m} + l_{n,m}] = T_{n,m}[w_{n,m} : w_{n,m} + l_{n,m}].$$

In the same way a small SPSS compactly determines the *presence* of a  $k$ -mer, a small SPT compactly specifies the *location* of a  $k$ -mer. For this work, we consider SPTs where any  $k$ -mer occurs only once in the set of tiles  $\mathcal{U}$ . The algorithms and ideas introduced in this paper still work with SPTs where a  $k$ -mer may occur more than once in  $\mathcal{U}$  (some extra book-keeping of a one-to-many  $k$ -mer-to-tile mapping would be needed, however). For ease of exposition, we ignore tile orientations here. We completely specify the SPT with orientations, allowing tiles to simultaneously represent reverse-complement sequences, in Section S.2.

### 3.2 A General and Modular Index over Spectrum Preserving Tilings

Any SPT is immediately amenable to indexing by an entire *class* of algorithms. This is because an SPT yields a natural decomposition of the MRP query (defined in Sect. 2) where  $k$ -mers first map to the tiles and tile-occurrences then map to positions in references. To index a reference collection, a data structure need only compose a query for the positions where  $k$ -mers occur on tiles in a SPSS with a query for the positions where tiles cover the input references.

Ideally, an index should find a small SPT where  $k$ -mers are compactly represented in the set of tiles where tiles are “long” and tiling sequences are “short”. Compact tilings exist for almost all practical applications since the amount of *unique* sequence grows much more slowly than the *total* length of reference sequences. Finding a small SPSS where  $k$ -mers occur only once has been solved efficiently [18–20]. However, it remains unclear if a small SPSS induces a small SPT, since an SPT must additionally encode tile-occurrence positions. Currently, tools like `pufferfish` index reference sequences using an SPT built from the *unitigs* of the compacted de Bruijn graph (CDBG) constructed over the input sequences, which has been found to be sufficiently compact for practical applications. Though the existence of SPSSs smaller than CDBGs suggest that smaller SPTs might be found for indexing, we leave the problem of finding small or even

optimal SPTs to future work. Here, we demonstrate how indexing any given SPT is *modular* and possible in general.

Given an SPT, the MRP query can be decomposed into two queries that can each be supported by sparse and efficient data structures. These queries are:

- **The kmer-to-tile query:** Given a  $k$ -mer  $x$ ,  $\text{k2tile}(x)$  returns  $(i, p)$ —the identity of the tile  $U_i$  that contains  $x$  and the offset (position) into the tile  $U_i$  where  $x$  occurs. That is,  $\text{k2tile}(x) = (i, p)$  iff  $U_i[p : p + k] = x$ . If  $x$  is not in  $\mathcal{R}$ ,  $\text{k2tile}(x)$  returns  $\emptyset$ .
- **The tile-to-occurrence query:** Given the  $r$ -th occurrence of the tile  $U_i$ ,  $\text{tile2occ}(i, r)$  returns the tuple  $(n, s, w, l)$  that encodes how  $U_i$  tiles the reference  $R_n$ . When  $\text{tile2occ}(i, r) = (n, s, w, l)$ , the  $r$ -th occurrence of  $U_i$  occurs on  $R_n$  at position  $(s + w)$ , with the sequence  $U_i[w : w + l]$ . Let the  $r$ -th occurrence of  $U_i$  be  $T_{n,m}$  on  $\mathcal{T}$ , then  $\text{tile2occ}(i, r)$  returns  $(n, s_{n,m}, w_{n,m}, l_{n,m})$ .

When these two queries are supported, the MRP query can be computed by Algorithm 1. By adding the offset of the queried  $k$ -mer  $x$  in a tile  $U_i$  to the positions where the tile  $U_i$  occurs, Algorithm 1 returns all positions where a  $k$ -mer occurs. Line 10 checks to ensure that any occurrence of the queried  $k$ -mer is returned only if the corresponding tile-occurrence of  $U_i$  contains that  $k$ -mer. We note that storing the number of occurrences of a tile and returning  $\text{num-occs}(U_i)$  requires negligible computational overhead. In practice, the length of tiling sequences,  $\mathcal{T}$ , are orders of magnitude larger than the number of unique tiles. In this work, we shall use  $\text{occs}_i$ , to denote the number of occurrences of  $U_i$  in tiling sequences  $\mathcal{T}$ .

---

**Algorithm 1:**


---

```

1 def mrp(x):
2   |   tup ← k2tile(x)
3   |   if tup = ∅ then
4   |     | return [ ]
5   |   (i, p) ← tup
6   |   occs_i ← num-occs(U_i)
7   |   ans ← [ ]
8   |   for r ← 0 to occs_i do
9   |     | (n, s, w, l) ← tile2occ(i, r)
10  |     | if w ≤ p ≤ (w + l - k) then
11  |     |   | ans.append(n, s + p)
12  |   return ans

```

---

### 3.3 “Drop in” Implementations for Efficient $k$ -mer-to-tile Queries

Naturally, prior work for indexing and compressing spectrum preserving string sets (SPSS) can be applied to implement the  $k$ -mer-to-tile query. When **pufferfish** was first developed, the data structures required to support the  $k$ -mer-to-tile query dominated the size of moderately sized indexes. Thus,

Almodaresi et al. [9] introduced a sampling scheme that samples  $k$ -mer positions in unitigs. Recently, Pibiri [21, 22] introduced `SSHash`, an efficient  $k$ -mer hashing scheme that exploits minimizer based partitioning and carefully handles highly-skewed distributions of minimizer occurrences. When built over an SPSS, `SSHash` stores the  $k$ -mers by their order of appearance in the strings (which we term tiles) of an SPSS and thus allows easy computation of a  $k$ -mer’s offset into a tile. Other methods based on the Burrows-Wheeler transform (BWT) [8], such as the Spectral BWT [23] and BOSS [25], could also be used. However, these methods implicitly sort  $k$ -mers in lexicographical order and would likely need an extra level of indirection to implement `k2tile`. Unless a compact scheme is devised, this can outweigh the space savings offered by the BWT.

### 3.4 Challenges of the Tile-to-Occurrence Query

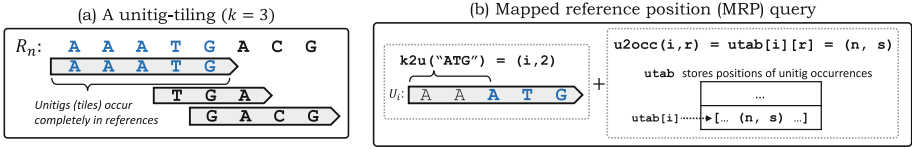
The straightforward solution to the tile-to-occurrence query is to store the answers in a table, `utab`, where `utab[i]` stores information for all occurrences of the tile  $U_i$  and computing `tile2occ(i, r)` amounts to a simple lookup into `utab[i][r]`. This is the approach taken in the `pufferfish` index and has proven to be effective for moderately sized indexes. This implementation is output optimal and is fast and cache-friendly since all  $occ_i$  occurrences of a tile  $U_i$  can be accessed contiguously. However, writing down all start positions of tile-occurrences in `utab` is impractical for large indexes.

For larger indexes (e.g. metagenomic references, many human genomes), explicitly storing `utab` becomes more costly than supporting the  $k$ -mer-to-tile query. This is because, as the number of indexed references grow, the number of distinct  $k$ -mers grows sub-linearly whereas the number of occurrences grows with the (cumulative) reference length. Problematically, the number of start positions of tile-occurrences grows *at least* linearly. For a reference collection with total sequence length  $L$ , a naive encoding for `utab` would take  $O(L \lg L)$  bits, as each position require  $\lceil \lg L \rceil$  bits and there can be at most  $L$  distinct tiles.

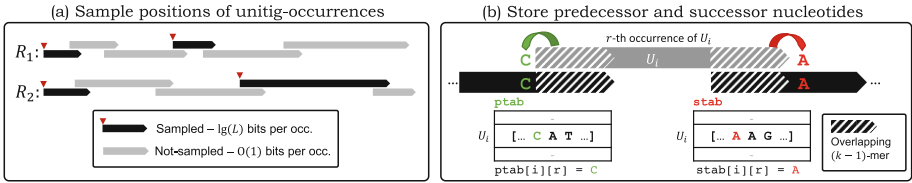
Other algorithms that support “locate” queries suffer from a similar problem. To answer queries in time proportional to the number of occurrences of a query, data structures must explicitly store positions of occurrences and access them in constant time. However, storing *all* positions is impractical for large reference texts or large  $k$ -mer-sets. To address this, some algorithms employ a scheme to *sample* positions at some small sampling rate  $s$ , and perform  $O(s)$  work to retrieve not-sampled positions. Since  $s$  is usually chosen to be a small constant, this extra  $O(s)$  work only imposes a slight overhead.

One may wonder if `utab`—which is an *inverted index*—can be compressed using the techniques developed in the Information Retrieval field [26]. For biological sequences, a large proportion of `utab` consists of very short inverted lists (e.g. unique variants in indexed genomes) that are not well-compressible. In fact, these short lists occur at a rate that is much higher than for inverted indexes designed for natural languages. So, instead applying existing compression techniques, we develop a novel *sampling* scheme for `utab` and the tile-to-occurrence query that exploits the properties of genomic sequences.





**Fig. 2.** (a) A *unitig-tiling* is an SPT where tiles, *unitigs*, always occur completely in the reference sequences. (b) The MRP query is performed by computing a  $k$ -mer’s offset into a unitig ( $k2u$ ), then adding the offset to the positions where *unitig-occurrences* appear in indexed reference sequences ( $u2occ$ ). To naively support the unitig-to-occurrence query, positions of all unitig-occurrences are stored in a table, *utab*.



**Fig. 3.** (a) *Pufferfish2* samples unitigs and their occurrences on a unitig-tiling. Only the positions of the occurrences of the *sampled* unitigs (black) are stored in *utab*. Positions of the *not-sampled* unitigs (gray) can be computed relative to the positions of sampled unitigs by traversing backwards on the visualized tiling of references. Sampling the zero-th unitig-occurrence on every reference sequence guarantees that traversals terminate. (b) Predecessor and successor nucleotides are obtained from adjacent unitig occurrences and are stored in the order in which they appear on the references. These nucleotides for the  $r$ -th occurrence of  $U_i$  is stored in  $ptab[i][r]$  and  $stab[i][r]$ , respectively.

## 4 Pufferfish2

Below, we introduce *pufferfish2*, an index built over an SPT consisting of *unitigs*. *Pufferfish2* applies a sampling scheme to sparsify the tile-to-occurrence query of a given *pufferfish* index [9].

### 4.1 Interpreting pufferfish as an Index over a Unitig-Based SPT

Though not introduced this way by Almodaresi et al., *pufferfish* is an index over a *unitig-tiling* of an input reference collection [9]. A *unitig-tiling* is an SPT which satisfies the property that all tiles always occur completely in references where, for every tile-occurrence  $T_{n,m} = U_i$ , offset  $w_{n,m} = 0$  and length  $l_{n,m} = |U_i|$ . When this property is satisfied, we term tiles *unitigs*.

An index built over unitig-tilings does not need to store tile-occurrence offsets,  $\mathcal{W}$ , or tile-occurrence lengths  $\mathcal{L}$  since all tiles have the same offset (zero) and occur with maximal length. For indexes constructed over unitig-tilings, we shall use  $k2u$  to mean  $k2tile$ , and  $u2occ$  to be  $tile2occ$  with one change.

That is, `u2occ` omits offsets and lengths of tile occurrences since they are uninformative for unitig-tilings and returns a tuple  $(n, s)$  instead of  $(n, s, w, l)$ . In prose, we shall refer to these queries as the  $k$ -mer-to-unitig and unitig-to-occurrence queries.

The MRP query over unitig-tilings can be computed with Algorithm 4 (in Section S.1) where Line 10 is removed from Algorithm 1. We illustrate the MRP query and an example of a unitig-tiling in Fig. 2.

## 4.2 Sampling Unitigs and Traversing Tilings to Sparsify the Unitig-to-Occurrence Query

`Pufferfish2` implements a sampling scheme for *unitig-occurrences* on a unitig-tiling. For some small constant  $s$ , our scheme samples  $1/s$  rows in `utab` each corresponding to *all* occurrences of a unique unitig. In doing so, it sparsifies the `u2occ` query and `utab` by only storing positions for a subset of *sampled* unitigs. To compute unitig-to-occurrence queries, it traverses unitig-occurrences on an indexed unitig-tiling.

Notably, `pufferfish2` traverses unitig-tilings that are *implicitly* represented. For unitig-tilings with positions stored in `utab`, there exists no contiguous sequence in memory representing occurrences that is obvious to traverse. However, when viewed as an SPT, *unitig-occurrences* have *ranks* on a tiling and traversals are possible because tiling sequences map uniquely to a sequence of unitig-rank pairs.

Specifically, we define the `pred` query—an atomic traversal step that enables traversals of arbitrary lengths over reference tilings. Given the  $r$ -th occurrence of the unitig  $U_i$ , the `pred` query returns the identity and rank of the *preceding* unitig. Let tile  $T_{n,m}$  be the  $r$ -th occurrence of the unitig  $U_i$  on all tiling sequences  $\mathcal{T}$ . Then, `pred( $i, r$ )` returns  $(j, q)$  indicating that  $T_{n,m-1}$ , the *preceding* unitig-occurrence, is the  $q$ -th occurrence of the unitig  $U_j$ . If there is no preceding occurrence and  $m = 1$ , `pred( $i, r$ )` returns the sentinel value  $\emptyset$ .

When an index supports `pred`, it is able to traverse “backwards” on a unitig-tiling. Successively calling `pred` yields the identities of unitigs that form a tiling sequence. Furthermore, since `pred` returns the identity  $j$  *and* the rank  $q$  of a preceding unitig-occurrence, accessing data associated with each visited occurrence is straightforward in a table like `utab` (i.e., with `utab[j][q]`).

Given the unitig-set  $\mathcal{U}$ , `pufferfish2` first samples a subset of unitigs  $\mathcal{U}_S \subseteq \mathcal{U}$ . For each sampled unitig  $U_i \in \mathcal{U}_S$ , it stores information for unitig-occurrences identically to `pufferfish` and records, for *all* occurrences of a sampled unitig  $U_i$ , a list of reference identity and position tuples in `utab[i]`.

To recover the position of the  $r$ -th occurrence a not-sampled unitig  $U_i$  and to compute `u2occ( $i, r$ )`, the index traverses the unitig-tiling and iteratively calls `pred` until an occurrence of a sampled unitig is found—let this be the  $q$ -th occurrence of  $U_j$ . During the traversal, `pufferfish2` accumulates number of nucleotides covered by the traversed unitig-occurrences. Since  $U_j$  is a sampled unitig, the position of the  $q$ -th occurrence can be found in `utab[j][q]`. To return `u2occ( $i, r$ )`, `pufferfish2` adds the number of nucleotides traversed to the start

position stored at  $\text{utab}[j][q]$ , the position of a preceding occurrence of the sampled unitig  $U_j$ .

This procedure is implemented in Algorithm 2 and visualized in Fig. 3. Traversals must account for  $(k-1)$  overlapping nucleotides of unitig-occurrences that tile a reference (Line 5). Storing the length of the unitigs is negligible since the number of unique unitigs is much smaller than the number of occurrences.

**On the Termination of Traversals.** Any unitig that occurs as the zero-th occurrence (i.e., with rank zero) of a tiling-sequence is always sampled. This way, backwards traversals terminate because every occurrence of a not-sampled unitig occurs after a sampled unitig. This can be seen from Fig. 3. Concretely, if  $T_{n,1} = U_i$  for some tiling-sequence  $T_n$ , then the unitig  $U_i$  must always be sampled.

---

**Algorithm 2:**


---

```

1 def u2occ( $i, r$ ):
2    $l \leftarrow 0$ 
3   while ! $isSamp[i]$  do
4      $(i, r) = \text{pred}(i, r)$ 
5      $l \leftarrow l + |U_i| - k + 1$ 
6    $(n, s) \leftarrow \text{utab}[i][r]$ 
7   return  $(n, s + l)$ 

```

---



---

**Algorithm 3:**


---

```

1 def pred( $i, r$ ):
2    $p \leftarrow \text{ptab}[i][r]$ 
3    $y \leftarrow p \circ U_i[:k-1]$ 
4    $(j, -) \leftarrow \text{k2u}(y)$ 
5    $s \leftarrow U_i[k]$ 
6    $t \leftarrow \text{rank}_p(\text{ptab}[i], r)$ 
7    $q \leftarrow \text{select}_s(\text{stab}[j], t)$ 
8   return  $(j, q)$ 

```

---

### 4.3 Implementing the pred Query with pufferfish2

Pufferfish2 computes the `pred` query in constant time while requiring only constant space per unitig-occurrence by carefully storing *predecessor* and *successor* nucleotides of unitig-occurrences.

**Predecessor and Successor Nucleotides.** Given the tiling sequence  $T_n = [T_{n,1}, \dots, T_{n,M_n}]$ , we say that a unitig-occurrence  $T_{n,m}$  is *preceded* by  $T_{n,m-1}$ , and that  $T_{n,m-1}$  is *succeeded* by  $T_{n,m}$ . Suppose  $T_{n,m} = U_i$ , and  $T_{n,m-1} = U_j$ , and let the unitigs have lengths  $\ell_i$  and  $\ell_j$ , respectively.

We say that,  $T_{n,m-1}$  precedes  $T_{n,m}$  with predecessor nucleotide  $p$ . The predecessor nucleotide is the nucleotide that precedes the unitig-occurrence  $T_{n,m}$  on the reference sequence  $R_n$ . Concretely,  $p$  is the first nucleotide on the last  $k$ -mer of the preceding unitig, i.e.,  $p = T_{n,m-1}[\ell_j - k]$ . We say that,  $T_{n,m}$  succeeds  $T_{n,m-1}$  with successor nucleotide  $s$ . Accordingly, the successor nucleotide,  $s$ , is the last nucleotide on the first  $k$ -mer of the succeeding unitig, i.e.,  $s = T_{n,m}[k]$ .

Abstractly, the preceding occurrence  $T_{n,m-1}$  can be “reached” from the succeeding occurrence  $T_{n,m}$  by prepending its predecessor nucleotide to the  $(k-1)$ -length prefix of  $T_{n,m}$ . Given  $T_{n,m}$  and its predecessor nucleotide  $p$ , the  $k$ -mer  $y$  that is the last  $k$ -mer on the preceding occurrence  $T_{n,m-1}$  can be obtained with  $y = p \circ T_{n,m}[:k-1]$ . Given an occurrence  $T_{n,m}$ , let the functions `pred-nuc` ( $T_{n,m}$ ) and `succ-nuc` ( $T_{n,m}$ ) yield the predecessor nucleotide and

the successor nucleotide of  $T_{n,m}$ , respectively. If  $T_{n,m}$  is the first or last unitig-occurrence pair on  $T_n$ , then  $\text{succ-nuc}(T_{n,m})$  and  $\text{pred-nuc}(T_{n,m})$  return the “null” character, ‘\$’.

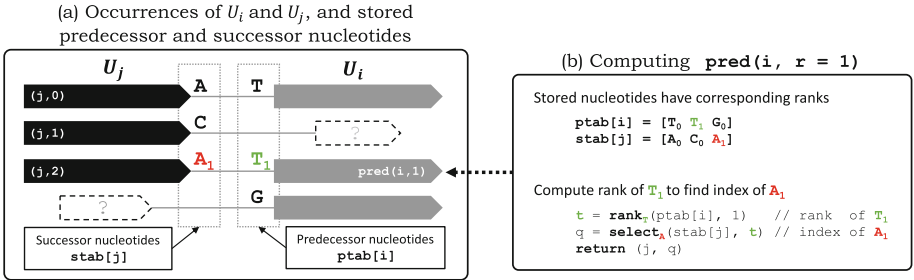
These notationally dense definitions can be more easily understood with a figure. Figure 3 shows how predecessor and successor nucleotides of a given unitig-occurrence on a tiling are obtained.

**Concrete Representation.** Pufferfish2 first samples a set of unitigs  $\mathcal{U}_S \subseteq \mathcal{U}$  from  $\mathcal{U}$  and stores a bit vector,  $\text{isSamp}$ , to record if a unitig  $U_i$  is sampled where  $\text{isSamp}[i] = 1$  iff  $U_i \in \mathcal{U}_S$ . Pufferfish2 stores in  $\text{utab}$  the reference identity and position pairs for occurrences of *sampled* unitigs only.

After sampling unique unitigs, pufferfish2 stores a *predecessor nucleotide table*,  $\text{ptab}$ , and a *successor nucleotide table*,  $\text{stab}$ . For each not-sampled unitig  $U_i$  only,  $\text{ptab}[i]$  stores a list of predecessor nucleotides for each occurrence of  $U_i$  in the unitig-tiling. For *all* unitigs  $U_i$ ,  $\text{stab}[i]$  stores a list of successor nucleotides for each occurrence of  $U_i$ . Concretely, when the unitig-occurrence  $T_{n,m}$  is the  $r$ -th occurrence of  $U_i$ ,

$$\text{ptab}[i][r] = \text{pred-nuc}(T_{n,m}) \quad \text{and} \quad \text{stab}[i][r] = \text{succ-nuc}(T_{n,m}).$$

As discussed in Sect. 4.2, unitigs that occur as the zero-th element on a tiling is always sampled so that every occurrence of a not-sampled unitig has a predecessor. If  $T_{n,m}$  has no successor and is the last unitig-occurrence on a tiling sequence,  $\text{stab}[i][j]$  contains the sentinel symbol ‘\$’. Figure 3 illustrates how predecessor and successor nucleotides are stored.



**Fig. 4.** Visualizing the  $\text{pred}$  query that finds the occurrence of  $U_j$  that precedes the queried occurrence of  $U_i$  with rank 1. (a) All occurrences of  $U_i$  and  $U_j$  are visualized (in sorted order) with their preceding and succeeding unitig occurrences, respectively. The figure shows stored successor nucleotides for  $U_j$ , and predecessor nucleotides for  $U_i$ . Whenever an occurrence of  $U_j$  precedes an occurrence of  $U_i$ , a corresponding pair of nucleotides “A” and “T” occur and are stored in  $\text{stab}[j]$  and  $\text{ptab}[i]$  respectively. (b) Their *ranks* (annotated with subscripts) of the corresponding predecessor-successor nucleotide pair *match* in  $\text{ptab}[i]$  and  $\text{stab}[j]$ , but the *indices* do not. A rank query for predecessor nucleotide “T” at index  $r = 1$  yields the matching rank of the successor nucleotide “A”. A select query for the nucleotide “A” with rank 1 yields the *index* and occurrence of the predecessor  $U_j$ .

**Computing the pred Query.** Given the  $k$ -mer-to-unitig query, `pufferfish2` supports the `pred` query for any unitig  $U_i$  that is not-sampled. When the  $r$ -th occurrence of  $U_i$  succeeds the  $q$ -th occurrence of  $U_j$ , it computes  $\text{pred}(i, r) = (j, q)$  with Algorithm 3. To compute `pred`, it constructs a  $k$ -mer to find  $U_j$ , and then computes one rank and one select query over the stored lists of nucleotides to find the correct occurrence.

`Pufferfish2` first computes  $j$ , the identity of the preceding unitig. The last  $k$ -mer on the preceding unitig must be the first  $(k - 1)$ -mer of  $U_i$  prepended with predecessor nucleotide of the  $r$ -th occurrence of  $U_i$ . Given  $\text{ptab}[i][r] = p$ , it constructs the  $k$ -mer,  $y = p \circ U_i[:k - 1]$ , that must be the last  $k$ -mer on  $U_j$ . So on Line 4, it computes `k2u(y)` to obtain the identity of the preceding unitig  $U_j$ .

It then computes the unitig-rank,  $q$ , of the preceding unitig-occurrence of  $U_j$ . Each time  $U_i$  is preceded by the nucleotide  $p$ , it must be preceded by the *same* unitig  $U_j$  since any  $k$ -mer occurs in only one unitig. Accordingly, each occurrence  $U_j$  that is succeeded by  $U_i$  must always be succeeded by the *same* nucleotide  $s$  equal to the  $k$ -th nucleotide of  $U_i$ ,  $U_i[k]$ . For the preceding occurrence of  $U_j$  that the algorithm seeks to find, the nucleotide  $s$  is stored at some unknown index  $q$  in  $\text{stab}[j]$ —the list of successor nucleotides of  $U_j$ .

Whenever an occurrence of  $U_i$  succeeds an occurrence of  $U_j$ , so do the corresponding pair predecessor and successor nucleotides stored in  $\text{ptab}[i]$  and  $\text{stab}[j]$ . Since  $\text{ptab}[i]$  and  $\text{stab}[j]$  store predecessor and successor nucleotides in the order in which unitig-occurrences appear in the tiling sequences, the following *ranks* of stored *nucleotides* must be equal: (1) the rank of the nucleotide  $p = \text{ptab}[i][r]$  at index  $r$  in the list of predecessor nucleotides,  $\text{ptab}[i]$ , of the succeeding unitig  $U_i$ , and (2) the rank of the nucleotide  $s = U_i[k]$  at index  $q$  in the list of successor nucleotides,  $\text{stab}[j]$ , of the preceding unitig  $U_j$ . We illustrate this correspondence between ranks in Fig. 4. So to find  $q$ , the rank of the preceding unitig-occurrence, `pufferfish2` computes the rank of the predecessor nucleotide,  $t = \text{rank}_p(\text{ptab}[i], r)$ . Then, computing  $\text{select}_s(\text{stab}[j], t)$ , the index where the  $t$ -th rank successor nucleotide of  $U_j$  occurs must yield  $q$ .

**Time and Space Analysis.** `Pufferfish2` computes the `pred` query in constant time. The  $k$ -mer for the query `k2u` is assembled in constant time, and the `k2u` query itself is answered in constant time, as already done in the `pufferfish` index [9].

For not-sampled unitigs, `pufferfish2` does not store positions of unitig-occurrences in `utab`. Instead, it stores nucleotides in tables `stab` and `ptab`. These tables are implemented by *wavelet matrices* that support rank, select, and access operations in  $O(\lg \sigma)$  time on sequences with alphabet size  $\sigma$  while requiring only  $\lg \sigma + o(\lg \sigma)$  bits per element [24].

As explained in Sect. 3.1, we have avoided the treatment of *orientations* of nucleotide sequences for brevity. In actuality, unitigs may occur in a *forward* or a *backwards* orientation (i.e., with a reverse complement sequence). When considering orientations, `pufferfish2` implements the `pred` query by storing and

querying over lists of *nucleotide-orientation* pairs. In this case, `ptab` and `stab` instead store predecessor-orientation and successor-orientation pairs. Accordingly, wavelet matrices are then built over alphabets of size 8 and 9 respectively—deriving from eight nucleotide-orientation pairs and one sentinel value for unitig-occurrences that have no predecessor. Thus, `ptab` and `stab` in total require  $\approx 7$  bits per unitig-occurrence (since  $7 = \lceil \lg 8 \rceil + \lceil \lg 9 \rceil$ ). We describe how the `pred` query is implemented with orientations in Section S.3.

**Construction.** The current implementation of `pufferfish2` sparsifies the unitig-to-occurrence query and compresses the table of unitig occurrences, `utab`, of an existing `pufferfish` index, and inherits its  $k$ -mer-to-unitig mapping. In practice, sampling and building a `pufferfish2` index always takes less time than the initial `pufferfish` index construction. In brief, building `pufferfish2` amounts to a linear scan over an SPT. We describe how `pufferfish2` is constructed in more detail in Section S.4.

#### 4.4 A Random Sampling Scheme to Guarantee Short Backwards Traversals

Even with a constant-time `pred` query, computing the unitig-to-occurrence query is fast only if the length of backwards traversals—the number of times `pred` is called—is small. So for some small constant  $s$ , a sampling scheme should sample  $1/s$  of *unique* unitigs, store positions of only  $1/s$  of unitig-occurrences in `utab`, and result in traversal lengths usually of length  $s$ .

At first, one may think that a greedy sampling scheme that traverses tiling sequences to sample unitigs could be used to bound traversal lengths to some given maximum length,  $s$ . However, when tiling sequences become much longer than the number of unique unitigs, such a greedy scheme samples almost *all* unitigs and only somewhat effective in limited scenarios (see Section S.5). Thus, we introduce the *random* sampling scheme that samples  $1/s$  of unitigs uniformly at random from  $\mathcal{U}$ . This scheme guarantees that traversals using the `pred` query terminate in  $s$  steps *in expectation* if each unitig-occurrence  $T_{n,m}$  is independent and identically distributed and drawn from an arbitrary distribution. Then, backwards traversals until the occurrence of a sampled unitig is a series of Bernoulli trials with probability  $1/s$ , and traversal lengths follow a geometric distribution with mean  $s$ . Although this property relies on a simplifying assumption, the random sampling scheme works well in practice.

**Table 1.** Size and speed of **pufferfish2** indexes querying 10 million random  $k$ -mers and 100,000 reads. Uncompressed, baseline implementations of the unitig-to-occurrence query (**pufferfish** indexes with the *sparse k2u* implementation [9]) are labeled with “None” sampling strategy. Relative sizes of compressed representations and relative slowdowns to the baseline are indicated in parentheses.

Dataset	Sampling strategy	u2occ size (GB)	10M $k$ -mers (secs)	100K reads (secs)
7 Humans	None	16.8	86.1	139.4
	Random ( $s = 3, t = .05$ )	7.8 (0.46)	4159.1 (43.8 $\times$ )	8092.8 (58.04 $\times$ )
	Random ( $s = 3, t = .25$ )	9.9 (0.59)	681.1 (7.9 $\times$ )	1466.2 (10.52 $\times$ )
4000 Bacteria	None	7.7	35.5	12.6
	Random ( $s = 3, t = .05$ )	3.7 (0.48)	420.4 (11.9 $\times$ )	15.6 (1.24 $\times$ )
	Random ( $s = 3, t = .25$ )	4.7 (0.61)	323.8 (9.1 $\times$ )	15.5 (1.23 $\times$ )
30K Human gut	None	86.3	80.6	178.7
	Random ( $s = 3, t = .05$ )	45.6 (0.53)	439.4 (5.5 $\times$ )	570.2 (3.19 $\times$ )
	Random ( $s = 3, t = .25$ )	54.4 (0.63)	365.2 (4.5 $\times$ )	576.9 (3.23 $\times$ )
	Random ( $s = 6, t = .05$ )	34.6 (0.40)	1037.5 (12.9 $\times$ )	644.8 (3.61 $\times$ )
	Random ( $s = 6, t = .25$ )	45.6 (0.53)	614.0 (7.6 $\times$ )	646.1 (3.56 $\times$ )

#### 4.5 Closing the Gap Between a Constant Time `pred` Query and Contiguous Array Access

Even though the `pred` query is constant time and traversals are short, it is difficult to implement `pred` queries in with speed comparable to *contiguous array accesses* that are used to compute the `u2occ` for when `utab` is “dense”—i.e., uncompressed and not sampled. In fact, any compression scheme for `utab` would have difficulty contending with constant time contiguous array access regardless of their asymptotics since dense implementations are output optimal, very cache friendly, and simply store the answers to queries in an array. To close the gap between theory and practice, **pufferfish2** exploits several optimizations.

In practice, a small proportion of unique unitigs are “popular” and occur extremely frequently. Fortunately, the total number of occurrences of popular unitigs is small relative to other unitigs. To avoid an excessively large number of traversals from a not-sampled unitig, **pufferfish2** modifies the sampling scheme to always sample popular unitigs that occur more than a preset number,  $\alpha$ , times. Better yet, we re-parameterize this optimization and set  $\alpha$  so that the total number of occurrences of popular unitigs sum to a given proportion  $0 < t \leq 1$  of the total occurrences of all the unitigs. For example, setting  $t = 0.25$  restricts **pufferfish2** to sample from 75% of the total size of `utab` consisting of unitigs that occur most infrequently.

Also, the MRP and `pred` query are especially amenable to caching. Notably, **pufferfish2** caches and memoizes redundant `k2u` queries in successive `pred` queries. Also, it caches “streaming” queries to exploit the fact that successive queried  $k$ -mers (e.g., from the same sequenced read) likely land on the same unitig. We describe in more detail these and other important optimizations in Section S.6.

## 5 Experiments

We assessed the space-usage of the indexes constructed by `pufferfish2` from several different whole-genome sequence collections, as well as its query performance with different sampling schemes. Reported experiments were performed on a server with an Intel Xeon CPU (E5-2699 v4) with 44 cores and clocked at 2.20 GHz, 512 GB of memory, and a 3.6 TB Toshiba MG03ACA4 HDD.

**Datasets.** We evaluated the performances on a number of datasets with varying attributes: (1) Bacterial collection: a random set of 4000 bacterial genomes from the NCBI microbial database; (2) Human collection: 7 assembled human genome sequences from [27]; and (3) Metagenomic collection: 30,691 representative sequences from the most prevalent human gut prokaryotic genomes from [28].

**Results.** To emulate a difficult query workload, we queried the indexes with 10 million random *true positive*  $k$ -mers sampled uniformly from the indexed references. Our results from Table 1 show that sampling *popular* unitigs is critical to achieve reasonable trade-offs between space and speed. When indexing seven human genomes, the difference in space between always sampling using  $t = 0.05$  and  $t = 0.25$ , is only 2.1 GB (12.5% of the uncompressed `utab`). However, explicitly recording 2.1 GB of positions of occurrences of popular unitigs, *substantially* reduces the comparative slowdown from  $43.8\times$  to  $7.9\times$ . This is because setting  $t = 0.25$  instead of  $t = 0.05$  greatly reduces the maximum number of occurrences of a *not-sampled* unitig—from  $\approx 87,000$  to  $\approx 9,000$  times, respectively. Here, setting  $t = 0.25$  means that random  $k$ -mer queries that land in not-sampled unitigs perform many fewer traversals over reference tilings.

On metagenomic datasets, indexes are compressed to a similar degree but differences in query speed at different parameter settings are small. `Pufferfish2` is especially effective for a *large* collection of bacterial genomes. With the fastest parameter setting, it incurs only a  $4.5\times$  slowdown for random queries while reducing the size of `utab` for the collection of 30,000 bacterial genomes by 37% (from 86.3 GB to 54.4 GB).

Apart from random lookup queries, we also queried the indexes with  $k$ -mers deriving from sequenced readsets [29,30]. We measured the time to query and recover the positions of all  $k$ -mers on 100,000 reads. This experiment demonstrates how the slowdown incurred from sampling can (in most cases) be further reduced when queries are positionally coherent or miss. Successive  $k$ -mer queries from the same read often land on the same unitig and can thus be cached (see Sect. 4.5). *True negative*  $k$ -mers that do not occur in the indexed reference collection neither require traversals nor incur any slowdowns.

To simulate a metagenomic analysis, we queried reads from a human stool sample against 4,000 bacterial genomes. This is an example of a low hit-rate analysis where 18% of queried  $k$ -mers map to indexed references. In this scenario, `pufferfish2` reduces the size of `utab` by *half* but incurs only a  $1.2\times$  slowdown. We also queried reads from the same human stool sample against the collection of 30,000 bacterial genomes representative of the human gut. Here, 88% of  $k$ -mers are found in the indexed references. At the sparsest setting, `pufferfish2` indexes incur only a  $3.6\times$  slowdown while reducing the size of `utab` by 60%.



We observe that `pufferfish2`'s sampling scheme is less effective when indexing a collection of seven human genomes. When sampled with  $s = 3$  and  $t = 0.25$ , `pufferfish2` incurs a  $10.5\times$  slowdown when querying reads from a DNA-seq experiment in which 92% of queried  $k$ -mers occur in reference sequences. Interestingly, the slowdown when querying reads is larger than the slowdown when querying random  $k$ -mers. This is likely due to biases from sequencing that cause  $k$ -mers and reads to map to non-uniformly indexed references. Nonetheless, this result motivates future work that could design sampling schemes optimized for specific distributions of query patterns.

We expect to see less-pronounced slowdowns in practice than those reported in Table 1. This is because tools downstream of an index like `pufferfish2` almost always perform operations *much* slower after straightforward exact lookups for  $k$ -mers. For example, aligners have to perform alignment accounting for mismatches and edits. Also, our experiments pre-process random  $k$ -mer sets and read-sets so that no benchmark is I/O bound. Critically, the compromises in speed that `pufferfish2` makes are especially palatable because it trades-off speed in the *fastest* operations in analyses—*exact*  $k$ -mer queries—while substantially reducing the space required for the *most space intensive* operation.

**Table 2.** Sizes in GB of possible, new indexes—with `k2u` implemented by `SSHash` and `u2occ` by `pufferfish2`—compared to the size of original `pufferfish` indexes. Selected sampling parameters for datasets (top-to-bottom) are  $(s = 3, t = 0.25)$ ,  $(s = 3, t = 0.05)$ , and  $(s = 6, t = 0.05)$ , respectively.

Dataset	u2occ w/ <code>pufferfish2</code>	k2u w/ <code>SSHash</code>	New index	Original <code>pufferfish</code> index
7 Human	9.9	3.2	<b>13.1</b>	28.0
4000 Bacteria	3.7	7.3	<b>11.0</b>	26.1
30K Human gut	34.6	22.0	<b>55.6</b>	131.7

**Using SSHAsh for Even Smaller Indexes.** For convenience, we have implemented our SPT compression scheme within an index that uses the *specific* sparse `pufferfish` implementation for the  $k$ -mer-to-tile ( $k$ -mer-to-unitig) mapping [9]. However, the SPT enables the construction of modular indexes that use *various* data structures for the  $k$ -mer-to-tile mapping and the tile-to-reference mapping, provided only a minimalistic API between them. A recent representation of the  $k$ -mer-to-tile mapping that supports all the necessary functionality is `SSHash` [22]. Compared to the `k2u` component of `pufferfish`, `SSHash` is almost always substantially smaller. Further, it usually provides faster query speed compared to the *sparse* `pufferfish` implementation of the  $k$ -mer-to-tile query, especially when streaming queries are being performed.

In Table 2, we calculate the size of indexes if `SSHash` is used for the  $k$ -mer-to-tile mapping—rather than the *sparse* `pufferfish` implementation. These sizes then represent overall index sizes that would be obtained by pairing a state-of-the-art representation of the  $k$ -mer-to-tile mapping with a state-of-the-art

representation of the tile-to-reference mapping (that we have presented in this work). Practically, the only impediment to constructing a fully-functional index from these components is that they are implemented in different languages (C++ for `SSHash` and `Rust` for `pufferfish2`)—we are currently addressing this issue.

Importantly, these results demonstrate that, when `SSHash` is used, the representation of the tile-to-occurrence query becomes a bottleneck in terms of space, occupying an increasingly larger fraction of the overall index. Table 2 shows that, in theory, if one fully exploits the modularity of SPTs, new indexes that combine `SSHash` with `pufferfish2` would be *half* the space of the original `pufferfish` index. As of writing, with respect to an index over 30,000 bacterial genomes, the estimated difference in *monetary* cost of an AWS EC2 instance that can fit a new 55.6 GB index versus a 131 GB `pufferfish` index in memory is 300USD per month (see Section S.7).

**Comparing to MONI and the r-Index.** We compared `pufferfish2` to MONI, a tool that builds an r-index to locate maximal exact matches in highly repetitive reference collections [6]. In brief, `pufferfish2` is faster and requires less space than MONI for our benchmarked bacterial dataset. Our tool does so with some trade-offs. `Pufferfish2` supports rapid locate queries for  $k$ -mers of a *fixed* length, while r-index based approaches supports locate queries for patterns of any *arbitrary* length and can be used to find MEMs. Notably, it has been shown that both  $k$ -mer and MEM queries can be used for highly effective read-mapping and alignment [1,6].

For reference, we built MONI on our collection of 4,000 bacterial genomes. Here, MONI required 51.0G of disk space to store which is 29% larger than the `pufferfish` index (39.5 GB) with its *dense k2u* implementation—its *least* space-efficient configuration. The most space efficient configuration of the `pufferfish2` index (with  $s = 3$ ,  $t = .25$ ) is 42% the size of MONI when built on from the same data and requires 21.7 GB of space. Compared to a theoretically possible index specified in Table 2 that would only require 11.0 GB, MONI would need  $4.6\times$  more space.

We also performed a best-effort comparison of query speed between `pufferfish2` and MONI. Unfortunately, it is not possible to directly measure the speed of exact locate queries for MONI because it does not expose an interface for such queries. Instead, we queried MONI to find MEMs on true-positive  $k$ -mers treating each  $k$ -mer as unique read (encoded in FASTQ format as MONI requires). We argue that this is a reasonable proxy to exact locate queries because, for each true-positive  $k$ -mer deriving from an indexed reference sequence, the entire  $k$ -mer itself is the maximal exact match. For MONI, just like in benchmarks for in Table 1, we report the time taken for computing queries only and ignore time required for I/O operations (i.e. loading the index and queries, and writing results to disk).

We found that `pufferfish2` is faster than MONI when querying  $k$ -mers against our collection of 4,000 bacterial genomes. MONI required 1,481.7s to query the same set of 10 million random true-positive  $k$ -mers queried in Table 1. When compared to the slowest built most space efficient configuration of `pufferfish2` benchmarked in Table 1, `pufferfish2` is  $3.5\times$  faster.

## 6 Discussion and Future Work

In this work, we introduce the *spectrum preserving tiling* (SPT), which describes how a spectrum preserving string set (SPSS) tiles and “spells” an input collection of reference sequences. While considerable research effort has been dedicated to constructing space and time-efficient indexes for SPSS, little work has been done to develop efficient representations of the tilings themselves, despite the fact that these tilings tend to grow more quickly than the SPSS and quickly become the size bottleneck when these components are combined into reference indexes. We describe and implement a sparsification scheme in which the space required for representing an SPT can be greatly reduced in exchange for an expected constant-factor increase in the query time. We also describe several important heuristics that are used to substantially lessen this constant-factor in practice. Having demonstrated that modular reference indexes can be constructed by composing a  $k$ -mer-to-tile mapping with a tile-to-occurrence mapping, we have thus opened the door to exploring an increasingly diverse collection of related reference indexing data structures.

Despite the encouraging progress that has been made here, we believe that there is much left to be explored regarding the representation of SPTs, and that many interesting questions remain open. Some of these questions are: (1) How would an algorithm sample individual unitig-occurrences instead of all occurrences of a unitig to *explicitly* bound the lengths of backwards traversals? (2) Does a smaller SPSS imply a small SPT and could one compute an optimally small SPT? (3) Given some distributional assumptions for queries, can an algorithm sample SPTs to minimize the expected query time? (4) In practice, how can an implemented tool combine our sampling scheme with existing compression algorithms for the highly skewed tile-to-occurrence query? (5) Can a *lossy* index over an SPT be constructed and applied effectively in practical use cases?

With excitement, we discuss in more detail these possibilities for future work in more detail in Section S.8.

**Funding.** This work is supported by the NIH under grant award numbers R01HG009937 to R.P.; the NSF awards CCF-1750472 to R.P. and CNS-1763680 to R.P.; and NSF award No. to DGE-1840340 J.F. This work was also partially supported by the project MobiDataLab (EU H2020 RIA, grant agreement N<sup>o</sup>101006879).

**Conflicts of Interest.** R.P. is a co-founder of Ocean Genomics Inc.

## References

1. Almodaresi, F., Zakeri, M., Patro, R.: PuffAligner: a fast, efficient and accurate aligner based on the pufferfish index. *Bioinformatics* **37**(22), 404–4055 (2021)
2. Patro, R., Duggal, G., Love, M.I., Irizarry, R.A., Kingsford, C.: Salmon provides fast and bias-aware quantification of transcript expression. *Nat. Methods* **14**(4), 417–419 (2017)
3. Bray, N.L., Pimentel, H., Melsted, P., Pachter, L.: Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.* **34**(5), 525–527 (2016)

4. 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)
5. Gagic, T., Navarro, G., Prezza, N.: Optimal-time text indexing in BWT-runs bounded space. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, USA*, pp. 1459–1477. Society for Industrial and Applied Mathematics (2018)
6. Rossi, M., Oliva, M., Langmead, B., Gagic, T., Boucher, C.: MONI: a pangenomic index for finding maximal exact matches. *J. Comput. Biol.* **29**(2), 169–187 (2022). PMID: 35041495
7. Ahmed, O., Rossi, M., Gagic, T., Boucher, C., Langmead, B.: SPUMONI 2: improved pangenome classification using a compressed index of minimizer digests. *BioRxiv* (2022)
8. Burrows, M., Wheeler, D.: A block-sorting lossless data compression algorithm. Digital SRC Research Report, Citeseer (1994)
9. Almodaresi, F., Sarkar, H., Srivastava, A., Patro, R.: A space and time-efficient index for the compacted colored de Bruijn graph. *Bioinformatics* **34**(13), i169–i177 (2018)
10. Kim, D., Paggi, J.M., Park, C., Bennett, C., Salzberg, S.L.: Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nat. Biotechnol.* **37**(8), 907–915 (2019)
11. Garrison, E., et al.: Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat. Biotechnol.* **36**(9), 875–879 (2018)
12. Minkin, I., Pham, S., Medvedev, P.: TwoPaCo: an efficient algorithm to build the compacted de Bruijn graph from many complete genomes. *Bioinformatics* **33**(24), 4024–4032 (2016)
13. Chikhi, R., Limasset, A., Medvedev, P.: Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* **32**(12), i201–i208 (2016)
14. Khan, J., Patro, R.: Cuttlefish: fast, parallel and low-memory compaction of de Bruijn graphs from large-scale genome collections. *Bioinformatics* **37**(Supplement\_1), i177–i186 (2021)
15. Khan, J., Kokot, M., Deorowicz, S., Patro, R.: Scalable, ultra-fast, and low-memory construction of compacted de Bruijn graphs with Cuttlefish 2. *Genome Biol.* **23**(1), 190 (2022). <https://doi.org/10.1186/s13059-022-02743-6>
16. Ekim, B., Berger, B., Chikhi, R.: Minimizer-space de Bruijn graphs: whole-genome assembly of long reads in minutes on a personal computer. *Cell Syst.* **12**(10), 958–968.e6 (2021)
17. Karasikov, M., Mustafa, H., Rättsch, G., Kahles, A.: Lossless indexing with counting de Bruijn graphs. *Genome Res.* **32**(9), 1754–1764 (2022)
18. Rahman, A., Medvedev, P.: Representation of  $k$ -mer sets using spectrum-preserving string sets. In: Schwartz, R. (ed.) *RECOMB 2020*. LNCS, vol. 12074, pp. 152–168. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45257-5\\_10](https://doi.org/10.1007/978-3-030-45257-5_10)
19. Schmidt, S., Alanko, J.N.: Eulertigs: minimum plain text representation of  $k$ -mer sets without repetitions in linear time. *BioRxiv* (2022)
20. Brinda, K., Baym, M., Kucherov, G.: Simplitigs as an efficient and scalable representation of de Bruijn graphs. *Genome Biol.* **22**(1), 96 (2021). <https://doi.org/10.1186/s13059-021-02297-z>
21. Pibiri, G.E.: On weighted  $k$ -mer dictionaries. In: *International Workshop on Algorithms in Bioinformatics (WABI)*, pp. 9:1–9:20 (2022)

22. Pibiri, G.E.: Sparse and skew hashing of k-mers. *Bioinformatics* **38**(Supplement\_1), i185–i194 (2022)
23. Alanko, J.N., Puglisi, S.J., Vuohtoniemi, J.: Succinct k-mer sets using subset rank queries on the spectral burrows-wheeler transform. *BioRxiv* (2022)
24. Claude, F., Navarro, G.: The wavelet matrix. In: Calderón-Benavides, L., González-Caro, C., Chávez, E., Ziviani, N. (eds.) *SPIRE 2012*. LNCS, vol. 7608, pp. 167–179. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34109-0\\_18](https://doi.org/10.1007/978-3-642-34109-0_18)
25. Bowe, A., Onodera, T., Sadakane, K., Shibuya, T.: Succinct de Bruijn graphs. In: Raphael, B., Tang, J. (eds.) *WABI 2012*. LNCS, vol. 7534, pp. 225–235. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33122-0\\_18](https://doi.org/10.1007/978-3-642-33122-0_18)
26. Pibiri, G.E., Venturini, R.: Techniques for inverted index compression. *ACM Comput. Surv.* **53**(6), 125:1–125:36 (2021)
27. Baier, U., Beller, T., Ohlebusch, E.: Graphical pan-genome analysis with compressed suffix trees and the Burrows-Wheeler transform. *Bioinformatics* **32**(4), 497–504 (2015)
28. Hiseni, P., Rudi, K., Wilson, R.C., Hegge, F.T., Snipen, L.: HumGut: a comprehensive human gut prokaryotic genomes collection filtered by metagenome data. *Microbiome* **9**(1), 165 (2021)
29. Zook, J.M., et al.: Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Sci. Data* **3**(1), 160025 (2016)
30. Mas-Lloret, J., et al.: Gut microbiome diversity detected by high-coverage 16S and shotgun sequencing of paired stool and colon sample. *Sci. Data* **7**(1), 92 (2020)
31. Moffat, A., Stuver, L.: Binary interpolative coding for effective index compression. *Inf. Retrieval* **3**(1), 25–47 (2000). <https://doi.org/10.1023/A:1013002601898>
32. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**(18), 3094–3100 (2018)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Statistically Consistent Rooting of Species Trees Under the Multispecies Coalescent Model

Yasamin Tabatabaee<sup>1</sup>, Sébastien Roch<sup>2</sup>, and Tandy Warnow<sup>1</sup>

<sup>1</sup> Department of Computer Science, University of Illinois at Urbana-Champaign,  
Urbana, IL, USA

warnow@illinois.edu

<sup>2</sup> Department of Mathematics, University of Wisconsin-Madison, Madison, USA

**Abstract.** Rooted species trees are used in several downstream applications of phylogenetics. Most species tree estimation methods produce unrooted trees and additional methods are then used to root these unrooted trees. Recently, Quintet Rooting (QR) (Tabatabaee et al., ISMB and Bioinformatics 2022), a polynomial-time method for rooting an unrooted species tree given unrooted gene trees under the multispecies coalescent, was introduced. QR, which is based on a proof of identifiability of rooted 5-taxon trees in the presence of incomplete lineage sorting, was shown to have good accuracy, improving over other methods for rooting species trees when incomplete lineage sorting was the only cause of gene tree discordance, except when gene tree estimation error was very high. However, the statistical consistency of QR was left as an open question. Here, we present QR-STAR, a polynomial-time variant of QR that has an additional step for determining the rooted shape of each quintet tree. We prove that QR-STAR is statistically consistent under the multispecies coalescent model, and our simulation study shows that QR-STAR matches or improves on the accuracy of QR. QR-STAR is available in open source form at <https://github.com/ytabatabaee/Quintet-Rooting>.

**Keywords:** Species Tree Estimation · Rooting · Statistical Consistency · Multispecies Coalescent

## 1 Introduction

Inferring rooted species trees is important for many downstream applications of phylogenetics, such as comparative genomics [7, 11] and dating [25]. These estimations use different loci from across the genomes of the selected species, and so are referred to as multi-locus analyses. If rooted gene trees can be accurately inferred, then the rooted species tree can be estimated from them [14]; however, this is not a reliable assumption [29]. Hence, the standard approach is to first

---

The full paper is available at <https://doi.org/10.1101/2022.10.26.513897> and includes the Supplementary Materials.

© The Author(s) 2023

H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 41–57, 2023.

[https://doi.org/10.1007/978-3-031-29119-7\\_3](https://doi.org/10.1007/978-3-031-29119-7_3)

estimate an unrooted species tree using multi-locus datasets, and then root that estimated tree.

The estimation of the unrooted species tree is challenged by biological processes, such as incomplete lineage sorting (ILS) or gene duplication and loss (GDL), that can result in different parts of the genome having different evolutionary trees. When ILS or GDL occur, statistically consistent estimation of the unrooted species tree requires techniques that take the source of heterogeneity into consideration [15,22]. The case of ILS, as modeled by the multispecies coalescent (MSC) model [10], is the most well-studied, and there are several methods for estimating unrooted species trees that have been proven statistically consistent under the MSC (see [22] for a survey of such methods).

The general problem of rooting a species tree (or indeed even a gene tree) is of independent interest, but presents many challenges. A common approach is the use of an outgroup taxon (i.e., the inclusion of a species that is outside the smallest clade containing the remaining species), so that the resultant tree is rooted on the edge leading to the outgroup [16]. However, outgroup selection has its own difficulties: if the outgroup is too distant, then it may be attached fairly randomly to the tree containing the remaining species, and if it is too close, it may even be an ingroup taxon [5,6,9,13]. Other approaches use branch lengths estimated on the tree to find the root based on specific optimization criteria; however, these approaches tend to degrade in accuracy unless the strict molecular clock holds (which assumes that all sites along the genome evolve under a constant rate) [8,18,31].

Quintet Rooting (QR) [30] is a recently introduced method that is designed to root a given species tree using the unrooted gene tree topologies, under the assumption that the gene trees can differ from the species tree due to ILS. QR is based on mathematical theory established by Allman, Degnan, and Rhodes [2], which showed that the rooted species tree topology is identifiable from the unrooted gene tree topologies whenever the number of species is at least five. In [30], QR was shown to provide good accuracy for rooting both estimated and true species trees in the presence of ILS, compared to alternative methods.

However, QR was not proven to be statistically consistent for locating the root. Thus, we do not have a proof that the root location selected by QR, when given the true species tree topology, will converge to the correct location as the number of gene trees in the input increases. Although much attention has been paid to establishing statistical consistency for unrooted species tree estimation methods and many methods, such as ASTRAL [19], SVDQuartets [32] and BUCKy [12], have been proven to be statistically consistent estimators of the unrooted species tree topology under the MSC, to the best of our knowledge, no prior study has addressed the statistical consistency properties of methods for rooting species trees.

In this paper, we argue that QR is not guaranteed to be statistically consistent under the MSC, but we also present a modification to QR, which we call QR-STAR, that we prove statistically consistent. Moreover, QR-STAR, like QR, runs in polynomial time. We also provide results of a simulation study comparing QR to QR-STAR. Due to space limitations, most of the proofs and results from the simulation study are presented in the full version of the paper available at <https://doi.org/10.1101/2022.10.26.513897>.

## 2 Background

We present the theory from [2] first, which establishes identifiability of the rooted species tree from unrooted quintet trees, and then we describe Quintet Rooting (QR), our earlier method for rooting species trees. Together these form the basis for deriving our new method, QR-STAR, which we present in the next section.

### 2.1 Allman, Degnan, and Rhodes (ADR) Theory

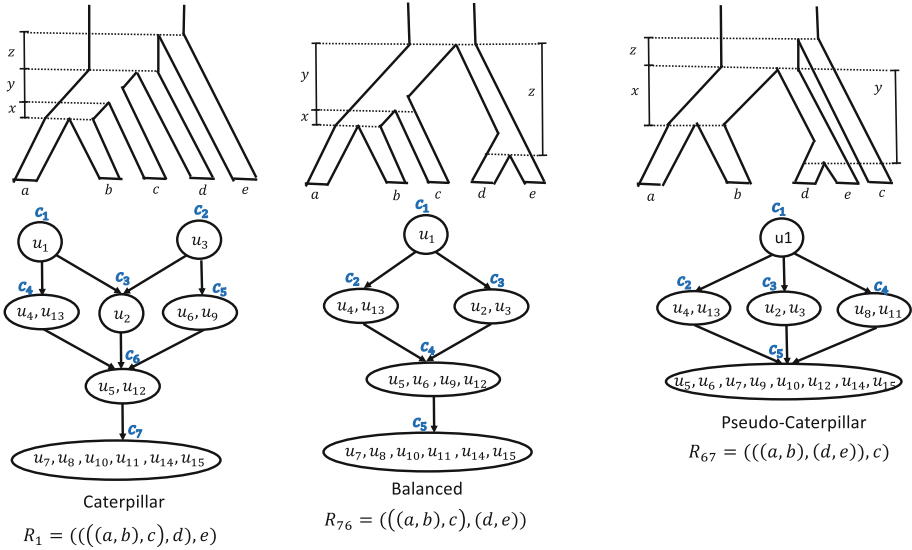
Allman, Degnan, and Rhodes (ADR) [2] established that the unrooted topology of the species tree is identifiable from four-leaf unrooted gene trees under the MSC, a result that is well known and used in several “quartet-based” methods for estimating species trees under the MSC [12, 17, 19, 24]. ADR also proved that the rooted species tree topology is identifiable from unrooted five-leaf gene tree topologies; this result is much less well known, but was recently used in the development of QR for rooting species trees.

ADR have described the probability distribution of unrooted gene tree topologies under each 5-taxon MSC model species tree. On a given set of five taxa, there exist 105 different rooted binary trees, labeled with  $R_1, \dots, R_{105}$ <sup>1</sup>, that can be categorized into three groups based on their (unlabeled) rooted shapes: caterpillar, balanced and pseudo-caterpillar [27]. An example of a tree from each category is shown in Fig. 1. Each 5-taxon model species tree defines a specific probability distribution over the 15 different unrooted gene tree topologies on the same leafset, shown with  $T_1, \dots, T_{15}$ . Theorem 9 in [2] states that this distribution uniquely determines the rooted tree topology and its internal branch lengths for trees with at least five taxa.

To prove this identifiability result, the ADR theory specifies a set of linear invariants (i.e., equalities) and inequalities that must hold between the probabilities of unrooted 5-taxon gene trees, for any choice of the parameters of the model species tree. These linear invariants and inequalities define a partial order on the probabilities of the topologies of the different 5-taxon unrooted gene trees. In other words, two gene tree probabilities  $u_i = \mathbb{P}(T_i)$  and  $u_j = \mathbb{P}(T_j)$  can have one of four possible relationships:  $u_i > u_j$ ,  $u_j > u_i$ ,  $u_i = u_j$ , or  $u_i$  and  $u_j$  are not comparable.

<sup>1</sup> The labeling of rooted and unrooted trees in this paper is consistent with the notations and leaf-labeling used in Tables 4–5 in [2] as well as in [30].





**Fig. 1. ADR invariants and inequalities for different rooted topological shapes.** The invariants and inequalities found by ADR define, for each rooted 5-taxon model tree topology, a partial order on the probabilities of the 15 unrooted 5-leaf gene trees; importantly, the partial order depends only on the “rooted shape” of the 5-taxon model species trees (i.e., caterpillar, balanced and pseudo-caterpillar). Thus, the topology of any 5-leaf rooted binary species tree is uniquely determined by the partial order, and so can be determined from the true distribution on unrooted 5-leaf gene trees (i.e., it is identifiable, as established by ADR).

Figure 1 shows examples of these partial orders with a Hasse diagram for a particular leaf labeling of trees from each rooted shape. Note that some probabilities are members of the same set (e.g., for  $R_1$ , set  $c_4$  contains both  $u_4$  and  $u_{13}$ , indicating that  $u_4 = u_{13}$ ), and so we refer to the sets  $c_i$  as equivalence classes on these probabilities. Furthermore, we will denote the set of equivalence classes associated with a 5-taxon rooted tree  $R$  with  $C_R$ . As can be seen in Fig. 1, the number of equivalence classes depends on the shape of the rooted species tree, with caterpillar, balanced and pseudo-caterpillar trees having 7, 5 and 5 equivalence classes, respectively.

Each directed edge between two equivalence classes in these Hasse diagrams defines an inequality, so that all gene tree probabilities in class  $c_a$  at the source of an edge are greater than all gene tree probabilities in class  $c_b$  at the target, and we denote this by  $c_a > c_b$ . The exact values of the unrooted gene tree probabilities depend on the internal branch lengths of the model tree, and ADR provide a set of formulas that relate the model tree parameters to the probability distribution of the unrooted gene trees in Appendix B of [2], which will be used in our proofs.

## 2.2 Quintet Rooting

The input to QR is an unrooted species tree  $T$  with  $n$  leaves and a set  $\mathcal{G}$  of  $k$  single-copy unrooted gene trees where the gene trees draw their leaves from the leafset of  $T$ , denoted by  $\mathcal{L}(T)$ . Given this input, QR searches over all possible rootings of  $T$  and returns a tree most consistent with the distribution of quintets (i.e., 5-taxon trees) in the input gene trees.

QR approaches this problem by selecting a set  $Q$  of quintets of taxa from  $\mathcal{L}(T)$  (called the “quintet sampling” step; refer to Supplementary Materials Sec. A for details), and scoring all rooted versions of  $T$  based on their induced trees on these quintets. The subtree  $T|_q$ ,  $T$  restricted to taxa in quintet set  $q$ , can be rooted on any of its seven edges. In a preprocessing step, QR computes a score for each of these seven different rootings for all trees induced on the quintets in set  $Q$ , based on a cost function. This results in  $7 \times |Q|$  computations, and therefore the preprocessing step takes  $O(k(|Q| + n))$ . Next, for every rooted version of  $T$ , QR sums up the costs of all its induced rooted trees on quintets in  $Q$  using the scores computed in the preprocessing step, and returns the rooting with the minimum overall cost. Since  $T$  can be rooted on any of its  $2n - 3$  edges, the scoring step takes  $O(n + |Q|)$  time. Therefore, the overall runtime of QR when using an  $O(n)$  sampling of quintets is  $O(nk)$ . Figure 2 shows the pipeline of QR and its individual steps.

Thus, QR provides an exact solution to the optimization problem with the following input and output:

- **Input:** An unrooted tree topology  $T$ , a set of  $k$  unrooted gene tree topologies  $\mathcal{G} = \{g_1, g_2, \dots, g_k\}$ , a set  $Q$  containing quintets of taxa from leafset  $\mathcal{L}(T)$  and a cost function  $\text{Cost}(r, \vec{u})$ .
- **Output:** Rooted tree  $R$  with topology  $T$  such that  $\sum_{q \in Q} \text{Cost}(R|_q, \vec{u}_q)$  is minimized, where  $\vec{u}_q$  is the distribution of unrooted gene tree quintets in  $\mathcal{G}|_q = \{g_1|_q, g_2|_q, \dots, g_k|_q\}$ .

**Cost Function.** The cost function  $\text{Cost}(R|_q, \vec{u}_q)$  measures the fitness of the rooted quintet tree  $R|_q$  with the distribution of the unrooted gene trees restricted to  $q$  (i.e.,  $\vec{u}_q$ ), according to the linear invariants and inequalities derived from the ADR theory. In particular, this cost function is designed to penalize a rooted tree  $R|_q$  if the estimated quintet distribution  $\vec{u}_q$  violates some of the inequalities or invariants in its partial order. To this end, a penalty term was considered for each invariant and inequality in the partial order of a 5-taxon rooted tree that is violated in a quintet distribution. The cost function was defined based on a linear combination of these penalty terms, and had the following form, where  $r$  is a 5-taxon rooted tree and  $\vec{u}$  is an estimated quintet distribution:

$$\text{Cost}(r, \vec{u}) = \underbrace{\sum_{c \in C_r} \frac{1}{|c|} \sum_{u_a, u_b \in c} |\hat{u}_a - \hat{u}_b|}_{\text{Invariants Penalty}} + \underbrace{\sum_{c > c' \in C_r} \frac{1}{|c'|} \sum_{u_a \in c, u_b \in c'} \max(0, \hat{u}_b - \hat{u}_a)}_{\text{Inequalities Penalty}}. \quad (1)$$

The normalization factors  $\frac{1}{|c|}$  and  $\frac{1}{|c'|}$  were used to reduce a topological bias that arose from differences in the sizes of the equivalence classes for each tree shape.

### 3 QR-STAR

QR-STAR is an extension to QR that has an additional step for determining the rooted shape (i.e., the rooted topology without the leaf labels) of a quintet tree, as well as an associated penalty term in its cost function. This penalty term compares the rooted shape of the 5-taxon tree, denoted by  $S(r)$ , with the rooted shape inferred by QR-STAR from the given quintet distribution, denoted by  $\hat{S}(\hat{u})$ . The motivation for this additional preprocessing step is that, as we argue in Supplementary Materials Sec. C, the cost function of QR does not guarantee statistical consistency. The cost function of QR-STAR takes the following general form

$$\begin{aligned} \text{Cost}^*(r, \vec{\hat{u}}) = & \underbrace{\sum_{c \in C_r} \sum_{u_a, u_b \in c} \alpha_{a,b} |\hat{u}_a - \hat{u}_b|}_{\text{Invariants Penalty}} + \underbrace{\sum_{c > c' \in C_r} \sum_{u_a \in c, u_b \in c'} \beta_{a,b} \max(0, \hat{u}_b - \hat{u}_a)}_{\text{Inequalities Penalty}} \\ & + \underbrace{C \mathbb{1}[S(r) \neq \hat{S}(\hat{u})]}_{\text{Shape Penalty}} \end{aligned} \quad (2)$$

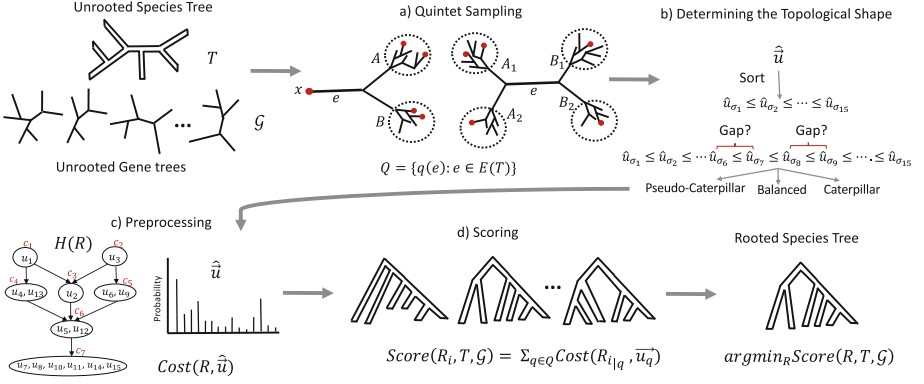
where  $\alpha_{a,b} \geq 0$  and  $\beta_{a,b}, C > 0$  are constant real numbers for all  $a, b$ <sup>2</sup>. Let  $\alpha_{\max} = \max_{a,b}(\alpha_{a,b})$  and  $\beta_{\min} = \min_{a,b}(\beta_{a,b})$  where  $a, b$  ranges over all pairs of indices  $a, b$  used in the penalty terms in Eq. 2.

Each of the 105 rooted binary trees on a given set of 5 leaves have a unique set of inequalities and invariants that can be derived from the ADR theory. The cost function in Eq. 2 considers a penalty term for these inequalities and invariants as well as the shape of the tree, so that  $\text{Cost}^*(r, \vec{\hat{u}})$  is minimized for a rooted 5-taxon tree  $r$  that best describes the given estimated quintet distribution.

#### 3.1 Determining the Rooted Shape

Model 5-taxon species trees with different rooted shapes (i.e., caterpillar, balanced, pseudo-caterpillar) define equivalence classes with different class sizes on the unrooted gene tree probability distribution. These class sizes can be used to determine the unlabeled shape of a rooted tree, when given the *true* gene tree probability distribution. For example, the size of the equivalence class with the smallest gene tree probabilities is 8 for the pseudo-caterpillar trees and 6 for balanced or caterpillar trees. Therefore, the size of the equivalence class corresponding to the minimal element in the partial order can differentiate a pseudo-caterpillar tree from other tree shapes. Moreover, both caterpillar and

<sup>2</sup> Refer to Remark 1 for why  $\alpha_{a,b}$  does not need to be strictly positive.



**Fig. 2. Pipeline of QR and QR-STAR.** The input is an unrooted species tree  $T$  and set of unrooted gene trees  $\mathcal{G}$  on the same leafset. a) The sampling step selects a set  $Q$  of quintets from the leafset of  $T$  (shown is the linear encoding sampling). b) An additional step in QR-STAR that determines the rooted shape for each selected quintet. c) The preprocessing step computes a cost for each of the seven possible rootings of each selected quintet. d) The scoring step computes a score for each rooted tree in the search space based on the costs computed in the preprocessing step, and returns a rooting of  $T$  with minimum score.

balanced trees have a unique class with the second smallest probability, which is of size 2 for caterpillar trees and 4 for balanced trees and this can be used to differentiate a caterpillar tree from a balanced tree. This approach is used in Theorem 9 in [2] for establishing the identifiability of rooted 5-taxon trees from unrooted gene trees.

However, given an *estimated* gene tree distribution, it is likely that none of the invariants derived from the ADR theory exactly hold, and so the class sizes cannot be directly determined and the approach above cannot be used as is to infer the shape of a rooted quintet. Here we propose a simple modification for determining the rooted shape of a tree from the estimated distribution of unrooted gene trees, by looking for significant gaps between quintet gene tree probabilities.

Let  $T$  be the unrooted species tree with  $n \geq 5$  leaves given to QR-STAR and  $q$  be a quintet of taxa from  $\mathcal{L}(T)$ . Let  $\vec{u}$  be the quintet distribution estimated from input gene trees induced on taxa in set  $q$ . QR-STAR first sorts  $\vec{u}$  in ascending order to get  $\hat{u}_{\sigma_1} \leq \hat{u}_{\sigma_2} \leq \dots \leq \hat{u}_{\sigma_{15}}$ . Let  $A(k) = \sqrt{\frac{2}{k} \ln(30|Q|k)}$  (refer to Lemma 4 for the derivation of  $A(k)$ ), where  $k$  is the number of input gene trees and  $|Q|$  is the size of the set of sampled quintets in QR-STAR (this depends on the number  $n$  of taxa and is assumed fixed), and note that  $\lim_{k \rightarrow \infty} A(k) = 0$ . The first step of QR-STAR computes an estimate of the rooted shape of a quintet  $q$ , denoted by  $\hat{S}(\hat{u})$  in Eq. 2, as follows:

- estimate the rooted shape  $\hat{S}(\hat{u})$  as pseudo-caterpillar if  $\hat{u}_{\sigma_7} - \hat{u}_{\sigma_6} < A(k)$ ;
- estimate the rooted shape  $\hat{S}(\hat{u})$  as balanced if  $\hat{u}_{\sigma_7} - \hat{u}_{\sigma_6} \geq A(k)$  and  $\hat{u}_{\sigma_9} - \hat{u}_{\sigma_8} < A(k)$ ;
- estimate the rooted shape  $\hat{S}(\hat{u})$  as caterpillar if  $\hat{u}_{\sigma_7} - \hat{u}_{\sigma_6} \geq A(k)$  and  $\hat{u}_{\sigma_9} - \hat{u}_{\sigma_8} \geq A(k)$ .

The runtime of QR-STAR is the same as QR, as determining the topological shape for each quintet is done in constant time, so that the overall runtime remains  $O(nk)$  when a linear sampling of quintets is used.

## 4 Theoretical Results

In this section, we provide the main theoretical results, starting with a series of lemmas and theorems that will be used in the proof of statistical consistency of QR-STAR in Theorem 2. Throughout this paper, we assume that discordance between species trees and gene trees is solely due to ILS. In establishing statistical consistency, we assume that input gene trees are true gene trees and, thus, have no gene tree estimation error. If not otherwise specified, all trees are assumed to be fully resolved (i.e., binary). Due to space constraints, the proofs are provided in Supplementary Materials Section B. We begin with some definitions and key observations.

**Definition 1 (Path length parameter).** *Let  $R$  be an MSC model species tree. Let  $f(R)$  be the length of the shortest internal branch of  $R$  and  $g(R)$  be the length of the longest internal path (i.e., a path formed from only the internal branches) of  $R$ . We define the path length parameter of  $R$  as*

$$h(R) = \frac{1}{18} e^{-3g(R)} (1 - e^{-f(R)})^2 \quad (3)$$

Note that  $h(R) \in (0, \frac{1}{18})$  since  $\exp(-x) \in (0, 1)$  for all  $x > 0$  and the branch lengths have positive values. The formula for Eq. 3 is derived from the proof of Lemma 2 in Supplementary Materials Sec. B.

**Lemma 1.** *Let  $R$  be an MSC model species tree with  $n \geq 5$  leaves and  $q$  be an arbitrary set of 5 leaves from  $\mathcal{L}(R)$ . Then  $h(R|_q) \geq h(R)$  where  $R|_q$  is the rooted tree  $R$  restricted to taxa in set  $q$ .*

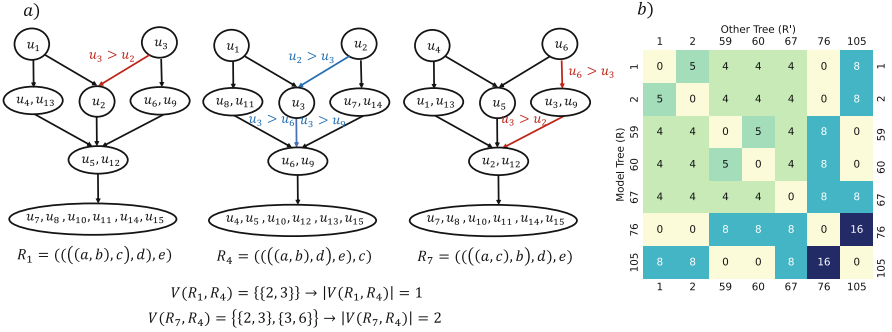
**Lemma 2.** *Let  $R$  be an MSC model species tree with 5 leaves and internal branch lengths  $x, y$ , and  $z$ . Let  $\vec{u}$  be the probability distribution that  $R$  defines on the unrooted 5-taxon gene tree topologies. If  $\vec{\hat{u}}$  is an estimate of  $\vec{u}$  such that given  $\epsilon > 0$ , we have  $|\hat{u}_i - u_i| < \epsilon$  for all  $1 \leq i \leq 15$ , then the following inequality holds:*

$$\forall_{c>c' \in C_R} \forall_{u_a \in c, u_b \in c'} : \hat{u}_a - \hat{u}_b > h(R) - 2\epsilon. \quad (4)$$

**Definition 2.** For a 5-taxon rooted tree  $R$ , we define  $I_R$  as the set of ordered pairs  $(i, j)$ ,  $1 \leq i \neq j \leq 15$ , corresponding to inequalities in the form  $u_i > u_j$  defined according to the partial order of  $R$ . The inequalities that are a result of transitivity (i.e.  $u_i > u_j$  and  $u_j > u_k$  implies  $u_i > u_k$ ) are not included in  $I_R$ .

**Definition 3.** Let  $V(R, R')$  be the set of violated inequalities of two rooted 5-taxon trees  $R$  and  $R'$ , i.e., all pairs  $\{i, j\}$  such that  $(i, j) \in I_R$  and  $(j, i) \in I_{R'}$ .

Figure 3a shows an example of  $V(R, R')$  computed for caterpillar trees and Fig. 3b is a heatmap showing the function  $|V(R, R')|$  computed for the seven possible rootings of an unrooted quintet tree. The set  $V(R, R')$  can be easily computed from  $I_R$  and  $I_{R'}$  for all pairs of rooted 5-taxon trees, and  $I_R$  is derived from the ADR theory for all 105 5-taxon rooted trees in the Supplementary Materials, Sec. S2 in [30].



**Fig. 3. Conflicting inequality penalty terms between rooted 5-taxon species trees.** a) Set of violated inequality penalty terms in the partial orders of  $R_1$  and  $R_7$  with respect to  $R_4$ , which are all caterpillar trees. The red edges show violations of inequalities in tree  $R_4$ , highlighted in blue. b) Heatmap showing the number of pairwise violated penalty terms (function  $|V(R, R')|$ ) of seven possible rooted trees having unrooted topology with bipartitions  $ab|cde$  and  $abc|de$ . The dark colors indicate more violations, and the lightest color corresponds to no violations ( $|V(R, R')| = 0$ ). (Color figure online)

**Lemma 3.** (a) For 5-taxon binary rooted trees  $R$  and  $R'$  with the same rooted shape, the set  $V(R, R')$  is always non-empty. (b) For each balanced tree  $B$ , there exist two caterpillar trees  $C_1$  and  $C_2$  such that  $V(B, C_i) = \emptyset$ .

#### 4.1 Statistical Consistency

In this section, we establish statistical consistency for QR-STAR under the MSC and provide the sufficient condition for a set of sampled quintets to lead to consistency. That is, we prove that as the number of input true gene trees increases,

the probability that QR-STAR and its variants correctly root the given unrooted species tree converges to 1. We first prove statistical consistency for QR-STAR when the model tree has only five taxa in Theorem 1 and then extend the proofs to trees with arbitrary numbers of taxa in Theorem 2. The main idea of the proof of consistency for 5-taxon trees is that we show as the number of input gene trees increases, the cost of the true rooted tree becomes arbitrarily close to zero, but the cost of any other rooted tree is bounded away from zero, where the bound depends on the path length parameter of the model tree,  $h(R)$ .

**Lemma 4.** *Let  $R$  be an MSC model species tree with  $n \geq 5$  leaves and  $Q$  be a set of quintets of taxa from  $\mathcal{L}(R)$ . Given  $\delta > 0$  and  $k > 0$  unrooted gene tree topologies, the following inequality holds, where  $A_\delta(k) = \sqrt{\frac{2}{k} \ln(\frac{30|Q|}{\delta})}$*

$$\mathbb{P} \left( \forall q \in Q \forall 1 \leq i \leq 15 |(\hat{u}_q)_i - (u_q)_i| < \frac{A_\delta(k)}{2} \right) \geq 1 - \delta. \quad (5)$$

Setting  $\delta = \frac{1}{k}$  in Eq. 5, we get  $A(k) = \sqrt{\frac{2}{k} \ln(30|Q|k)}$ , which is the bound that is used for determining the rooted shape of each quintet in the first step of QR-STAR as well as the proofs of statistical consistency. When  $R$  has only five taxa,  $A(k)$  becomes  $\sqrt{\frac{2}{k} \ln(30k)}$ , as  $Q$  can only contain one quintet.

**Lemma 5 (Correct determination of rooted shape).** *Let  $R$  be a 5-taxon model species tree and  $\vec{u}$  be the probability distribution that it defines on the unrooted 5-taxon gene tree topologies. There is an integer  $k > 0$  such that if we are given at least  $k$  unrooted gene trees drawn i.i.d. from the distribution  $\vec{u}$ , the first step of QR-STAR will correctly determine the rooted shape of  $R$  with probability at least  $1 - \frac{1}{k}$ .*

**Lemma 6 (Upper bound on the cost of the model tree).** *Let  $R$  be a 5-taxon model species tree and  $\vec{u}$  be the probability distribution that it defines on the unrooted 5-taxon gene tree topologies. There is an integer  $k > 0$  such that if we are given at least  $k$  unrooted gene trees drawn i.i.d. from distribution  $\vec{u}$ , then  $\text{Cost}^*(R, \vec{u})$  is less than  $31\alpha_{\max}A(k)$  with probability at least  $1 - \frac{1}{k}$ .*

**Theorem 1 (Statistical Consistency of QR-STAR for 5-taxon trees).** *Let  $R$  be a 5-taxon model species tree and  $\vec{u}$  be the distribution that it defines on the unrooted 5-taxon gene tree topologies. Given a set  $\mathcal{G}$  of unrooted true quintet gene trees drawn i.i.d. from  $\vec{u}$ , QR-STAR is a statistically consistent estimator of  $R$  under the MSC.*

*Remark 1.* Note that when  $\alpha_{\max} = 0$ , meaning that the invariant penalty terms are removed from the cost function, the cost of the true tree would become exactly zero according to the proof of Lemma 6, and the cost of any other tree would be positive when  $k$  is large enough so that the conditions of Theorem 1 hold. Hence in this case, the condition in Eq. 7 (see full version of the paper) will reduce to  $A(k) < \frac{1}{2}h(R)$ .

*Remark 2.* Note that Lemma 3(a) holds for *all* pairs of 5-taxon rooted trees with the same rooted shape and with different permutations of the leaf-labeling, regardless of whether they have the same unrooted topology or not. Due to this property, it is possible to differentiate all pairs of 5-taxon rooted trees in a statistically consistent manner with the cost function of QR-STAR, without prior knowledge about the unrooted tree topology, and hence Theorem 1 does not assume that the unrooted topology is given as input.

The next lemma and theorem extend the proof of statistical consistency to trees with  $n > 5$  taxa. The linear encoding of a tree  $T$  by quintets is defined in Supplementary Materials Section A.

**Lemma 7 (Identifiability of the root from the linear encoding).** *Let  $R$  and  $R'$  be rooted trees with unrooted topology  $T$  and distinct roots. Let  $Q_{LE}(T)$  be the set of quintets of leaves in a linear encoding of  $T$ . There is at least one quintet of taxa  $q \in Q_{LE}(T)$  so that  $R|_q$  and  $R'|_q$  have different rooted topologies.*

Lemma 7 states that no two distinct rooted trees with topology  $T$  induce the same set of rooted quintet trees on quintets of taxa in set  $Q_{LE}(T)$ . Clearly, the same is true for any superset  $Q$  such that  $Q_{LE}(T) \subseteq Q$ , including the set  $Q_5$  of all quintets of taxa on the leafset of  $T$ . There might also be other quintet sets that are not a superset of  $Q_{LE}(T)$ , but have the property that no two rooted versions of  $T$  define the same set of rooted quintets on their elements. We generalize the proof of consistency to all set of sampled quintets with this property.

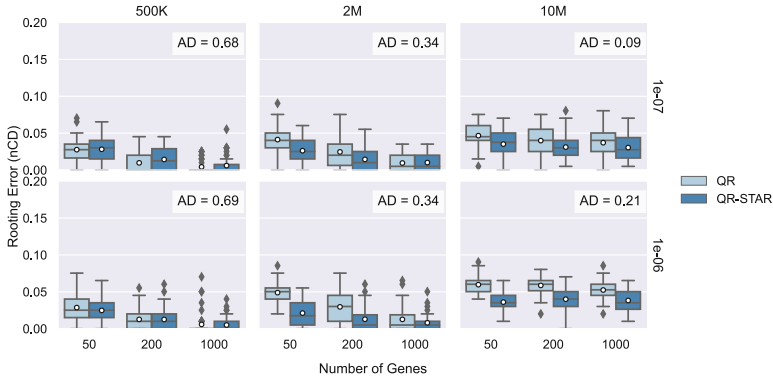
**Definition 4.** *Let  $T$  be an unrooted tree and  $Q$  be a set of quintets of taxa from  $\mathcal{L}(T)$ . We say  $Q$  is “root-identifying” if every rooted tree  $R$  with topology  $T$  is identifiable from  $T$  and the set of rooted quintet trees in  $\{R|_q : q \in Q\}$ , i.e., no two rooted trees with topology  $T$  induce the same set of rooted quintet trees on  $Q$ .*

**Theorem 2 (Statistical Consistency of QR-STAR).** *Let  $R$  be an MSC model species tree with  $n \geq 5$  leaves and let  $T$  denote its unrooted topology. Given  $T$  and a set  $\mathcal{G}$  of unrooted true gene trees on the leafset  $\mathcal{L}(T)$ , QR-STAR is a statistically consistent estimator of the rooted version of  $T$  under the MSC, if the set of sampled quintets  $Q$  is root-identifying.*

## 5 Experimental Study

We performed an experimental study on simulated datasets to explore the parameter space of QR-STAR on a training dataset, and then compared its accuracy to QR on a test dataset. We used the 101-taxon simulated datasets from [34] as our training data, which had model conditions characterized by four levels of gene tree estimation error (GTEE) ranging from 0.23 to 0.55 (measured in terms of normalized Robinson-Foulds (RF) [26] distance between true and estimated gene trees) for 1000 genes. The normalized RF distance between the model species tree and





**Fig. 4. Rooting the model species tree with estimated gene trees on S200 datasets.** Comparison between QR and QR-STAR in terms of rooting error (nCD) for rooting the true unrooted species tree topology using estimated gene trees (GTEE levels vary from 0.22 (for low ILS) to 0.49 (for high ILS)) on the 201-taxon datasets from [20] with 50 replicates in each model condition. The columns show tree height (500K for high ILS, 2M for moderate ILS, and 10M for low ILS), and the rows show speciation rate ( $1e-06$  or  $1e-07$ ).

true gene trees (denoted average distance, or AD) in this dataset was 0.46, which indicates moderate ILS. For the test data (see Table D1 in the Supplementary Materials for empirical statistics), we used a set of 201-taxon simulated datasets from [20]; these are characterized by two different speciation rates and three tree heights (500K for high ILS, 2M for moderate ILS, and 10M for low ILS) and three numbers of genes for each combination of speciation rate and tree height. GTEE levels on the test data varied from 0.22 (for low ILS) to 0.49 (for high ILS). The AD levels ranged from 0.09 (for the 10M,  $1e-07$  condition) to 0.69 (for the 500K,  $1e-06$  condition). The number of replicates for each model condition for both the training and test datasets was 50.

We measured the error in the rooted species tree in terms of average normalized clade distance (nCD) [30], which is an extension of RF error for rooted trees. For our training experiment, we only rooted the true species tree topology to directly observe the rooting error. In our test experiments, we rooted both the model species tree and estimated species tree, as produced by ASTRAL, using both true and estimated gene trees (which were estimated using FastTree [23]). Additional information about the simulation study, datasets, and software commands are provided in Supplementary Materials Section D.

In our training experiments, we explored the impact of the shape coefficient  $C$  and the ratio  $\frac{\alpha_{max}}{\beta_{min}}$  (that describes the relative impact of invariants and inequalities) on the accuracy of QR-STAR. Results for the training experiments (provided in Supplementary Materials Sec. E1) show that there are wide ranges of settings for the algorithmic parameters that provide the best accuracy. We used these training results and theoretical considerations related to sample

complexity of QR-STAR to set the algorithmic parameters to  $C = 1e-02$  and  $\frac{\alpha_{max}}{\beta_{min}} = 0$ .

Figure 4 shows a comparison between QR and QR-STAR in terms of rooting error for rooting the model species tree topology using estimated gene trees on the test datasets. Increasing the ILS level (by reducing tree height) decreases the rooting error, and increasing the number of genes also generally reduces rooting error (although much less under the lowest ILS level where tree height is 10M). To understand the impact of ILS in Fig. 4, note that the true species tree is being rooted and so ILS will not impact species tree estimation accuracy. However, the level of ILS impacts information about rooting location, which comes from the distribution of gene tree topologies. Thus, with lower ILS, it is likely that many gene trees that have low probability of appearing will not appear in the input. In this case, some estimates of quintet probabilities would become zero, and it may not be possible to differentiate some of the rooted quintets using the inequality and invariants derived from the ADR theory. In the extreme case, when there is no discordance, there will be only one quintet gene tree with non-zero probability, and the identifiability theorem in [2] would not hold and it becomes impossible to find the root. This trend can be compared to the impact of ILS level on the problem of *estimating* the unrooted topology of the species tree, where increases in ILS generally lead to increases in error [19–21].

A comparison between QR and QR-STAR shows that QR-STAR generally matched or improved on QR; the only exception was for the high ILS conditions, where the two methods were very close but with perhaps a small advantage to QR. On these high ILS conditions, however, GTEE is also large, and QR-STAR is more accurate than QR when used with true gene trees, even under high ILS (Supplementary Materials Sec. E). Hence, the issue is likely to be high GTEE rather than high ILS, suggesting that QR-STAR is slightly more affected by GTEE compared to QR.

## 6 Conclusion

In this work we presented QR-STAR, a polynomial time statistically consistent method for rooting species trees under the multispecies coalescent model. QR-STAR is an extension to QR, a method for rooting species trees introduced in [30]. QR-STAR differs from QR in that it has an additional step for determining the topological shape of each unrooted quintet selected in the QR algorithm, and incorporates the knowledge of this shape in its cost function, alongside the invariants and inequalities previously used in QR. We also showed that the statistical consistency for QR-STAR holds for a larger family of optimization problems based on cost functions and sampling methods.

To the best of our knowledge, this is the first work that established the statistical consistency of any method for rooting species trees under a model that incorporates gene tree heterogeneity. It remains to be investigated whether other rooting methods can also be proven statistically consistent under models of gene evolution inside species trees, such as the MSC or models of GDL. For example,

STRIDE [4] and DISCO+QR [33] are methods that have been developed for rooting species trees from gene family trees, where genes evolve under gene duplication and loss (GDL); however, it is not known whether these methods are statistically consistent under any GDL model.

Our simulation study showed as well that QR-STAR generally improved on QR in a wide range of model conditions. Given that QR itself improved on other methods for rooting species trees (as shown in [30]), this experimental study suggests that QR-STAR may be a useful tool for rooting species trees when gene tree discordance due to ILS is present.

This study suggests several directions for future research. For example, we proved statistical consistency for one class of cost functions, which was a linear combination of the invariant, inequality and shape penalty terms; however, cost functions in other forms could also be explored and proven statistically consistent. The proof of Theorem 1 suggests that the sample complexity of QR-STAR depends on the function  $h(R)$ , which is based on both the length of the shortest branch and the longest path in the model tree. This suggests that having very short or very long branches can both confound rooting under ILS, which is also suggested in previous studies [1, 2]. This is unlike what is known for species tree estimation methods such as ASTRAL, where the sample complexity is only affected by the shortest branch of the model tree [3, 28], and trees with long branches are easier to estimate.

Another theoretical direction is the construction of the rooted species tree directly from the unrooted gene trees. As explained in Remark 2, the proof of consistency of QR-STAR for 5-taxon trees does not depend upon the knowledge of the unrooted tree topology; this suggests that it is possible to estimate the rooted topology of the species tree in a statistically consistency manner *directly* from unrooted gene tree topologies. Future work could focus on developing statistically consistent methods for this problem, which is significantly harder than the problem of rooting a given tree.

There are also directions for improving empirical results. An important consideration in designing a good cost function is its empirical performance, as many cost functions can lead to statistical consistency but may not provide accurate estimations of the rooted tree in practice (see Figures E1 and E2 in the Supplementary Materials). One potential direction is to incorporate estimated branch lengths, whether of the gene trees or the unrooted species tree, into the rooting procedure.

**Acknowledgments.** SR was supported by NSF grants DMS-1902892, DMS1916378 and DMS-2023239 (TRIPODS Phase II), as well as a Vilas Associates Award. TW was supported by the Grainger Foundation. SR thanks Cécile Ané and her group for helpful discussions. YT thanks Mohammed El-Kebir for helpful suggestions on an earlier version of this work. The authors thank the reviewers for their feedback.

## References

1. Alanzi, A.R., Degnan, J.H.: Inferring rooted species trees from unrooted gene trees using approximate Bayesian computation. *Mol. Phylogenet. Evol.* **116**, 13–24 (2017). <https://doi.org/10.1016/j.ympev.2017.07.017>
2. Allman, E.S., Degnan, J.H., Rhodes, J.A.: Identifying the rooted species tree from the distribution of unrooted gene trees under the coalescent. *J. Math. Biol.* **62**(6), 833–862 (2011). <https://doi.org/10.1007/s00285-010-0355-7>
3. Chan, Y., Li, Q., Scornavacca, C.: The large-sample asymptotic behaviour of quartet-based summary methods for species tree inference. *J. Math. Biol.* **85**(3), 1–22 (2022). <https://doi.org/10.1007/s00285-022-01786-4>
4. Emms, D.M., Kelly, S.: STRIDE: species tree root inference from gene duplication events. *Mol. Biol. Evol.* **34**(12), 3267–3278 (2017)
5. Felsenstein, J.: Cases in which parsimony or compatibility methods will be positively misleading. *Syst. Zool.* **27**(4), 401–410 (1978)
6. Graham, S.W., Olmstead, R.G., Barrett, S.C.: Rooting phylogenetic trees with distant outgroups: a case study from the commelinoid monocots. *Mol. Biol. Evol.* **19**(10), 1769–1781 (2002)
7. Skarp-de Haan, C.: Comparative genomics of unintrogressed *Campylobacter coli* clades 2 and 3. *BMC Genomics* **15**(1), 1–14 (2014). <https://doi.org/10.1186/1471-2164-15-129>
8. Hess, P.N., De Moraes Russo, C.A.: An empirical test of the midpoint rooting method. *Biol. J. Linn. Soc.* **92**(4), 669–674 (2007)
9. Holland, B., Penny, D., Hendy, M.: Outgroup misplacement and phylogenetic inaccuracy under a molecular clock—a simulation study. *Syst. Biol.* **52**(2), 229–238 (2003). <https://doi.org/10.1080/10635150390192771>
10. Hudson, R.R.: Testing the constant-rate neutral allele model with protein sequence data. *Evolution* 203–217 (1983). <https://doi.org/10.2307/2408186>
11. Jun, S.R., et al.: Ebolavirus comparative genomics. *FEMS Microbiol. Rev.* **39**(5), 764–778 (2015)
12. Larget, B.R., Kotha, S.K., Dewey, C.N., Ané, C.: BUCKy: gene tree/species tree reconciliation with Bayesian concordance analysis. *Bioinformatics* **26**(22), 2910–2911 (2010). <https://doi.org/10.1093/bioinformatics/btq539>
13. Li, C., Matthes-Rosana, K.A., Garcia, M., Naylor, G.J.: Phylogenetics of Chondrichthyes and the problem of rooting phylogenies with distant outgroups. *Mol. Phylogenet. Evol.* **63**(2), 365–373 (2012)
14. Liu, L., Yu, L., Edwards, S.V.: A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. *BMC Evol. Biol.* **10**(1), 1–18 (2010). <https://doi.org/10.1186/1471-2148-10-302>
15. Maddison, W.P.: Gene trees in species trees. *Syst. Biol.* **46**(3), 523–536 (1997). <https://doi.org/10.1093/sysbio/46.3.523>
16. Maddison, W.P., Donoghue, M.J., Maddison, D.R.: Outgroup analysis and parsimony. *Syst. Biol.* **33**(1), 83–103 (1984)
17. Mahbub, M., Wahab, Z., Reaz, R., Rahman, M.S., Bayzid, M.S.: wQFM: highly accurate genome-scale species tree estimation from weighted quartets. *Bioinformatics* **37**(21), 3734–3743 (2021). <https://doi.org/10.1093/bioinformatics/btab428>
18. Mai, U., Sayyari, E., Mirarab, S.: Minimum variance rooting of phylogenetic trees and implications for species tree reconstruction. *PLoS ONE* **12**(8), e0182238 (2017)
19. Mirarab, S., Reaz, R., Bayzid, M.S., Zimmermann, T., Swenson, M.S., Warnow, T.: ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics* **30**(17), i541–i548 (2014). <https://doi.org/10.1093/bioinformatics/btu462>

20. 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). <https://doi.org/10.1093/bioinformatics/btv234>
21. Molloy, E.K., Warnow, T.: To include or not to include: the impact of gene filtering on species tree estimation methods. *Syst. Biol.* **67**(2), 285–303 (2018)
22. Posada, D.: Phylogenomics for systematic biology. *Syst. Biol.* **65**(3), 353–356 (2016). <https://doi.org/10.1093/sysbio/syw027>
23. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree 2-approximately maximum-likelihood trees for large alignments. *PLoS ONE* **5**(3), e9490 (2010)
24. Rabiee, M., Mirarab, S.: QuCo: quartet-based co-estimation of species trees and gene trees. *Bioinformatics* **38**(Supplement\_1), i413–i421 (2022)
25. Renner, S.S., Grimm, G.W., Schneeweiss, G.M., Stuessy, T.F., Ricklefs, R.E.: Rooting and dating maples (*Acer*) with an uncorrelated-rates molecular clock: implications for North American/Asian disjunctions. *Syst. Biol.* **57**(5), 795–808 (2008). <https://doi.org/10.1080/10635150802422282>
26. Robinson, D.F., Foulds, L.R.: Comparison of phylogenetic trees. *Math. Biosci.* **53**(1–2), 131–147 (1981). [https://doi.org/10.1016/0025-5564\(81\)90043-2](https://doi.org/10.1016/0025-5564(81)90043-2)
27. Rosenberg, N.A.: Counting coalescent histories. *J. Comput. Biol.* **14**(3), 360–377 (2007). <https://doi.org/10.1089/cmb.2006.0109>
28. Shekhar, S., Roch, S., Mirarab, S.: Species tree estimation using ASTRAL: how many genes are enough? *IEEE/ACM Trans. Comput. Biol. Bioinf.* **15**(5), 1738–1747 (2017). <https://doi.org/10.1109/TCBB.2017.2757930>
29. Simmons, M.P., Springer, M.S., Gatesy, J.: Gene-tree misrooting drives conflicts in phylogenomic coalescent analyses of palaeognath birds. *Mol. Phylogenet. Evol.* **167**, 107344 (2022). <https://doi.org/10.1016/j.ympev.2021.107344>
30. Tabatabaee, Y., Sarker, K., Warnow, T.: Quintet Rooting: rooting species trees under the multi-species coalescent model. *Bioinformatics* **38**(Supplement\_1), i109–i117 (2022). <https://doi.org/10.1093/bioinformatics/btac224>
31. Tria, F.D.K., Landan, G., Dagan, T.: Phylogenetic rooting using minimal ancestor deviation. *Nat. Ecol. Evol.* **1**(1), 1–7 (2017). <https://doi.org/10.1038/s41559-017-0193>
32. Wascher, M., Kubatko, L.: Consistency of SVDQuartets and maximum likelihood for coalescent-based species tree estimation. *Syst. Biol.* **70**(1), 33–48 (2021). <https://doi.org/10.1093/sysbio/syaa039>
33. Willson, J., Tabatabaee, Y., Liu, B., Warnow, T.: DISCO+QR: rooting species trees in the presence of GDL and ILS. *Bioinform. Adv.* **3**(1), vbad015 (2023). <https://doi.org/10.1093/bioadv/vbad015>
34. Zhang, C., Rabiee, M., Sayyari, E., Mirarab, S.: ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinform.* **19**(6), 15–30 (2018). <https://doi.org/10.1186/s12859-018-2129-y>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Sequence to Graph Alignment Using Gap-Sensitive Co-linear Chaining

Ghanshyam Chandra and Chirag Jain<sup>(✉)</sup>

Department of Computational and Data Sciences, Indian Institute of Science,  
Bangalore 560012, KA, India  
{ghanshyamc,chirag}@iisc.ac.in

**Abstract.** Co-linear chaining is a widely used technique in sequence alignment tools that follow seed-filter-extend methodology. It is a mathematically rigorous approach to combine short exact matches. For co-linear chaining between two sequences, efficient subquadratic-time chaining algorithms are well-known for linear, concave and convex gap cost functions [Eppstein *et al.* JACM'92]. However, developing extensions of chaining algorithms for directed acyclic graphs (DAGs) has been challenging. Recently, a new sparse dynamic programming framework was introduced that exploits small path cover of pangenome reference DAGs, and enables efficient chaining [Makinen *et al.* TALG'19, RECOMB'18]. However, the underlying problem formulation did not consider gap cost which makes chaining less effective in practice. To address this, we develop novel problem formulations and optimal chaining algorithms that support a variety of gap cost functions. We demonstrate empirically the ability of our provably-good chaining implementation to align long reads more precisely in comparison to existing aligners. For mapping simulated long reads from human genome to a pangenome DAG of 95 human haplotypes, we achieve 98.7% precision while leaving < 2% reads unmapped.

**Implementation:** <https://github.com/at-cg/minichain>.

**Keywords:** Variation graph · Sparse dynamic programming · Minimum path cover · Pangenome

## 1 Introduction

A significant genetic variation rate among genomes of unrelated humans, plus the growing availability of high-quality human genome assemblies, has accelerated computational efforts to use pangenome reference graphs for common genomic analyses [25, 41, 42]. The latest version of industry-standard DRAGEN software by Illumina now uses a pangenome graph for mapping reads in highly polymorphic regions of a human genome [15]. For surveys of the recent algorithmic developments in this area, see [2, 7, 11, 34]. Among the many computational tasks associated with pangenome graphs, sequence-to-graph alignment remains a core computational problem. Accurate alignments are required for variation analysis and construction of pangenome graph from multiple genomes [10, 23]. Sequence-to-graph

alignment is also useful in other applications including genome assembly [14] and long-read error correction [40].

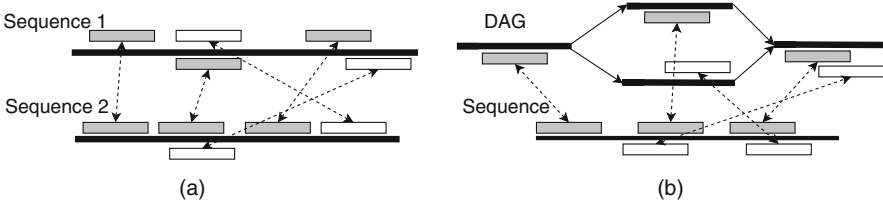
Suppose a pangenome graph is represented as a character labeled DAG  $G(V, E)$  where each vertex  $v \in V$  is labeled with a character from alphabet  $\{A, C, G, T\}$ . The sequence-to-DAG alignment problem seeks a path in  $G$  that spells a string with minimum edit distance from the input query sequence. An  $O(m|E|)$  time algorithm for this problem has long been known, where  $m$  is the length of input sequence [30]. Conditioned on Strong Exponential Time Hypothesis (SETH), the  $O(m|E|)$  algorithm is already worst-case optimal up to sub-polynomial improvements because algorithms for computing edit distance in strongly sub-quadratic time cannot exist under SETH [3]. As a result, heuristics must be used for alignment of high-throughput sequencing data against large DAGs to obtain approximate solutions in less time and space.

All practical long read to DAG aligners that scale to large genomes rely on seed-filter-extend methodology [9, 23, 26, 28, 35]. The first step is to find a set of *anchors* which indicate short exact matches, e.g.,  $k$ -mer or minimizer matches, between substrings of a sequence to subpaths in a DAG. This is followed by a clustering step that identifies promising subsets of anchors which should be kept within the alignments. Different aligners implement this step in different ways. *Co-linear chaining* is a mathematically rigorous approach to do clustering of anchors. It is well studied for the case of sequence-to-sequence alignment [1, 12, 13, 17, 27, 29, 33], and is widely used in present-day long read to reference sequence aligners [19, 22, 36, 38, 39]. For the sequence-to-sequence alignment case, the input to the chaining problem is a set of  $N$  weighted anchors where each anchor is a pair of intervals in the two sequences that match exactly. A *chain* is defined as an ordered subset of anchors such that their intervals appear in increasing order in both sequences (Fig. 1a). The desired output of the co-linear chaining problem is the chain with maximum score where score of a chain is calculated by the sum of weights of the anchors in the chain minus the penalty corresponding to gaps between adjacent anchors. For linear gap costs, this problem is solvable in  $O(N \log N)$  time by using range-search queries [1].

Solving chaining problem for sequence-to-DAG alignment remained open until Makinen *et al.* [28] introduced a framework that enables sparse dynamic programming on DAGs. Suppose  $K$  denotes cardinality of a minimum-sized set of paths such that every vertex is covered by at least one path. The algorithm in [28] works by mimicking the sequence-to-sequence chaining algorithm on each path of the *minimum path cover*. After a polynomial-time indexing of the DAG, their algorithm requires  $O(KN \log N + K|V|)$  time for chaining. Parameterizing the time complexity in terms of  $K$  is useful because  $K$  is expected to be small for pangenome DAGs. This result was further improved in [26] with an  $O(KN \log KN)$  time algorithm. However, the problem formulations in these works did not include gap cost. Without penalizing gaps, chaining is less effective [17]. A challenge in enforcing gap cost is that measuring gap between two loci in a DAG is not a simple arithmetic operation like in a sequence [21].

We present novel co-linear chaining problem formulations for sequence-to-DAG alignment that penalize gaps, and we develop efficient algorithms to solve





**Fig. 1.** Illustration of co-linear chaining for (a) sequence-to-sequence and (b) sequence-to-DAG alignment. It is assumed that vertices of DAG are labeled with strings. Pairs of rectangles joined by dotted arrows denote anchors (exact matches). A subset of these anchors that form a valid chain are shown in gray.

them. We carefully design gap cost functions such that they enable us to adapt the sparse dynamic programming framework of Makinen *et al.* [28], and solve the chaining problem optimally in  $O(KN \log KN)$  time. We implemented and benchmarked one of our proposed algorithms to demonstrate scalability and accuracy gains. Our experiments used human pangenome DAGs built by using 94 high quality *de novo* haplotype assemblies provided by the Human Pangenome Reference Consortium [25] and CHM13 human genome assembly provided by the Telomere-to-Telomere consortium [31]. Using a simulated long read dataset with  $0.5\times$  coverage, we demonstrate that our implementation achieves the highest read mapping precision (98.7%) among the existing methods (Minigraph: 98.0%, GraphAligner: 97.0% and GraphChainer: 95.1%). In this experiment, our implementation used 24 min and 25 GB RAM with 32 threads, demonstrating that the time and memory requirements are well within practical limits.

## 2 Concepts and Notations

### 2.1 Co-linear Chaining on Sequences Revisited

Let  $R$  and  $Q$  be two sequences over alphabet  $\Sigma = \{A, C, G, T\}$ . Let  $M[1..N]$  be an array of anchors. Each anchor is denoted using an interval pair  $([x..y], [c..d])$  with the interpretation that substring  $R[x..y]$  matches substring  $Q[c..d]$ ,  $x, y, c, d \in \mathbb{N}$ . Anchors are typically either fixed-length matches (e.g., using  $k$ -mers) or variable-length matches (e.g., maximal exact matches). Suppose function *weight* assigns weights to the anchors. The co-linear chaining problem seeks an ordered subset  $S = s_1 s_2 \dots s_p$  of anchors from  $M$  such that

- for all  $2 \leq j \leq p$ ,  $s_{j-1}$  precedes ( $\prec$ )  $s_j$ , i.e.,  $s_{j-1}.y < s_j.x$  and  $s_{j-1}.d < s_j.c$ .
- $S$  maximises chaining score, defined as  $\sum_{j=1}^p \text{weight}(s_j) - \sum_{j=2}^p \text{gap}(s_{j-1}, s_j)$ . Define  $\text{gap}(s_{j-1}, s_j)$  as  $f(\text{gap}_R(s_{j-1}, s_j), \text{gap}_Q(s_{j-1}, s_j))$ , where  $\text{gap}_R(s_{j-1}, s_j) = s_j.x - s_{j-1}.y - 1$ ,  $\text{gap}_Q(s_{j-1}, s_j) = s_j.c - s_{j-1}.d - 1$  and  $f(g_1, g_2) = g_1 + g_2$ .

The above problem can be trivially solved in  $O(N^2)$  time and  $O(N)$  space. First sort the anchors by the component  $M[\cdot].x$ , and let  $T[1..N]$  be an

integer array containing a permutation of set  $[1..N]$  which specifies the sorted order, i.e.,  $M[T[1]].x \leq M[T[2]].x \leq \dots \leq M[T[N]].x$ . Define array  $C[1..N]$  such that  $C[j]$  is used to store a partial solution, i.e., the score of an optimal chain that ends at anchor  $M[j]$ . Naturally, the final score will be obtained as  $\max_j C[j]$ . Array  $C$  can be filled by using the following dynamic programming recursion:  $C[T[j]] = \text{weight}(M[T[j]]) + \max(0, \max_{i: M[i] \prec M[T[j]]} (C[i] - \text{gap}(M[i], M[T[j]])))$ , in increasing order of  $j$ . A straight-forward way of computing  $C[T[j]]$  will need an  $O(N)$  linear scan of arrays  $C$  and  $M$ , resulting in overall  $O(N^2)$  time. However, the  $O(N^2)$  algorithm can be optimized to use  $O(N \log N)$  time by using the following search tree data structure (ref. [4]).

**Lemma 1.** *Let  $n$  be the number of leaves in a balanced binary search tree, each storing a (key, value) pair. The following operations can be supported in  $O(\log n)$  time:*

- *update( $k, \text{val}$ ): For the leaf  $w$  with key =  $k$ ,  $\text{value}(w) \leftarrow \max(\text{value}(w), \text{val})$ .*
- *RMQ( $l, r$ ): Return  $\max\{\text{value}(w) \mid l < \text{key}(w) < r\}$ . This is range maximum query.*

Moreover, given  $n$  (key, value) pairs, the balanced binary search tree can be constructed in  $O(n \log n)$  time and  $O(n)$  space.

The dynamic programming recursion for array  $C[1..N]$  can be computed more efficiently using range maximum queries [1, 12]. To achieve this, a search tree needs to be initialized, updated and queried properly (Algorithm 1). Note that  $\text{argmax}_{i: M[i] \prec M[j]} (C[i] - \text{gap}(M[i], M[j]))$  is equal to  $\text{argmax}_{i: M[i] \prec M[j]} (C[i] + M[i].y + M[i].d)$ . Accordingly, we compute optimal  $C[j]$  in Line 11 by using an  $O(\log N)$  time RMQ operation of the form  $M[i].d \in (0, M[j].c)$  that returns maximum  $C[i] + M[i].y + M[i].d$  from search tree  $\mathcal{T}$ . The algorithm performs  $N$  update and  $N$  RMQ operations over search tree  $\mathcal{T}$  of size at most  $N$ , thus solving the problem in  $O(N \log N)$  time and  $O(N)$  space.

---

**Algorithm 1:**  $O(N \log N)$  time chaining between two sequences

---

**Input:** Array of weighted anchors  $M[1..N]$   
**Output:** Array  $C[1..N]$  such that  $C[j]$  = score of an optimal chain that ends at  $M[j]$

- 1 Initialize search tree  $\mathcal{T}$  using keys  $\{M[j].d \mid 1 \leq j \leq N\}$  and values  $-\infty$
- 2 Initialize  $C[j]$  as  $\text{weight}(M[j])$ , for all  $j \in [1, N]$
- 3 /\* Create array  $Z$  that stores tuples of the form ( $\text{pos}, \text{task}, \text{anchor}$ ), where  $\text{pos} \in \mathbb{N}$ ,  $\text{anchor} \in [1, N]$  and  $\text{task} \in \{0, 1\}$ .  $\text{task}$  is either 0 or 1 for querying or updating the search tree  $\mathcal{T}$  respectively.\*/
- 4 **for**  $j \leftarrow 1$  **to**  $N$  **do**
- 5      $Z.\text{push}(M[j].x, 0, j)$
- 6      $Z.\text{push}(M[j].y, 1, j)$
- 7 **end**
- 8 **for**  $z \in Z$  in lexicographically ascending order based on the key ( $\text{pos}, \text{task}$ ) **do**
- 9      $j \leftarrow z.\text{anchor}$ ,  $wt \leftarrow \text{weight}(M[j])$
- 10     **if**  $z.\text{task} = 0$  **then**
- 11          $C[j] \leftarrow \max(C[j], wt + \mathcal{T}.\text{RMQ}(0, M[j].c) - M[j].x - M[j].c + 2)$
- 12     **else**
- 13          $\mathcal{T}.\text{update}(M[j].d, C[j] + M[j].y + M[j].d)$
- 14     **end**
- 15 **end**

---

## 2.2 Sparse Dynamic Programming on DAGs Using Minimum Path Cover

Our work builds on the work of Makinen *et al.* [28], who provided a parameterized algorithm to extend co-linear chaining on DAGs without considering gap costs. In the following, we present useful notations and a summary of their algorithm.

In a weighted string-labeled DAG  $G(V, E, \sigma)$ , function  $\sigma : V \rightarrow \Sigma^+$  labels each vertex  $v \in V$  with string  $\sigma(v)$ . Edge  $v \rightarrow u$  has length  $|\sigma(v)|$ . The length of a path in  $G$  is the sum of the lengths of the edges traversed in the path. Let  $Q \in \Sigma^+$  be a query sequence. Let  $M$  be an array of  $N$  anchors. An anchor is denoted using a 3-tuple of the form  $(v, [x..y], [c..d])$  with the interpretation that substring  $\sigma(v)[x..y]$  in DAG  $G$  matches substring  $Q[c..d]$ , where  $x, y, c, d \in \mathbb{N}$  and  $v \in V$  (Fig. 2). A *path cover* of DAG  $G(V, E)$  is a set of paths in  $G$  such that every vertex in  $V$  belongs to at least one path. A *minimum path cover* (MPC) is one having the minimum number of paths. If  $K$  denotes the size of MPC of DAG  $G$ , then MPC can be computed either in  $O(K|E| \log |V|)$  [28] or  $O(K^3|V| + |E|)$  [5] time.

To extend co-linear chaining for sequence-to-DAG alignment, we can use a search tree containing keys equal to the sequence coordinates of anchors, similar to Algorithm 1. However, the order in which the search tree should be queried and updated is not trivial with DAGs. Makinen *et al.* [28] suggested decomposing the DAG into a path cover  $\{P_1, \dots, P_K\}$ , and then performing the computation only along these paths. The algorithm uses  $K$  search trees  $\{\mathcal{T}_1, \dots, \mathcal{T}_K\}$ , one per path. Search tree  $\mathcal{T}_i$  maintains  $M[\cdot].d$  as keys and partial solutions  $C[\cdot]$  as values of all the anchors that lie on path  $P_i$ . Similar to Algorithm 1, the  $K$  search trees need to be updated and queried in a proper order. Suppose  $\mathcal{R}(v) \subseteq V$  denotes the set of vertices which can reach  $v$  using a path in  $G$ . Set  $\mathcal{R}(v)$  always includes  $v$ . Define  $last2reach(v, i)$  as the last vertex on path  $P_i$  that belongs to  $\mathcal{R}(v)$ , if one exists. Also define  $paths(v)$  as  $\{i : P_i \text{ covers } v\}$ . Naturally  $last2reach(v, i) = v$  iff  $i \in paths(v)$ . The main algorithm works by visiting vertices  $u$  of  $G$  in topological order, and executing the following two tasks:

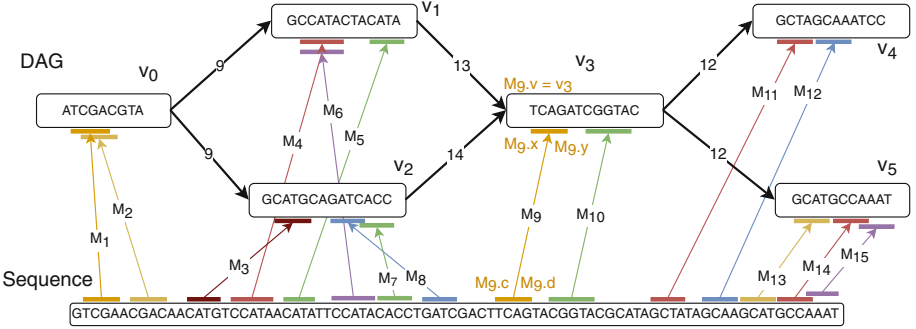
- **Compute optimal scores of all anchors in vertex  $u$ :** First, process all the anchors for which  $M[j].v = u$  in the same order that is used for co-linear chaining on two sequences (Algorithm 1). While performing an update task, update all search trees  $\mathcal{T}_i$ , for all  $i \in paths(u)$ . Similarly, while performing a range query, query search trees  $\mathcal{T}_i$  to maximize  $C[j]$ .
- **Update partial scores of selected anchors outside vertex  $u$ :** Next, for all pairs  $(w, i)$ ,  $w \in V, i \in [1, K]$  such that  $last2reach(w, i) = u$  and  $i \notin paths(w)$ , query search tree  $\mathcal{T}_i$  to update score  $C[j]$  of every anchor  $M[j]$  for which  $M[j].v = w$ .

Based on the above tasks, once the algorithm reaches  $v \in V$  in the topological ordering, the scores corresponding to anchors in vertex  $v$  would have been updated from all other vertices that reach  $v$ . A well-optimized implementation of this algorithm uses  $O(KN \log KN)$  time [26]. This result assumes that the DAG is preprocessed, i.e., path cover and  $last2reach$  information is precomputed in  $O(K^3|V| + K|E|)$  time.

### 3 Problem Formulations

We develop six problem formulations for co-linear chaining on DAGs with different gap cost functions. In each problem, we seek an ordered subset  $S = s_1 s_2 \cdots s_p$  of anchors from array  $M$  such that

- for all  $2 \leq j \leq p$ ,  $s_{j-1}$  precedes ( $\prec$ )  $s_j$ , i.e., the following three conditions are satisfied (i)  $s_{j-1}.d < s_j.c$ , (ii)  $s_{j-1}.v \in \mathcal{R}(s_j.v)$ , and (iii)  $s_{j-1}.y < s_j.x$  if  $s_{j-1}.v = s_j.v$ .
- $S$  maximizes the chaining score defined as  $\sum_{j=1}^p \text{weight}(s_j) - \sum_{j=2}^p \text{gap}(s_{j-1}, s_j)$ . Define  $\text{gap}(s_{j-1}, s_j)$  as  $f(\text{gap}_G(s_{j-1}, s_j), \text{gap}_S(s_{j-1}, s_j))$ , where functions  $\text{gap}_G$  and  $\text{gap}_S$  will be used to specify gap cost in the DAG and the query sequence respectively.



**Fig. 2.** An example showing multiple anchors as input for co-linear chaining. The DAG has a minimum path cover of size two  $\{(v_0, v_1, v_3, v_4), (v_0, v_2, v_3, v_5)\}$ . Anchors  $M_1, M_4, M_5, M_9, M_{10}, M_{11}, M_{12}$  form a valid chain. The interval coordinates of anchor  $M_9$  in the sequence and the DAG are annotated for illustration.

$\text{gap}_S(s_{j-1}, s_j)$  equals  $s_j.c - s_{j-1}.d - 1$ , i.e., the count of characters in sequence  $Q$  that occur between the two anchors. However, defining  $\text{gap}_G$  is not as straightforward because multiple paths may exist from  $s_{j-1}.v$  to  $s_j.v$ , and the correct alignment path is unknown. We formulate and solve the following problems:

**Problems 1a–1c:**  $\text{gap}_G(s_{j-1}, s_j)$  is computed by using the shortest path from  $s_{j-1}.v$  to  $s_j.v$ . Suppose  $D(v_1, v_2)$  denotes the shortest path length from vertex  $v_1$  to  $v_2$  in  $G$ . We seek the optimal chaining score when

$$\text{gap}_G(s_{j-1}, s_j) = D(s_{j-1}.v, s_j.v) + (s_j.x - s_{j-1}.y - 1).$$

The above expression calculates the count of characters in the string path between anchors  $s_{j-1}$  and  $s_j$ . Define Problems 1a, 1b and 1c using  $f(g_1, g_2) = g_1 + g_2$ ,  $f(g_1, g_2) = \max(g_1, g_2)$  and  $f(g_1, g_2) = |g_1 - g_2|$  respectively. These definitions of function  $f$  are motivated from the previous co-linear chaining formulations for sequence-to-sequence alignment [1, 29].

**Problems 2a–2c:**  $gap_G(s_{j-1}, s_j)$  is measured using a path from  $s_{j-1}.v$  to  $s_j.v$  that is chosen based on path cover  $\{P_1, \dots, P_K\}$  of DAG  $G$ . For each  $i \in paths(s_{j-1}.v)$ , consider the following path in  $G$  that starts from source  $s_{j-1}.v$  along the edges of path  $P_i$  till the middle vertex  $last2reach(s_j.v, i)$ , and finally reaches destination  $s_j.v$  by using the shortest path from  $last2reach(s_j.v, i)$  to  $s_j.v$ . Among  $|paths(s_{j-1}.v)|$  such possible paths, measure  $gap_G(s_{j-1}, s_j)$  using the path which minimizes  $gap(s_{j-1}, s_j) = f(gap_G(s_{j-1}, s_j), gaps(s_{j-1}, s_j))$ . More precisely,  $gap_G(s_{j-1}, s_j)$  equals the element of the set

$$\{dist2begin(\mu, i) - dist2begin(s_{j-1}.v, i) + D(\mu, s_j.v) + s_j.x - s_{j-1}.y - 1 \mid i \in paths(s_{j-1}.v), \mu = last2reach(s_j.v, i)\}$$

which minimizes  $gap(s_{j-1}, s_j)$ , where  $dist2begin(v, i)$  denotes the length of sub-path of path  $P_i$  from the start of  $P_i$  to  $v$ . We will show that this definition enables significantly faster parameterized algorithms with respect to  $K$ . Again, define Problems 2a, 2b and 2c with  $f(g_1, g_2) = g_1 + g_2$ ,  $f(g_1, g_2) = max(g_1, g_2)$  and  $f(g_1, g_2) = |g_1 - g_2|$  respectively.

## 4 Proposed Algorithms

Our algorithm to address Problems 1a-1c uses a brute-force approach that evaluates all  $O(N^2)$  pairs of anchors. We use single-source shortest distances computations for measuring gaps.

**Lemma 2.** *Problems 1a, 1b and 1c can be solved optimally in  $O(N(|V| + |E| + N))$  time.*

*Proof.* We will process anchors in array  $M[1..N]$  one by one in a topological order of  $M[\cdot].v$ . If there are two anchors with equal component  $M[\cdot].v$ , then the anchor with lower component  $M[\cdot].x$  is processed first. Suppose DAG  $G'$  is obtained by reversing the edges of  $G$ . While processing anchor  $M[j]$ , we will compute partial score  $C[j]$ , i.e., the optimal score of a chain that ends at anchor  $M[j]$ . We identify all the anchors that precede  $M[j]$  using a depth-first traversal starting from  $M[j].v$  in  $G'$ . Next, we compute single-source shortest distances from  $M[j].v$  in  $G'$  which requires  $O(|V| + |E|)$  time for DAGs [8]. Finally,  $C[j]$  is computed as  $weight(M[j]) + \max(0, \max_{i: M[i] \prec M[j]} (C[i] - f(gap_G(M[i], M[j]), gaps(M[i], M[j])))$  in  $O(N)$  time.  $\square$

The above algorithm is unlikely to scale to a mammalian dataset. We leave it open whether there exists a faster algorithm to solve Problems 1a-c. Next, we propose  $O(KN \log KN)$  time algorithm for addressing Problem 2a, assuming  $O(K^3|V| + K|E|)$  time preprocessing is done for DAG  $G$ . The preprocessing stage is required to compute (a) an MPC  $\{P_1, \dots, P_K\}$  of  $G$ , (b)  $last2reach(v, i)$ , (c)  $D(last2reach(v, i), v)$  and (d)  $dist2begin(v, i)$ , for all  $v \in V$ ,  $i \in [1, K]$ .

**Lemma 3.** *The preprocessing of DAG  $G(V, E, \sigma)$  can be done in  $O(K^3|V| + K|E|)$  time.*

*Proof.* An MPC  $\{P_1, \dots, P_K\}$  can be computed in  $O(K^3|V| + |E|)$  time [5]. To compute the remaining information, we will use dynamic programming algorithms that process vertices  $\in V$  in a fixed topological order. Suppose function  $rank : V \rightarrow [1, |V|]$  assigns rank to each vertex based on its topological ordering. Let  $\mathcal{N}(v)$  denote the set of adjacent in-neighbors of  $v$ . Similar to [28],  $last2reach(v, i)$  is computed in  $O(K|V| + K|E|)$  time for all  $v \in V$ ,  $i \in [1, K]$ . Initialize  $last2reach(v, i) = 0$  for all  $v$  and  $i$ . Then, use the following recursion:

$$last2reach(v, i) = \begin{cases} rank(v) & \text{if } i \in paths(v) \\ \max_{u:u \in \mathcal{N}(v)} last2reach(u, i) & \text{otherwise} \end{cases}$$

At the end of the algorithm,  $last2reach(v, i) = 0$  will hold for only those pairs  $(v, i)$  for which  $last2reach(v, i)$  does not exist. Next, we compute  $D(last2reach(v, i), v)$ , for all  $v \in V$ ,  $i \in [1, K]$ , also in  $O(K|V| + K|E|)$  time. Initialize  $D(last2reach(v, i), v) = \infty$  for all  $v$  and  $i$ . Then, update  $D(last2reach(v, i), v)$

$$= \begin{cases} 0 & \text{if } last2reach(v, i) = v \\ \min_{u:u \in \mathcal{N}(v), last2reach(u, i) = last2reach(v, i)} D(last2reach(u, i), u) + |\sigma(u)| & \text{otherwise} \end{cases}$$

Finally,  $dist2begin(v, i)$ , for all  $v \in V$ ,  $i \in [1, K]$  is computed by linearly scanning  $K$  paths in  $O(K|V|)$  time.  $\square$

**Lemma 4.** *Assuming DAG  $G(V, E, \sigma)$  is preprocessed, Problem 2a can be solved in  $O(KN \log KN)$  time and  $O(KN)$  space.*

*Proof.* The choice of gap cost definition in Problem 2a allows us to make efficient use of range-search queries. Algorithm 2 gives an outline of the proposed dynamic programming algorithm. Similar to the previously discussed algorithms (Sect. 2.1), it also saves partial scores in array  $C[1..N]$ . We use  $K$  search trees, one per path. Search tree  $\mathcal{T}_i$  maintains partial scores  $C[\ ]$  of those anchors  $M[j]$  whose coordinates on DAG are covered by path  $P_i$ . Each search tree is initialized with keys  $M[j].d$ , and values  $-\infty$ . Subsequently,  $K$  search trees are queried and updated in a proper order.

- If  $K = 1$ , i.e., when DAG  $G$  is a linear chain, the condition in Line 6 is always satisfied and the term  $D(v, M[j].v)$  (Line 18) is always zero. In this case, Algorithm 2 works precisely as the co-linear chaining algorithm on two sequences (Algorithm 1).
- For  $K > 1$ , we use  $last2reach$  information associated with vertex  $M[j].v$  (Lines 9-11). This ensures that partial score  $C[j]$  is updated from scores of the preceding anchors on path  $P_i$  for all  $i \in [1, K] \setminus paths(M[j].v)$ .

All the query and update operations done in the search trees together use  $O(KN \log N)$  time because the count of these operations is bounded by  $O(KN)$ , and the size of each tree is  $\leq N$ . The sorting step in Line 15 requires  $O(KN \log KN)$  time to sort  $O(KN)$  tuples. The overall space required by  $K$  search trees and array  $Z$  is  $O(KN)$ .  $\square$

**Algorithm 2:**  $O(KN \log KN)$  time chaining algorithm for Problem 2a

---

**Input:** Array of weighted anchors  $M[1..N]$ , preprocessed DAG  $G(V, E, \sigma)$   
**Output:** Array  $C[1..N]$  such that  $C[j]$  = score of an optimal chain that ends at  $M[j]$

```

1 Initialize search tree  $T_i$ , for all  $i \in [1, K]$  using keys  $\{M[j].d \mid 1 \leq j \leq N\}$  and values  $-\infty$ 
2 Initialize  $C[j]$  as  $weight(M[j])$ , for all  $j \in [1, N]$ 
3 /* Create array  $Z$  that stores tuples of the form  $(v, pos, task, anchor, path)$ , where  $v \in V$ ,
    $pos \in \mathbb{N}$ ,  $task \in \{0, 1\}$ ,  $anchor \in [1, N]$  and  $path \in [1, K]$ .*/
4 for  $j \leftarrow 1$  to  $N$  do
5   for  $i \leftarrow 1$  to  $K$  do
6     if  $i \in paths(M[j].v)$  then
7        $Z.push(M[j].v, M[j].x, 0, j, i)$ 
8        $Z.push(M[j].v, M[j].y, 1, j, i)$ 
9     else if  $last2reach(M[j].v, i)$  exists then
10       $v \leftarrow last2reach(M[j].v, i)$ 
11       $Z.push(v, |\sigma(v)| + 1, 0, j, i)$ 
12
13   end
14 end
15 for  $z \in Z$  in lexicographically ascending order based on the key  $(rank(v), pos, task)$  do
16    $j \leftarrow z.anchor, i \leftarrow z.path, v \leftarrow z.v, wt \leftarrow weight(M[j])$ 
17   if  $z.task = 0$  then
18      $C[j] \leftarrow \max(C[j], wt + T_i.RMQ(0, M[j].c) - M[j].x - dist2begin(v, i) -$ 
19        $D(v, M[j].v) - M[j].c + 2)$ 
20   else
21      $T_i.update(M[j].d, C[j] + M[j].y + dist2begin(v, i) + M[j].d)$ 
22 end

```

---

For simplicity of notations, we have not allowed an anchor to span two or more connected vertices in a DAG, but the proposed framework can be easily generalized to handle this [26, 28]. Finally, we design algorithms for Problems 2b and 2c by using 2-dimensional RMQs. We summarize the result below. The proof is provided in the expanded version of this paper [6].

**Lemma 5.** *Assuming DAG  $G(V, E, \sigma)$  is preprocessed, Problems 2b and 2c can be solved in  $O(KN \log^2 N + KN \log KN)$  time and  $O(KN \log N)$  space.*

## 5 Implementation Details

Among the proposed algorithms, Algorithm 2 has the best time complexity. We implemented this algorithm in C++, and developed a practical long read alignment software Minichain.

**Pangenome Graph Representation:** A path in pangenome reference graph  $G(V, E, \sigma)$  spells a sequence in a single orientation, whereas a read may be sampled from either the same or the opposite orientation due to the double-stranded nature of DNA. To address this internally in Minichain, for each vertex  $v \in V$ , we also add another vertex  $\bar{v}$  whose string label is the reverse complement of string  $\sigma(v)$ . For each edge  $u \rightarrow v \in E$ , we also add the complementary edge  $\bar{v} \rightarrow \bar{u}$ . This process doubles the count of edges and vertices.

**Optimization for Whole-Genome Pangenome Graphs:** A pan-genome reference graph associated with a complete human genome is a union of weakly

connected components, one per chromosome, because there is no edge which connects two chromosome components. We actually maintain two components per chromosome, one being the reverse complement of the other. During both preprocessing and chaining stages of the proposed algorithms, each component is treated independently. The parameter  $K$  in our time complexity results is determined by the maximum  $K$  value among the components. We use GraphChainer implementation [26] to compute minimum path cover and range queries. We also optimize runtime by performing parallel preprocessing of different components (Lemma 3) using multiple threads.

**Computing Multiple Best Chains and Confidence Scores:** When a read is sampled from a repetitive region of a genome, computing read’s true path of origin becomes challenging. Practical methods often report more than one alignment per read in such cases. The highest-scoring alignment is marked as *primary* alignment, and the remaining are marked as *secondary*. Additionally, based on the score difference between the primary and the highest-scoring secondary alignment, a confidence score  $\in [0, 60]$  is provided as *mapping quality* that represents the likelihood that the primary alignment is correct [24]. In Minichain, we also implement an algorithm to identify multiple high-scoring chains so that multiple base-to-base alignment records can be reported to a user. Algorithm 2 returns partial scores  $C[1..N]$  in the end. We perform backtracking from anchor  $\text{argmax}_j C[j]$  to compute the optimal chain. The anchors involved in this chain are marked as *visited*. Iteratively, we check presence of another chain (a) whose score is  $\geq \tau \cdot \max_j C[j]$ , where  $\tau \in [0, 1]$  is a user-specified threshold with default value 0.95, and (b) none of the anchors in the chain are previously *visited*. We stop when no additional chains exist that satisfy these conditions.

**Computing Anchors and Final Base-to-Base Alignments:** In Minichain, we use the seeding and base-to-base alignment methods from Minigraph [23]. The seeding method in Minigraph works by identifying common minimizers between query sequence and string labels  $\sigma(v)$  of graph vertices. Given a pre-defined ordering of all  $k$ -mers and  $w$  consecutive  $k$ -mers in a sequence,  $(w, k)$ -minimizer is the smallest  $k$ -mer among the  $w$   $k$ -mers [37]. The common minimizer occurrences between a query and vertex labels form anchors. In our experiments, we use same parameters  $k = 17, w = 11$  as Minigraph. The weight of each anchor is  $k$  times a user-specified constant which is set to 200 by default. Algorithm 2 is used to compute the best chains and discard those anchors which do not contribute to these chains. Finally, we return the filtered anchors to Minigraph’s alignment module to compute base-to-base alignments [43].

## 6 Experiments

**Benchmark Datasets:** We built string-labeled DAGs of varying sizes by using Minigraph v0.19 [23]. Each DAG is built by using a subset of 95 publicly available haplotype-resolved human genome assemblies [25, 31]. In Minigraph, a DAG

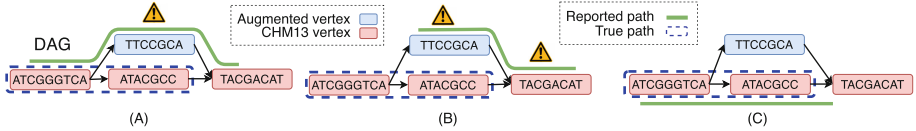


is iteratively constructed by aligning each haplotype assembly to an intermediate graph, and augmenting additional vertices and/or edges for each structural variant observed. We disabled inversion variants by using `--inv=no` parameter to avoid introducing cycles in the DAG. CHM13 human genome assembly [31] is used as the starting reference, and we added other haplotype assemblies during DAG construction. In the CHM13 assembly, the first 24 contigs represent individual chromosome (1–22, X, Y) sequences, and the last 25<sup>th</sup> contig represents mitochondrial DNA. Using this data, we constructed five DAGs, labeled as 1H, 10H, 40H, 80H and 95H respectively. In each of these DAG labels, the integer prefix reflects the count of haplotype assemblies present in the DAG. Properties of these DAGs are shown in Table 1. Parameter  $K$ , i.e., the size of MPC, is presented as a range because different connected components in a DAG have different MPCs. For all DAGs, note that the maximum  $K$  is  $\ll |V|$ . We used PBSIM2 v2.0.1 [32] to simulate long reads from CHM13 human assembly. For each simulated read and each DAG, we know the true string path where the read should align. PBSIM2 input parameters were set such that we get sequencing error rate and N50 read length as 5% and 10 kbp respectively. The commands used to run the tools are available in the expanded version of this paper [6].

**Table 1.** Properties of DAGs used in our experiments. Total sequence length indicates the sum of length of string labels of all vertices in the DAG.

DAG	$ V $	$ E $	No. of structural variants	N50 length of vertex labels (kbp)	Total sequence length (Gbp)	$\mathbf{K}$ (min-max)
1H	25	0	0	150,617	3.11	1–1
10H	141,755	203,160	61,430	225	3.15	1–9
40H	340,451	489,612	149,186	126	3.23	1–20
80H	553,271	797,528	244,282	85	3.31	1–29
95H	611,949	882,739	270,815	78	3.34	1–35

**Evaluation Methodology:** Alignment output of a read specifies the string path in the input DAG against which the read is aligned. An appropriate evaluation criteria is needed to classify the reported string path as either correct or incorrect by comparing it to the true path. We followed a similar criteria that was used in previous studies [22, 23]. First, the reported string path should include only those vertices which correspond to CHM13 assembly, i.e., it should not span an edge augmented from other haplotypes (Fig. 3). Second, the reported interval in CHM13 assembly should overlap with the true interval, and the overlapping length should exceed  $\geq 10\%$  length of the union of the true and the reported intervals. A correct alignment should satisfy both the conditions. We use `paftools` [22] which implements this evaluation method. All our experiments were done on AMD EPYC 7742 64-core processor with 1 TB RAM. We used 32 threads to run each aligner because all the tested tools support multi-threading by considering each read independently. Wall clock time and peak memory usage were measured using `/usr/bin/time` Linux command.



**Fig. 3.** Illustration of the evaluation criteria. Among the three reported paths above, only (C) is correct.

**Performance Comparison with Existing Algorithms:** We compared Minichain (v1.0) to three existing sequence-to-DAG aligners: Minigraph v0.19 [23], GraphAligner v1.0.16 [35], and GraphChainer v1.0 [26]. Minigraph uses a two-stage co-linear chaining approach. The first stage ignores edges in the graph and solves co-linear chaining between query sequence and vertex labels. The second stage combines the vertex-specific-chains. In contrast, GraphAligner does not use co-linear chaining and instead relies on its own clustering heuristics. GraphChainer solves co-linear chaining on DAG without penalizing gaps. All the aligners, except GraphChainer, also compute mapping quality (confidence score) for each alignment. We excluded optimal sequence-to-DAG aligners (e.g., [16, 18]) because they do not scale to DAGs built by using entire human genomes.

We evaluated accuracy and runtime of Minichain using three DAGs 1H, 10H and 95H (Tables 2, 3 and 4). While using DAG 1H, we also tested Minimapp2 v2.24 [22], a well-optimized sequence-to-sequence aligner, by aligning reads directly to CHM13 genome assembly. The results show that Minichain consistently achieves the highest precision among the existing sequence-to-DAG aligners. It aligns a higher fraction of reads compared to Minigraph. The gains are also visible when mapping quality (MQ) cutoff 10 is used to filter out low-confidence alignments. GraphAligner and GraphChainer align 100% reads consistently, but this is supplemented with much higher fraction of incorrectly aligned reads. Both Minigraph and Minichain do not align 100% reads. This likely happens because the seeding method used in these two aligners filters out the most frequently occurring minimizers from DAG to avoid processing too many anchors. This can leave several reads originating from long-repetitive genomic regions as unaligned [20].

Among the four aligners, Minigraph performs the best in terms of runtime. Runtime of Minichain increases for DAG 95H because of higher value of  $K$ . However, we expect that this can be partly addressed with additional improvements in the proposed chaining algorithm, e.g., by dynamically deleting the anchors from search trees whose gap from all the remaining unprocessed anchors exceeds an acceptable limit. Overall, the results demonstrate practical advantage of Minichain for accurate long-read alignment to DAGs. Superior accuracy of Minichain is also illustrated using precision-recall plots in Fig. 4.

**Impact of Increasing DAG Size on Accuracy:** Alignment accuracy generally deteriorates as count of haplotypes increases in DAGs for all the tested aligners. For each read that was not aligned correctly, we checked if the corresponding reported string path overlaps with the true interval (Fig. 3, case A). Such reads are aligned to *correct region* in the DAG but the reported path uses

**Table 2.** Performance comparison of long read aligners using DAG 1H.

	Minichain	Minigraph	GraphAligner	GraphChainer	Minimap2
Indexing time (sec)	<b>65</b>	<b>65</b>	395	458	55
Alignment time (sec)	205	<b>104</b>	6298	6714	133
Memory usage (GB)	20.06	<b>19.66</b>	38.12	126.14	12.48
Unaligned reads	0.94%	2.11%	<b>0%</b>	<b>0%</b>	0%
Incorrect aligned reads	<b>0.58%</b>	0.66%	1.06%	1.33%	0.56%
Unaligned reads (MQ $\geq$ 10)	3.89%	5.82%	0.80%	<b>0%</b>	2.29%
Incorrect aligned reads (MQ $\geq$ 10)	<b>0.02%</b>	0.11%	0.53%	1.33%	0.0013%

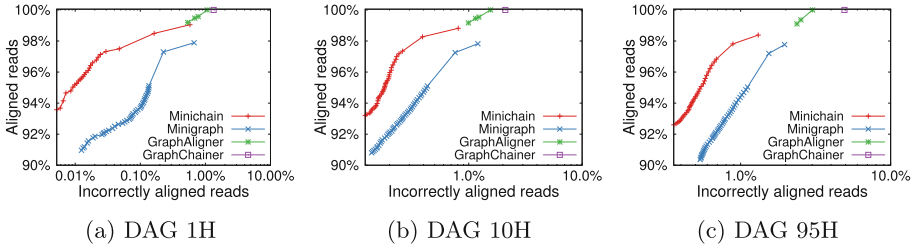
**Table 3.** Performance comparison of long read aligners using DAG 10H.

	Minichain	Minigraph	GraphAligner	GraphChainer
Indexing time (sec)	67	<b>66</b>	321	537
Alignment time (sec)	610	<b>132</b>	5479	9642
Memory usage (GB)	<b>23.15</b>	23.16	38.41	143.94
Unaligned reads	1.17%	2.17%	<b>0%</b>	<b>0%</b>
Incorrect aligned reads	<b>0.80%</b>	1.20%	1.55%	2.10%
Unaligned reads (MQ $\geq$ 10)	4.03%	5.88%	0.28%	<b>0%</b>
Incorrect aligned reads (MQ $\geq$ 10)	<b>0.20%</b>	0.34%	0.99%	2.10%

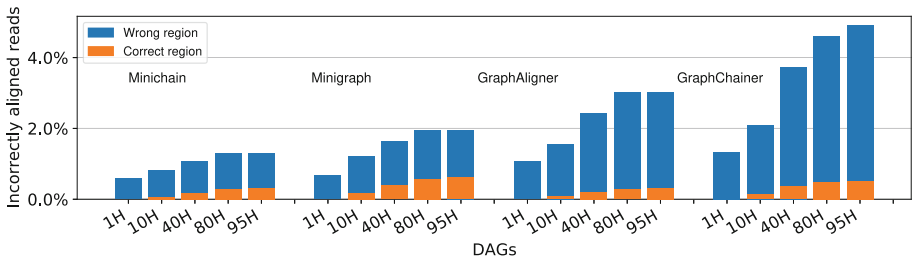
one or more augmented edges. The remaining fraction of incorrectly aligned reads align to *wrong region* in the DAG. We observe that the fraction of incorrectly-aligned reads which align to correct region in DAG increases with increasing count of haplotypes (Fig. 5). This happens because the count of alternate paths increases combinatorially with more number of haplotypes which makes precise alignment of a read to its path of origin a challenging problem. Addressing this issue requires further algorithmic improvements.

**Table 4.** Performance comparison of long read aligners using DAG 95H.

	Minichain	Minigraph	GraphAligner	GraphChainer
Indexing time (sec)	77	<b>71</b>	342	763
Alignment time (sec)	1414	<b>154</b>	5695	17336
Memory usage (GB)	<b>24.75</b>	24.76	40.79	192.36
Unaligned reads	1.62%	2.23%	<b>0%</b>	<b>0%</b>
Incorrect aligned reads	<b>1.31%</b>	1.96%	3.01%	4.92%
Unaligned reads (MQ $\geq$ 10)	4.75%	6.26%	0.88%	<b>0%</b>
Incorrect aligned reads (MQ $\geq$ 10)	<b>0.56%</b>	0.89%	2.38%	4.92%



**Fig. 4.** Precision-recall curves obtained by using different aligners. X-axis indicates percentage of incorrectly aligned reads in log-scale. These curves are obtained by setting different mapping quality cutoffs  $\in [0, 60]$ . GraphChainer curve is a single point because it reports fixed mapping quality 60 in all alignments.



**Fig. 5.** The fraction of incorrectly aligned reads is shown using DAGs 1H, 10H, 40H, 80H and 95H. Each incorrectly-aligned read is further classified as aligned to either a wrong or a correct region in the DAG based on whether the reported string path overlaps with the true string path (e.g., cases A, B in Fig. 3).

**Acknowledgements.** This work was supported by funding from the National Supercomputing Mission, India under DST/NSM/R&D\_HPC\_Applications. We used computing resources provided by the C-DAC National PARAM Supercomputing Facility, India, and the National Energy Research Scientific Computing Center, USA.

## References

1. Abouelhoda, M., Ohlebusch, E.: Chaining algorithms for multiple genome comparison. *J. Discrete Algorithms* **3**(2–4), 321–341 (2005)
2. Baaijens, J.A., et al.: Computational graph pangenomics: a tutorial on data structures and their applications. *Nat. Comput.* **21**, 81–108 (2022). <https://doi.org/10.1007/s11047-022-09882-6>
3. Backurs, A., Indyk, P.: Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In: *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 51–58 (2015)
4. de Berg, M., Cheong, O., van Kreveld, M.J., Overmars, M.H.: *Computational Geometry: Algorithms and Applications*, 3rd edn. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-3-540-77974-2>

5. Cáceres, M., Cairo, M., Mumey, B., Rizzi, R., Tomescu, A.I.: Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 359–376. SIAM (2022)
6. Chandra, G., Jain, C.: Sequence to graph alignment using gap-sensitive co-linear chaining. *BioRxiv* (2022). <https://doi.org/10.1101/2022.08.29.505691>
7. Computational Pan-Genomics Consortium: Computational pan-genomics: status, promises and challenges. *Brief. Bioinform.* **19**(1), 118–135 (2018)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2022)
9. Dvorkina, T., Antipov, D., Korobeynikov, A., Nurk, S.: SPAligner: alignment of long diverged molecular sequences to assembly graphs. *BMC Bioinform.* **21**(12), 1–14 (2020)
10. Eggertsson, H.P., Jonsson, H., Kristmundsdottir, S., et al.: Graphtyper enables population-scale genotyping using pangenome graphs. *Nat. Genet.* **49**(11), 1654–1660 (2017)
11. Eizenga, J.M., et al.: Pangenome graphs. *Annu. Rev. Genomics Hum. Genet.* **21**, 139 (2020)
12. Eppstein, D., Galil, Z., Giancarlo, R., Italiano, G.F.: Sparse dynamic programming I: linear cost functions. *J. ACM* **39**(3), 519–545 (1992)
13. Eppstein, D., Galil, Z., Giancarlo, R., Italiano, G.F.: Sparse dynamic programming II: convex and concave cost functions. *J. ACM* **39**(3), 546–567 (1992)
14. Garg, S., Rautiainen, M., Novak, A.M., et al.: A graph-based approach to diploid genome assembly. *Bioinformatics* **34**(13), i105–i114 (2018)
15. Illumina: DRAGEN v3.10.4 software release notes. [https://support.illumina.com/content/dam/illumina-support/documents/downloads/software/dragen/20001606\\_5\\_00\\_DRAGEN-3.10-Customer-Release-Notes.pdf](https://support.illumina.com/content/dam/illumina-support/documents/downloads/software/dragen/20001606_5_00_DRAGEN-3.10-Customer-Release-Notes.pdf). Accessed 08 Aug 2022
16. Ivanov, P., Bichsel, B., Vechev, M.: Fast and optimal sequence-to-graph alignment guided by seeds. In: Pe’er, I. (ed.) RECOMB 2022. LNBI, vol. 13278, pp. 306–325. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-04749-7\\_22](https://doi.org/10.1007/978-3-031-04749-7_22)
17. Jain, C., Gibney, D., Thankachan, S.V.: Co-linear chaining with overlaps and gap costs. In: Pe’er, I. (ed.) RECOMB 2022. LNBI, vol. 13278, pp. 246–262. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-04749-7\\_15](https://doi.org/10.1007/978-3-031-04749-7_15)
18. Jain, C., Misra, S., Zhang, H., Dilthey, A., Aluru, S.: Accelerating sequence alignment to graphs. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 451–461. IEEE (2019)
19. Jain, C., Rhie, A., Hansen, N.F., Koren, S., Phillippy, A.M.: Long-read mapping to repetitive reference sequences using Winnowmap2. *Nat. Methods* **19**(6), 705–710 (2022)
20. Jain, C., et al.: Weighted minimizer sampling improves long read mapping. *Bioinformatics* **36**(Supplement\_1), i111–i118 (2020)
21. Jain, C., Zhang, H., Dilthey, A., Aluru, S.: Validating paired-end read alignments in sequence graphs. In: 19th International Workshop on Algorithms in Bioinformatics (WABI 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
22. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**(18), 3094–3100 (2018). <https://doi.org/10.1093/bioinformatics/bty191>
23. Li, H., Feng, X., Chu, C.: The design and construction of reference pangenome graphs with minigraph. *Genome Biol.* **21**(1), 265 (2020). <https://doi.org/10.1186/s13059-020-02168-z>
24. Li, H., Ruan, J., Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.* **18**(11), 1851–1858 (2008)

25. Liao, W.W., et al.: A draft human pangenome reference. *BioRxiv* (2022). <https://doi.org/10.1101/2022.07.09.499321>
26. Ma, J., Cáceres, M., Salmela, L., Mäkinen, V., Tomescu, A.I.: GraphChainer: co-linear chaining for accurate alignment of long reads to variation graphs. *BioRxiv* (2022)
27. Mäkinen, V., Sahlin, K.: Chaining with overlaps revisited. In: 31st Annual Symposium on Combinatorial Pattern Matching (CPM 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
28. Mäkinen, V., Tomescu, A.I., Kuosmanen, A., Paavilainen, T., Gagie, T., Chikhi, R.: Sparse dynamic programming on DAGs with small width. *ACM Trans. Algorithms* **15**(2), 1–21 (2019)
29. Myers, G., Miller, W.: Chaining multiple-alignment fragments in sub-quadratic time. In: *SODA*, vol. 95, pp. 38–47 (1995)
30. Navarro, G.: Improved approximate pattern matching on hypertext. *Theor. Comput. Sci.* **237**(1–2), 455–463 (2000)
31. Nurk, S., Koren, S., Rhie, A., Rautiainen, M., et al.: The complete sequence of a human genome. *Science* **376**(6588), 44–53 (2022). <https://doi.org/10.1126/science.abj6987>
32. Ono, Y., Asai, K., Hamada, M.: PBSIM2: a simulator for long-read sequencers with a novel generative model of quality scores. *Bioinformatics* **37**(5), 589–595 (2020). <https://doi.org/10.1093/bioinformatics/btaa835>
33. Otto, C., Hoffmann, S., Gorodkin, J., Stadler, P.F.: Fast local fragment chaining using sum-of-pair gap costs. *Algorithms Mol. Biol.* **6**(1), 4 (2011). <https://doi.org/10.1186/1748-7188-6-4>
34. Paten, B., Novak, A.M., Eizenga, J.M., Garrison, E.: Genome graphs and the evolution of genome inference. *Genome Res.* **27**(5), 665–676 (2017)
35. Rautiainen, M., Marschall, T.: GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biol.* **21**(1), 1–28 (2020). <https://doi.org/10.1186/s13059-020-02157-2>
36. Ren, J., Chaisson, M.J.: *lra*: a long read aligner for sequences and contigs. *PLoS Comput. Biol.* **17**(6), e1009078 (2021)
37. 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). <https://doi.org/10.1093/bioinformatics/bth408>
38. Sahlin, K., Baudeau, T., Cazaux, B., Marchet, C.: A survey of mapping algorithms in the long-reads era. *BioRxiv* (2022)
39. Sahlin, K., Mäkinen, V.: Accurate spliced alignment of long RNA sequencing reads. *Bioinformatics* **37**(24), 4643–4651 (2021)
40. Salmela, L., Rivals, E.: LoRDEC: accurate and efficient long read error correction. *Bioinformatics* **30**(24), 3506–3514 (2014)
41. Sirén, J., Monlong, J., Chang, X., et al.: Pangenomics enables genotyping of known structural variants in 5202 diverse genomes. *Science* **374**(6574), abg8871 (2021)
42. Wang, T., Antonacci-Fulton, L., Howe, K., et al.: The human pangenome project: a global resource to map genomic diversity. *Nature* **604**(7906), 437–446 (2022)
43. Zhang, H., Wu, S., Aluru, S., Li, H.: Fast sequence to graph alignment using the graph wavefront algorithm. *arXiv preprint arXiv:2206.13574* (2022)



# DM-Net: A Dual-Model Network for Automated Biomedical Image Diagnosis

Xiaogen Zhou, Zhiqiang Li, and Tong Tong<sup>(✉)</sup>

College of Physics and Information Engineering, Fuzhou University,  
No. 2, Wulong Jiangbei Avenue, Fuzhou 350108, Fujian, China  
ttraveltong@gmail.com

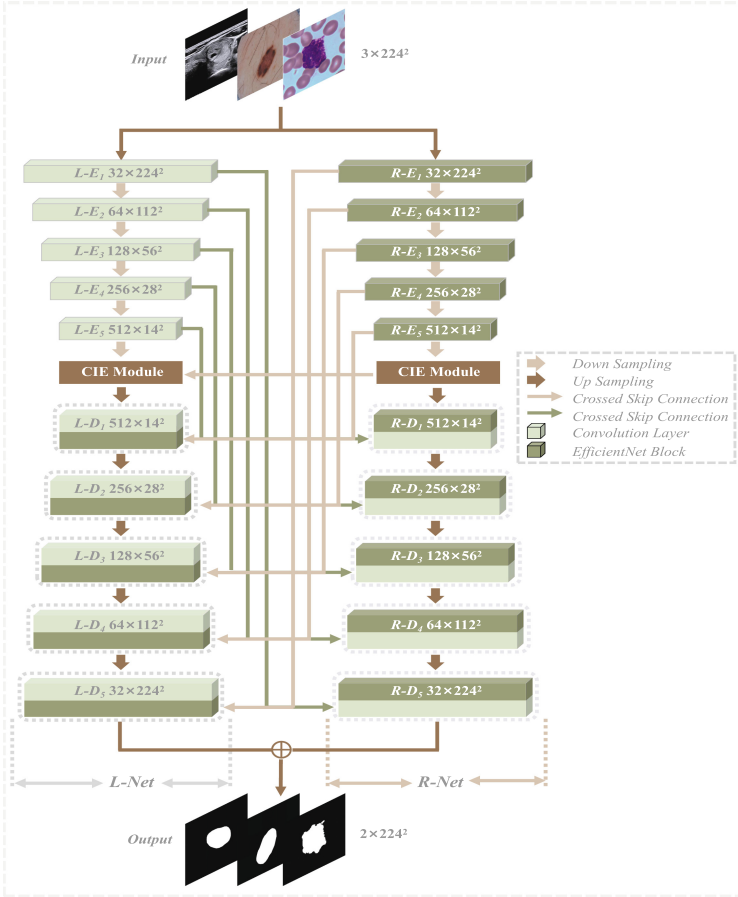
**Abstract.** Biomedical image segmentation is an essential task in the computer-aided diagnosis system. An encoder-decoder based on a shallow or deep convolutional neural network (DCNN) is an extensively used framework for biomedical image analysis. To study and rethink the effectiveness of compounding both the shallow and deep networks for the medical image segmentation task, we propose a dual-model CNN architecture, called DM-Net, for biomedical image segmentation. DM-Net is composed of a shallow CNN structure at its left, called L-Net and a deeper CNN structure at its right, named R-Net. The L-Net is proposed to encode low-level contextual information and the R-Net is presented to produce high-level semantic feature maps. Furthermore, a novel crossed-skip connection (CSC) strategy is proposed to transfer information between the two side networks mutually. Extensive experiments demonstrate that our method outperforms representative approaches on three public medical image datasets.

**Keywords:** Medical image segmentation · Crossed-skip connection · Contextual information encoding

## 1 Introduction

Accurate segmentation of skin lesion, white blood cell (WBC), and thyroid nodule from biomedical images, such as dermoscopic images, human blood smear images, and thyroid ultrasound images, are of great significance in providing clear guidance for clinical applications such as disease diagnosis and further treatment. In clinical practice, biomedical images are analyzed by doctors via a hybrid visual inspection system for diagnosing human diseases. It needs a high degree of ability and concentration and is labour-intensive. An automated computer-aided diagnosis system is able to release physicians from time-consuming work, and assist doctors in improving human disease diagnosis in terms of accuracy, efficiency, and objectivity.

The emergence of deep convolutional neural networks (DCNNs) has shown a strong ability in biomedical image segmentation [1–3, 7, 18–22]. However,



**Fig. 1.** Illustration of the proposed DM-Net architecture. DM-Net is composed of a shallow CNN model at its left, called L-Net and a deeper CNN at its right, named R-Net. The L-Net is proposed to encode low-level contextual information and the R-Net is presented to produce high-level semantic feature maps. Furthermore, a novel CSC strategy is proposed to bridge information between the L-Net and the R-Net mutually.

most existing DCNNs are based on U-Net architecture [8–11], which is a shallow CNN structure. Moreover, the shallower CNNs tend to extract more coarse/fine-grained features and it is difficult to produce the vanishing gradient issue. In addition, the shallower CNNs are beneficial to train. The classical U-Net [3] and its variations [4–6, 23] can generate coarse-grained and fine-grained low-level feature maps with the uniformly scaled network width using the standard skip connection scheme. Though these CNN models have obtained notable performance in medical image processing, they are still limited to difficulty in capturing more high-dimensional and complex information. To overcome this problem, scaling



up the depth of CNN models is widely exploited from various DCNN models [7, 12, 13]. Intuitively, the deeper CNN models tend to extract richer and more high-dimensional feature maps than those shallower CNN models. For instance, Gu et al. [7] used pre-trained ResNet’s encoder blocks as a fixed feature encoder to encode contextual information for achieving the medical image segmentation tasks. Li et al. [12] proposed a deeper and more efficient CNN framework to extract hybrid feature maps for biomedical image segmentation. However, these deep networks have a common shortcoming in producing fine-grained feature maps by utilizing the deeper depth of convolutional layers and being hard to train because of the vanishing gradient issue.

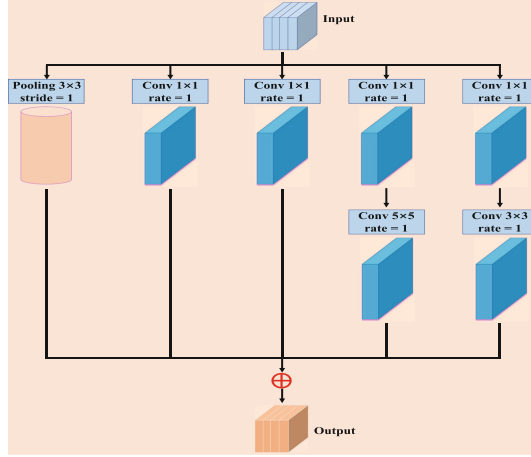
In this paper, to alleviate those above issues, we want to rethink the process of compounding the shallow and deep networks. In particular, a question arises: is there a simple yet effective compound method that can achieve better accuracy and efficiency for general medical image analysis? Intuitively, we propose a new compound structure for automated biomedical image segmentation, called DM-Net. It compounds a shallow network and a deep network. In DM-Net, the proposed CSC strategy is used to bridge both the two side networks. In this way, the dual-model of the proposed DM-Net is allowed to interact by the CSC strategy. Firstly, the L-Net transfers low-dimensional information to the decoder blocks of the R-Net and further help it preserve more low-dimensional information. On the other hand, the R-Net also transfers high-dimensional semantic information to the decoder blocks of the L-Net and further facilitate it facilitate preserve more high-level semantic information. Extensive experiments demonstrate the effectiveness of the proposed DM-Net across different biomedical image datasets.

## 2 Methodology

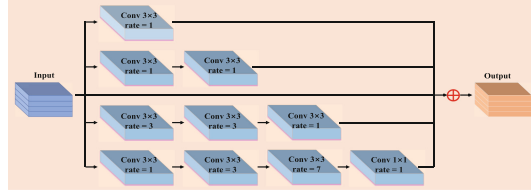
Figure 1 shows our proposed DM-Net’s architecture, which composes a shallow network structure and a deep network structure. The details of the proposed DM-Net are illustrated as follows.

### 2.1 The Shallow CNN Model: L-Net

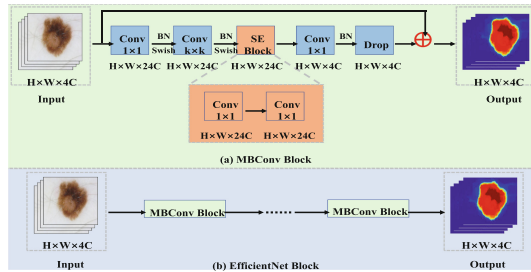
Our proposed L-Net is a shallow CNN model to capture low-level features, which includes an encoder-decoder structure, which utilizes U-Net [3] as its backbone. Furthermore, motivated by the dense atrous convolution [7], we present a contextual information encoding (CIE) module to encode the high-level semantic information.



**Fig. 2.** The illustration of the cascaded dilated inception module.



**Fig. 3.** An illustration of our proposed stacked atrous convolutional (SAC) module for extracting high-level features.



**Fig. 4.** The illustration of the MBConv block and the EfficientNet block.

(1) **Encoder-Decoder Structure:** Our L-Net is formed using the backbone of U-Net [3] structure. There are the encoder pathway and decoder pathway in the L-Net. The encoder flow performs each convolutional layer with a filter to produce a set of feature maps, and a Relu activation function is followed. The decoder pathway also applies each convolutional layer to produce feature maps. In addition, our proposed crossed skip connection strategy transfers the corresponding feature maps from the encoder pathway and concatenates them to upsample feature maps of the R-Net's decoder blocks.

**(2) CIE Module:** We propose a novel contextual information encoding (CIE) module to encode more high-level semantic features. Figure 1 shows an architecture of our proposed CIE module, which is integrated using a novel CDI module and a SAC module, which can be illustrated as follows.

**CDI Module:** We present a cascaded densely inception (CDI) module to extract the high-level semantic information maps. It is spirited by the Inception module [15]. As shown in Fig. 2, the proposed CDI module includes  $3 \times 3$  max pooling layer,  $1 \times 1$  convolutional layer (CL),  $1 \times 1$  and  $5 \times 5$  CLs with an atrous rate of 1, and  $1 \times 1$  and  $3 \times 3$  CLs with an atrous rate of 1.

**SAC Module:** In addition, we propose a novel stacked atrous convolutional (SAC) module to encode high-dimensional contextual information. It is motivated by atrous convolution [16]. As shown in Fig. 3, our proposed SAC module comprises several cascaded pipelines with the gradual increment of the number of the atrous convolutional layer (CL) from 1 to 1, 3, and 7. Each branch’s receptive field (RF) is 3, 7, 9 and 23. The proposed SAC module uses different receptive fields. In each atrous convolution pipeline, we use 11 CLs before rectifying the linear activation function. Furthermore, we add the original feature maps with other feature maps. Generally, the CL with an extensive RF can encode and generate high-dimensional information for large target objects, but the CL with a small RF is beneficial for encoding small targets’ feature maps. Therefore, we cascade the atrous CL integrated with different atrous rates to capture different scale targets’ features.

## 2.2 The Deep CNN Model: R-Net

Moreover, a deep network model is proposed at the right of the proposed DM-Net to encode high-level semantic information, named R-Net. It has a deeper depth of convolutional layers than the L-Net, which can transfer high-dimensional information (i.e., spatial and contextual information) to the L-Net. Specifically, we apply the pre-trained EfficientNet’s encoder block [14] as its encoder. The proposed DM-Net is a simple yet highly effective compound coefficient by uniformly scaling all depth/width/resolution dimensions. A flowchart view of the EfficientNet block is shown in Fig. 4(b). Each EfficientNet block consists of several MBConv blocks (see Fig. 4(a)), which are formed by several  $1 \times 1$  convolutional layers, batch normalization, swish activate layers, and squeeze and excitation (SE) block. Thus, our R-Net can achieve remarkable performance with the same parameters by uniformly scaling the network width, depth, or resolution in a fixed proportion. To use the EfficientNet encoder in the proposed DM-Net architecture, we add the CDI and SAC module as the last convolutional layer of the encoder block. Meanwhile, the decoder flow also applies each convolutional layer to produce feature maps. The proposed CSC strategy transfers its corresponding feature maps from the encoder pathway to the decoder pathway of the L-Net. Using three biomedical image datasets, we co-train the two side networks

(i.e., the L-Net and the R-Net). Finally, we optimize the R-Net by minimizing a dice loss.

### 2.3 Loss Function

We used a hybrid loss to optimize our proposed DM-Net during the training phase, which jointly combined the Dice loss and a cross-entropy loss to address the training problems of class imbalance. Dice loss  $\mathcal{L}_{dice}$ , cross-entropy loss  $\mathcal{L}_{ce}$ , and hybrid loss  $\mathcal{L}_{hybrid}$  are defined as follows:

$$\mathcal{L}_{dice} = 1 - \sum_k^K \frac{2\omega_k \sum_i^N p(k, i)g(k, i)}{\sum_i^N p^2(k, i) + \sum_i^N g^2(k, i)} \quad (1)$$

$$\mathcal{L}_{ce} = -\frac{1}{N} \sum_{i=1}^N G(k, i).log(P(k, i)) \\ + (1 - G(k, i)).log(1 - P(k, i)) \quad (2)$$

$$\mathcal{L}_{hybrid} = \lambda\mathcal{L}_{ce} + \mu\mathcal{L}_{dice} \quad (3)$$

$$1 = \mu + \lambda \quad (4)$$

where  $N$  is the number of pixels,  $P(k, i) \in (0, 1)$  and  $G(k, i) \in (0, 1)$  denote the produced probability and the ground truth label for class  $k$ , respectively.  $K$  is the class number, and  $\sum_k \omega_k = 1$  are the class weights. In our study, we set  $\omega_k = \frac{1}{K}$  empirically.  $\mathcal{L}_{ce}$  is the cross-entropy loss.  $\mathcal{L}_{dice}$  is the dice loss.  $\lambda$  and  $\mu$  are two weighted factors that balance the impact of  $\mathcal{L}_{ce}$  and  $\mathcal{L}_{dice}$ . In addition,  $\lambda$  and  $\mu$  are set at 0.4 and 0.6 empirically, respectively in our experiments.

## 3 Experiments

**Table 1.** A quantitative skin lesion segmentation performance of the proposed DM-Net and other comparison CNN models on the ISIC-2017 dataset.

Datasets	Methods	Accuracy	mIoU	Sensitivity	Specificity
ISIC-2017	U-Net [3]	0.9299	0.8237	0.8235	0.9748
	AG-Net [2]	0.927	0.8208	0.8222	0.9669
	M-Net [8]	0.9291	0.8231	0.8171	0.9737
	U-Net++ [10]	0.931	0.8281	0.8354	0.9743
	U-Net3+ [11]	0.9315	0.8271	0.8277	0.9753
	CE-Net [7]	0.9396	0.8442	0.8294	0.9821
	MB-DCNN [1]	0.9275	0.821	0.7922	0.9824
	Ours	<b>0.9398</b>	<b>0.8531</b>	<b>0.8442</b>	<b>0.9869</b>

**Table 2.** A quantitative WBC segmentation performance of the proposed DM-Net and other comparison CNN models on the SCISC dataset.

Datasets	Methods	Accuracy	mIoU	Sensitivity	Specificity
SCISC	U-Net [3]	0.9613	0.871	0.8697	0.9663
	AG-Net [2]	0.9678	0.8939	0.8657	0.989
	M-Net [8]	0.9756	0.9184	0.9276	0.9868
	U-Net++ [10]	0.968	0.8923	0.8749	0.9886
	U-Net3+ [11]	0.9709	0.8993	0.8771	0.9899
	CE-Net [7]	0.9727	0.9092	0.9396	0.98
	MB-DCNN [1]	0.9627	0.8736	0.8626	0.983
	Ours	<b>0.9905</b>	<b>0.9273</b>	<b>0.9509</b>	<b>0.9901</b>

**Table 3.** A quantitative thyroid nodule segmentation performance of the proposed DM-Net and other comparison CNN models on the TNUI-2021 dataset.

Datasets	Methods	Accuracy	mIoU	Sensitivity	Specificity
TNUI-2021	U-Net [3]	0.9875	0.8555	0.7821	0.9977
	AG-Net [2]	0.9875	0.8587	0.8349	0.9957
	M-Net [8]	<b>0.9877</b>	0.8639	0.8225	0.9974
	U-Net++ [10]	0.9876	0.8558	0.8077	0.9973
	U-Net3+ [11]	0.9872	0.854	0.7858	0.9973
	CE-Net [7]	0.9396	0.8431	0.8242	0.984
	MB-DCNN [1]	0.9817	0.7463	0.5989	0.9946
	Ours	0.977	<b>0.9028</b>	<b>0.8031</b>	<b>0.9997</b>

### 3.1 Datasets and Settings

- **ISIC-2017 Dataset.** The 2017 International Skin Imaging Collaboration (ISIC) dataset is associated with a disease type and ground truth of each skin lesion image. It contains 2k training samples, 150 validation samples, and 600 testing samples. Each skin lesion image is annotated by an expert as ground truth (GT) for the segmentation task.
- **SCISC Dataset:** The second medical image dataset [17], called SCISC. It includes 268 single WBC of human blood smear images (51 samples of neutrophils, 54 samples of eosinophils, 56 samples of basophils, 54 samples of monocytes, and 53 samples of lymphocytes). There are 185 training, 53 validation, and 30 testing WBC blood smear images. Each image is paired with an expert manual segmentation as the GT labeled by pathologists.
- **TNUI-2021 Dataset:** The third medical image dataset, called TNUI-2021. It contains 1381 thyroid nodule ultrasound images. We randomly divided it into a training set (i.e., 966 ultrasound images), a validation set (i.e., 276 ultrasound images) and a testing set (i.e., 139 ultrasound images) in a ratio of

7:2:1. Each ultrasound image is marked with a physician manual segmentation as the GT labeled by pathologists.

- **Evaluation Metrics:** To evaluate the segmentation performance of the proposed DM-Net, we utilized the Specificity, Sensitivity, Accuracy, and mean intersection-over-union (mIoU), as measurements for the medical image segmentation tasks.
- **Training phase and validation phase:** In our proposed DM-Net were trained on the ISIC-2017, the SCISC, and the TNUI-2021 training dataset using pixel-level labels. First, to further decrease the risk of overfitting, we augmented the training dataset using data augmentation processing, including horizontal image flip, vertical image flip, diagonal image flip, and random rotation. The Adam optimizer with a batch size of 16 was utilized to optimize the segmentation and classification networks separately. An initialized learning rate is set to 0.0001. Finally, we used a validation dataset to supervise the training processing to avoid our proposed DM-Net trapping in overfitting under the training phase. We implemented our framework and other comparison CNN models in PyTorch using one NVIDIA GTX 1080TI GPU.

**Testing phase.** In the testing phase, we also took advantage of the data augmentation strategy for the three datasets during testing, i.e., horizontal, vertical flips, and diagonal flips. After that, we averaged the predictions as the final segmentation results. All comparison CNN models adopt the same scheme during training and testing phases.

### 3.2 Segmentation Results

We compared the proposed DM-Net with other comparison state-of-the-art (SOTA) biomedical image segmentation CNN models, including U-Net [3], AG-Net [2], AttU-Net [9], M-Net [8], U-Net++ [10], U-Net3+ [11], CE-Net [7], and MB-DCNN [1]. Four metrics were used, including the mean Accuracy, mIoU, mean Sensitivity, and mean Specificity, to evaluate the segmentation performance of those methods. As for the skin lesion segmentation task, a quantitative comparison is shown in Table 1 of the proposed DM-Net, and other comparisons medical image segmentation CNN models on the ISIC-2017 dataset. As can be seen, the proposed DM-Net achieves comparable performance in terms of the mean Accuracy, mIoU, mean Sensitivity, and mean Specificity. Our proposed DM-Net achieves 0.9398, 0.8531, 0.8442, and 0.9869 in terms of the mean Accuracy, mIoU, mean Sensitivity, and mean Specificity, more accurate than several other comparison CNN models. Compared with the U-Net [3], the mean Accuracy increases from 0.9299 to 0.9398 by 0.19%, the mIoU increases from 0.8237 to 0.8531 by 2.94%, the mean Sensitivity increases from 0.8235 to 0.8442 by 2.07%, the mean Specificity increases from 0.9748 to 0.9869 by 1.21%, which indicates the advantage of our proposed DM-Net on the skin lesion segmentation task. As for the WBC segmentation task, a quantitative comparison result is shown in Table 2 of the proposed DM-Net and other comparison CNN models

**Table 4.** Ablation studies performance for each component of our proposed DM-Net on skin lesion segmentation from the ISIC-2017 dataset.

Methods	Accuracy	mIoU	Sensitivity
L-Net	0.883	0.819	0.798
R-Net	0.895	0.834	0.781
the L-Net ensemble with the R-Net	0.9234	0.823	0.802
DM-Net+CDI module	0.9324	0.8512	0.8421
DM-Net+CDI module+SAC module	<b>0.9398</b>	<b>0.8531</b>	<b>0.8442</b>

on the SCISC dataset. As can be seen, our DM-Net achieves the best performance in terms of mean Accuracy, mIoU, mean Sensitivity, and mean Specificity. Our model achieves 0.9905, 0.9273, 0.8868, 0.9509, and 0.9901 in terms of the mean Accuracy, mIoU, mean Sensitivity, and mean Specificity, more accurate than several other CNN models. Compared with the U-Net [3], the mean Accuracy increases from 0.9613 to 0.9905 by 2.92%, the mIoU increases from 0.871 to 0.9273 by 5.63%, the mean Sensitivity increases from 0.8697 to 0.9509 by 8.12%, the mean Specificity increases from 0.9663 to 0.9901 by 2.38%, which indicates the superiority of our proposed DM-Net on the WBC segmentation. As for the thyroid nodule segmentation task, a quantitative comparison result is shown in Table 3 of the proposed DM-Net and other comparison CNN models on the TNUI-2021 dataset. As can be seen, the proposed DM-Net achieves comparable performance in terms of mIoU, mean Sensitivity, and mean Specificity. Our proposed DM-Net achieves 0.9028, 0.8031, 0.9481, and 0.9997 in mIoU, mean Sensitivity, and mean Specificity, more accurate than the other several CNN models. Compared with the U-Net [3], the mIoU increases from 0.8555 to 0.9028 by 4.73%, the mean Sensitivity increases from 0.7821 to 0.9481 by 6.60%, the mean Specificity increases from 0.9977 to 0.9997 by 0.20%, which shows the superiority of our proposed DM-Net for thyroid nodule segmentation.

### 3.3 Ablative Evaluation on Segmentation

To verify the effectiveness of the contextual information encoding (CIE) module and the crossed skip connection (CSC) strategy on the segmentation task on the ISIC-2017 dataset, we further performed ablation experiments (see Table 4). Experiments show that both the CIE module and the CSC strategy can improve the biomedical image segmentation performance. Further improvements can be achieved when these two modules are used together.

## 4 Conclusions

In this study, we design a novel H-shaped network for automated skin lesion, WBC, and thyroid nodule segmentation. Specifically, we propose a novel CSC

strategy, which tends to bring better information fusion. In addition, we propose a CIE module, which can preserve more spatial information. Meanwhile, to alleviate the imbalance training problem, we design a novel hybrid loss. Experiments on three public medical image datasets show the effectiveness of the proposed DM-Net. In addition, the proposed DM-Net is validated on 2D data now, and the extension to 3D data would be a possible future study.

**Acknowledgments.** This study was supported by the National Natural Science Foundation of China under Grant 61901120 and 62171133.

## References

1. Xie, Y., Zhang, J., Xia, Y., Shen, C.: A mutual bootstrapping model for automated skin lesion segmentation and classification. *IEEE Trans. Med. Imaging* **39**(7), 2482–2493 (2020)
2. Zhang, S., et al.: Attention guided network for retinal image segmentation. In: Shen, D., et al. (eds.) *MICCAI 2019*. LNCS, vol. 11764, pp. 797–805. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-32239-7\\_88](https://doi.org/10.1007/978-3-030-32239-7_88)
3. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) *MICCAI 2015*. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-24574-4\\_28](https://doi.org/10.1007/978-3-319-24574-4_28)
4. Jha, D., Riegler, M., Johansen, D., Halvorsen, P., Johansen, H.: DoubleU-Net: a deep convolutional neural network for medical image segmentation. In: 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), pp. 558–564 (2020)
5. Jin, Q., Cui, H., Sun, C., Meng, Z., Su, R.: Cascade knowledge diffusion network for skin lesion diagnosis and segmentation. *Appl. Soft Comput.* **99**, 106881 (2021)
6. Zhang, Y., Lai, H., Yang, W.: Cascade UNet and CH-UNet for thyroid nodule segmentation and benign and malignant classification. In: Shusharina, N., Heinrich, M.P., Huang, R. (eds.) *MICCAI 2020*. LNCS, vol. 12587, pp. 129–134. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-71827-5\\_17](https://doi.org/10.1007/978-3-030-71827-5_17)
7. Gu, Z., Cheng, J., Fu, H., et al.: CE-Net: context encoder network for 2D medical image segmentation. *IEEE Trans. Med. Imaging* **38**(10), 2281–2292 (2019)
8. Fu, H., Cheng, J., Xu, Y., et al.: Joint optic disc and cup segmentation based on multi-label deep network and polar transformation. *IEEE Trans. Med. Imaging* **37**(7), 1597–1605 (2018)
9. Oktay, O., Schlemper, J., Folgoc, L., et al.: Attention U-Net: learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999* (2018)
10. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: UNet++: a nested U-Net architecture for medical image segmentation. In: Stoyanov, D., et al. (eds.) *DLMIA/ML-CDS -2018*. LNCS, vol. 11045, pp. 3–11. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-00889-5\\_1](https://doi.org/10.1007/978-3-030-00889-5_1)
11. Huang, H., Lin, L., Tong, R., et al.: UNet 3+: a full-scale connected UNet for medical image segmentation. In: 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1055–1059. IEEE (2020)
12. Li, X., Chen, H., Qi, X., Dou, Q., et al.: H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE Trans. Med. Imaging* **37**(12), 2663–2674 (2018)



13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
14. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning 2019, pp. 6105–6114. PMLR (2019)
15. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. In: Thirty-First AAAI (2017)
16. Chen, L., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint [arXiv:1706.05587](https://arxiv.org/abs/1706.05587) (2017)
17. Zhou, X., Li, Z., Xie, H., et al.: Leukocyte image segmentation based on adaptive histogram thresholding and contour detection. *Curr. Bioinform.* **15**(3), 187–195 (2020)
18. Zhou, X., Nie, X., Li, Z., et al.: H-Net: a dual-decoder enhanced FCNN for automated biomedical image diagnosis. *Inf. Sci.* **613**, 575–590 (2022)
19. Zhou, X., Li, Z., Tong, T.: DTSC-Net: semi-supervised 3D biomedical image segmentation through dual-teacher simplified consistency. In: 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1429–1434 (2022)
20. Li, Z., Lai, T., Zhou, X.: Saliency detection based on weighted saliency probability. In: 2019 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCLOUD/SocialCom/SustainCom), pp. 1550–1555 (2019)
21. Zhou, X., Wang, C., Li, Z., Zhang, F.: Adaptive histogram thresholding-based leukocyte image segmentation. In: Pan, J.-S., Li, J., Tsai, P.-W., Jain, L.C. (eds.) *Advances in Intelligent Information Hiding and Multimedia Signal Processing*. SIST, vol. 157, pp. 451–459. Springer, Singapore (2020). [https://doi.org/10.1007/978-981-13-9710-3\\_47](https://doi.org/10.1007/978-981-13-9710-3_47)
22. Zhou, X., Tong, T., Zhong, Z., et al.: Saliency-CCE: exploiting colour contextual extractor and saliency-based biomedical image segmentation. *Comput. Biol. Med.* **154**, 106551 (2023)
23. Zhou, X., Li, Z., Xue, Y., et al.: CUSS-Net: a cascaded unsupervised-based strategy and supervised network for biomedical image diagnosis and segmentation. *IEEE J. Biomed. Health Inform.* **154**, 1–12 (2023)



# MTGL-ADMET: A Novel Multi-task Graph Learning Framework for ADMET Prediction Enhanced by Status-Theory and Maximum Flow

Bing-Xue Du<sup>1</sup>, Yi Xu<sup>1</sup>, Siu-Ming Yiu<sup>2</sup>, Hui Yu<sup>3</sup>, and Jian-Yu Shi<sup>1</sup>

<sup>1</sup> School of Life Sciences, Northwestern Polytechnical University, Xi'an 710072, China  
jianyushi@nwpu.edu.cn

<sup>2</sup> Department of Computer Science, The University of Hong Kong, Hong Kong, China

<sup>3</sup> School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China  
huiyu@nwpu.edu.cn

**Abstract.** It is a vital step to evaluate drug-like compounds in terms of absorption, distribution, metabolism, excretion, and toxicity (ADMET) in drug design. Classical single-task learning based on abundant labels has achieved inspiring progress in predicting individual ADMET endpoints. Multi-task learning (MTL), having the low requirement of endpoint labels, can predict multiple ADMET endpoints simultaneously. Nonetheless, it is still an ongoing issue that the performance of existing MTL-based approaches depends on how appropriate participating tasks are. Furthermore, there is a need to elucidate what substructures are crucial to specific ADMET endpoints. To address these issues, this work constructs a Multi-Task Graph Learning framework for predicting multiple ADMET properties of drug-like small molecules (MTGL-ADMET) under a new paradigm of MTL, ‘one primary, multiple auxiliaries’. It first leverages the status theory and the maximum flow to select appropriate auxiliary tasks of a specific ADMET endpoint task. Then, it designs a novel primary-centered multi-task learning model, which consists of a task-shared atom embedding module, a task-specific molecular embedding module, a primary task-centered gating module, and a multi-task predictor. The comparison with state-of-the-art MTL-based methods demonstrates the superiority of MTGL-ADMET. More elaborate experiments validate its contributions, including the status theory-based auxiliary selection algorithm and the novel MTL architecture. Furthermore, a case study illustrates the interpretability of MTGL-ADMET by indicating crucial substructures w.r.t. the primary task. It’s anticipated that this work can boost pharmacokinetic and toxicity analysis in drug discovery. The code and data underlying this article are freely available at <https://github.com/dubingxue/MTGL-ADMET>.

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_6](https://doi.org/10.1007/978-3-031-29119-7_6).

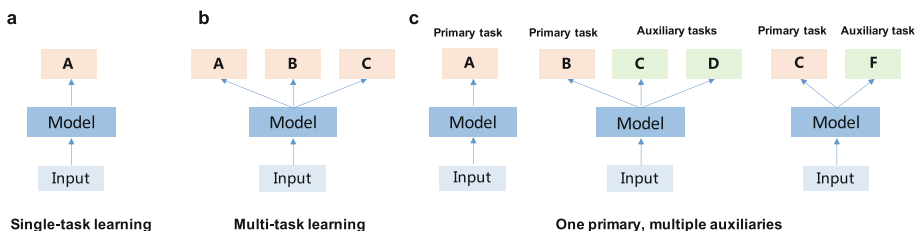
**Keywords:** ADMET prediction · Multi-tasking learning · Status Theory · Maximum Flow · Interpretability

## 1 Introduction

Pharmaceutical companies usually spend approximately 10 years and 1.1 billion dollars in discovering and developing a novel drug [1,2]. One of the undesired and painful events in such a costly process is the failure of drug candidates at clinical trials. It is mainly caused by undesirable pharmacokinetic (PK) properties or unacceptable toxicities [3]. By leveraging experimental and clinical data, AI-based computational methods are promising in a rapid and low-risk manner to predict the PK properties and toxicities of drug-like molecules before performing clinical trials [4,5]. Aiming to this perspective, machine learning-based (ML-based) methods, including classical shallow learning and modern deep learning, were popularly developed over the past years. Their typical tasks cover predicting Absorption, Distribution, Metabolism, Excretion, Toxicity (ADMET), and other physic-chemical properties of small-molecule compounds.

Through building a predictor for each ADMET endpoint task, ML-based approaches can precisely infer unknown properties for newly given compounds. Their success relies on known ADMET properties (labels) of compounds and molecular representation algorithms, where the latter is referred to as feature extraction in classical machine learning and as embedding in modern deep learning. For example, MoleculeNet contributes a library of machine learning-based ADMET predictive approaches [6], including classical machine learning (support vector machines, random forest, etc.) and modern deep learning (deep neural networks and graph neural networks (GNNs), etc.). In general, these methods belong to the paradigm of single-task learning (STL), ‘one model, one task’, where sufficient labels are one of the crucial factors when training a good predictive model (Fig. 1a). However, it is costly to acquire multiple molecular properties in most practical cases. The resulting scarce labels would cause poor molecular representations and trigger the overfitting issue, further resulting in a poor prediction of ADMET properties. To alleviate this predicament, two modern representation techniques, pre-training and fine-tuning, are utilized to achieve improved predictions in the case of scarce labels. The pre-training learns good initial molecular representations by abundant unlabeled data, while the fine-tuning further learns task-specific molecular representations [7–9]. In short, pre-training is to train a general set of parameters and fine-tune them for a specific task.

Multi-task learning (MTL) doesn’t need to require a compound to be measured all the ADMET properties. It solves multiple tasks at the same time while exploiting commonalities and differences across ADMET endpoint tasks. During its training, the underlying knowledge among ADMET endpoints can be transferred between them such that the issue of scarce labels can be compensated [10]. The superiority of MTL to STL is demonstrated by recent works in ADMET endpoint predictions [11]. In common, the models in these works first



**Fig. 1. Learning paradigms.** (a) ‘one model, one task’ in the single-task learning framework; (b) ‘one model fits all tasks’ in the popular multi-task learning framework; and (c) ‘one primary, multiple auxiliaries’ in our new multi-task learning framework.

design a GNNs model (such as graph convolution network (GCN) [12], relational graph convolution network (R-GCN) [13], and graph isomorphism network [14]) to extract task-shared embeddings. Then, they leverage parallel fully-connected neural networks to generate task-specific embeddings for multiple ADMET endpoints simultaneously. In short, existing MTL-based ADMET models follow the paradigm ‘one model fits all tasks’ (Fig. 1b). However, such joint learning cannot guarantee that one can always achieve better performance by MTL than that by STL w.r.t. a specific task, since it assumes that all the tasks have the same ranking or assume a certain learning trade-off among tasks (e.g., task-specific weights in the loss function) [10]. In other words, one model cannot fit all tasks in general. It depends on whether appropriate auxiliary tasks can be selected, which is still an ongoing issue [15].

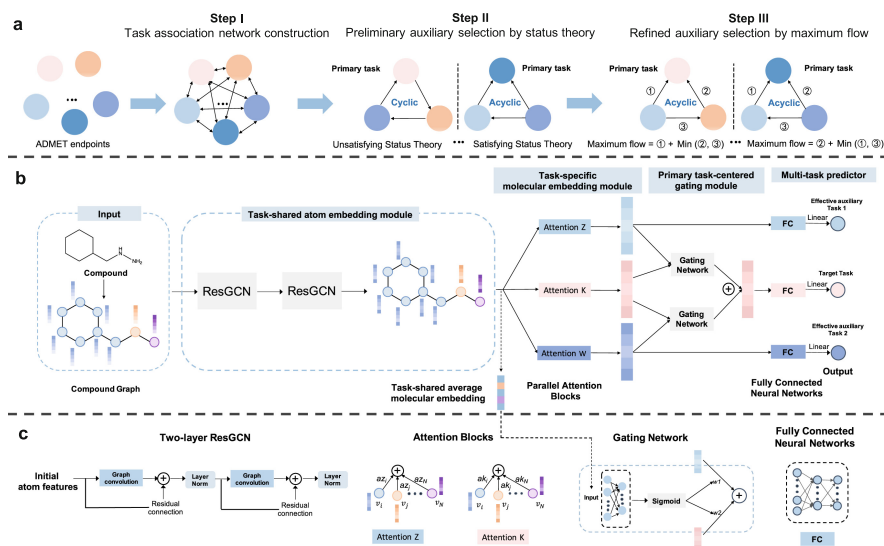
We hold two assumptions in mind. We first believe that the utilization of task associations can boost the selection of appropriate auxiliary tasks. For example, Cytochrome P450 enzymes (CYP450s) are hemoproteins that participate in the metabolism of compounds. Their inhibition can increase the plasma concentration (i.e., an endpoint of Distribution), reduce the Clearance (an endpoint of Excretion), and prolong the half-life (another endpoint of Excretion) of therapeutic agents [16]. The associations between tasks could help select auxiliary endpoint tasks. Moreover, we believe that the presence of certain substructures of a compound is strongly related to its PK endpoint properties. For instance, the absorption endpoint, Human Intestinal Absorption (HIA), is usually related to hydrophilicity functional groups [17]. Similarly, another case of excretion endpoint, Clearance, is usually related to lipophilicity functional groups [18]. The capture of important functional groups (i.e., task-specific crucial substructures) could provide insights into the underlying mechanism of different ADMET properties for a compound/drug.

Based on these ideas, we propose a new paradigm of MTL, ‘one primary, multiple auxiliaries’, where auxiliary tasks boost the primary task even with their own degradation (Fig. 1c). Under this paradigm, we construct a Multi-Task Graph Learning framework for predicting multiple ADMET properties of drug-like small molecules (MTGL-ADMET) in this work. We first build a task association network by training individual and pairwise tasks. Then we leverage the Status Theory and the Maximum Flow in complex network science to col-

lect appropriate tasks (i.e., friendly auxiliary tasks) for each specific task (i.e., the primary task). After that, we design a multi-task graph learning model to train the primary task and its auxiliary tasks together. The model technically includes a task-shared atom embedding module, a task-specific molecular embedding module, a primary task-centered gating module, and a multi-task predictor. In brief, our contributions are as follows. (1) Holding the paradigm ‘one primary, multiple auxiliaries’, we design a novel task selection algorithm, which utilizes the status theory to determine friendly auxiliaries of a specific task, and the maximum flow to estimate the increments of MTL w.r.t. STL. (2) We provide a novel model of MTL-based ADMET prediction, where atom embeddings are shared by multiple tasks and they are aggregated further into generating task-specific molecular embeddings. (3) By the aggregation weights of atoms, the proposed model provides an interpretable manner to indicate crucial compound substructures significantly associated with each ADMET task.

## 2 Methodology

### 2.1 Problem Formulation



**Fig. 2.** Overview of MTGL-ADMET (a) The friendly auxiliary task selection containing three sub-steps. (b-c) The novel multi-task learning model for both a primary task and its selected auxiliaries. It is an end-to-end model, which contains a task-shared atom feature module, a task-specific molecular feature module, a primary task-centered gating module, and a multi-task predictor module from left to right.

Given a set of  $n$  ADMET endpoint tasks  $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ , we construct  $n$  multi-task neural networks (MTNN)  $\mathcal{M} = \{m_1, m_2, \dots, m_n\}$  accordingly. Each MTNN  $m_k$  adopts a paradigm, ‘one primary, multiple auxiliaries’, which enhances the primary task by its auxiliary tasks (shown in Fig. 1). Suppose that a specific primary task  $t_k$  and its auxiliary tasks  $\mathcal{T}_k = \{t_1^{(k)}, t_2^{(k)} \dots\} \subseteq \mathcal{T}$ , where  $t_k$ ’s auxiliary task  $t_i^{(k)} \in \mathcal{T}$ . One of our goals is to determine  $\mathcal{T}_k$  among  $\mathcal{T}$  for  $t_k$  for further training  $m_k$ .

Given  $M$  compounds  $\{c_i, i = 1, 2, \dots, M\}$  and their ADMET properties  $y = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$  w.r.t.  $\mathcal{T}$ , where  $\mathbf{y}_i \in R^n$ ,  $\mathbf{y}_i(j) \in \{1, 0, -\}$  or  $\mathbf{y}_i(j) \in \mathcal{R}$ ,  $j \in \{1, 2, \dots, n\}$ , where ‘-’ has no property measured. The former type of  $\mathbf{y}_i(j)$  indicates the binary classification problem, which determines whether a molecule has an ADMET endpoint of interest or not. The latter accounts for the regression problem, which reflects how well its endpoint is. For example, *Phenobarbital*, an anticonvulsant drug, has significant Hepatotoxicity (i.e.,  $\mathbf{y}_i(j) = 1$ ) but none of the four kinds of Cardiotoxicities (i.e.,  $\mathbf{y}_i(j) = 0$ ) while exhibiting the value 3.16 of Lethal Dose 50 % (LD50), which is a measure of Acute oral toxicity (i.e.,  $\mathbf{y}_i(j) = 3.16$ ), but has no other measured property (i.e.,  $\mathbf{y}_i(j) = \text{‘-’}$ ). One of our goals is to predict  $n$  ADMET properties  $\mathbf{y}_x$  of a new coming compound  $c_x$  by the well-trained MTNNs  $\{m_k\}$ . For this purpose, we design a Multi-Task Graph Learning framework enhanced by Status Theory and Maximum Flow (MTGL-ADMET) (shown in Fig. 2).

## 2.2 Overview of Framework

As shown in Fig. 2, our MTGL-ADMET model consists of two stages. One is the friendly auxiliary tasks selection while another is a novel multi-task learning model for both a primary task and its selected auxiliaries. The auxiliary task selection (Sect. 2.3) contains three steps. **Step I:** We leverage the differences between the performances of individual tasks and those of pairwise tasks to calculate inter-task associations. **Step II:** we perform a preliminary selection of auxiliary tasks for each task by the status theory [19]. **Step III:** we run a further selection of friendly auxiliary tasks by the maximum flow calculation [20]. The novel multi-task learning model is designed for each task group under the paradigm ‘one primary task, multiple auxiliary tasks’ (Sect. 2.4). It contains four components, including a task-shared atom embedding module, a task-specific molecular embedding module, a primary task-centered gating module, and a multi-task predictor.

## 2.3 Auxiliary Task Selection

**Task Association Network Construction.** To investigate how well a task can boost another one, we measure the differences between the performances of individual ADMET tasks and their corresponding pairwise tasks. These differences are regarded as inter-task associations. Suppose that  $t_w$  and  $t_k$  are two tasks,  $\mathcal{S}_w$  and  $\mathcal{S}_k$  are two single-task learning models accounting for  $t_w$  and  $t_k$

individually, and  $\mathcal{D}_{w,k}$  is a dual-task learning model training the tasks simultaneously (Fig. S1). These learning models contain a two-layer ResGCN, an attention block, and fully connected neural networks. We design an index  $\hat{Z}_{w \rightarrow k}$  to reflect the fluence of  $t_w$  on  $t_k$  as follows,

$$\hat{Z}_{w \rightarrow k} = Z_{k|w}^{(d)} - Z_k^{(s)} \quad (1)$$

where  $Z_k^{(s)}$  is the performance of  $t_k$  given by  $\mathcal{S}_k$ , and  $Z_{k|w}^{(d)}$  is the performance of  $t_k$  given by  $\mathcal{D}_{w,k}$ . Similarly, we can define  $\hat{Z}_{k \rightarrow w} = (Z_{w|k}^{(d)} - Z_w^{(s)})$  to reflect how  $t_k$  fluences  $t_w$ , where  $Z_{w|k}^{(d)}$  is the performance of  $t_w$  given by  $\mathcal{D}_{w,k}$ . In general,  $\hat{Z}_{t_w \rightarrow t_k} \neq \hat{Z}_{t_k \rightarrow t_w}$ , since the influence between two tasks is asymmetric. Moreover, such an influence could be either positive or negative. If  $\hat{Z}_{w \rightarrow k} > 0$ ,  $t_w$  boosts  $t_k$ ; otherwise,  $t_w$  depresses  $t_k$ . Finally, regarding these differences as inter-task associations among all the tasks  $\mathcal{T}$ , we organize them into a bi-directed and signed network, where nodes are ADMET tasks and edges are inter-task associations. The adjacent matrix of the task association network is illustrated in Fig. S2. Based on the task association network, a preliminary selection of primary task-auxiliary tasks shall be performed by leveraging complex network analysis.

**Preliminary Auxiliary Selection by Status Theory.** Inspired by status theory in social network [19], we perform a preliminary selection of auxiliaries for each task. The status theory states the rule of ‘the person respected by me should have higher status than me’ where social status represents the prestige of nodes (persons) in a social network. Analogously, the status represents the ranking of tasks in the ADMET task association network. Holding the paradigm ‘one primary, multiple auxiliaries’, we attempt to construct a primary task-specific pool containing a high-ranking primary and its low-ranking auxiliaries. The status theory helps find the lower-ranking auxiliaries. In terms of  $\hat{Z}_{w \rightarrow k}$ , we empirically selected  $\leq 5$  auxiliary tasks in descending order with considering the high complexity of status theory-satisfied triads in the case of multiple auxiliaries.

Given a primary task  $t_k$ , our task is to select a set of auxiliary tasks  $\overline{T}_k = \{t_i\} \subset \mathcal{T}$ ,  $i \neq k$  and  $i = 1, 2, \dots, n$ . We consider two tasks  $t_w \in \overline{T}_k$  and  $t_z \in \overline{T}_k$  accompanying with  $t_k$  to form a triad. There are two types of triads in directed networks, which correspond to acyclic and cyclic triads (Fig. 2a). If the triad is acyclic, then the triad satisfies the status theory [19]. In this case, the status theory implies that both  $t_w$  and  $t_z$  have higher status than  $t_k$  if there are positive associations from them to  $t_k$  or negative associations from  $t_k$  to them respectively. However, we cannot determine the result directly because the inter-task associations are signed and bi-directed. Thus, we first turn to bi-directed associations among  $t_k$ ,  $t_w$  and  $t_z$  into mono-directed associations as follows,

$$e_{i,j} = \hat{Z}_{i \rightarrow j} - \hat{Z}_{j \rightarrow i}, i, j \in \{w, z, k\} \quad (2)$$

where  $e_{i,j}$  is the mono-directed association between  $t_i$  and  $t_j$ . Then, we reverse the directions of all negative  $e_{i,j}$  and flip their signs to positive (i.e.,  $e_{j,i} = -e_{i,j}$ , if  $e_{i,j} < 0$ ). So far, the task association network can be treated as a pure directed network when omitting positive signs. Sequentially, it can examine whether a given triad satisfies the status theory (acyclic) or not (cyclic). In this manner, acyclic triads w.r.t. the primary task  $t_k$  could be found to expand the auxiliary group.

**Refined Auxiliary Selection by Maximum Flow.** Holding the paradigm ‘one primary, multiple auxiliaries’, we suppose that  $t_k$  is the primary task, and two other tasks  $t_w$  and  $t_z$  are its auxiliaries. Let  $M_{k|w,z}$  be a multi-task learning model accounting for them,  $Z_{k|w,z}^{(m)}$  be the performance of  $t_k$  obtained by the model. The status theory guarantees that mono-directed associations flow from all the auxiliaries to the primary task, but the corresponding bi-directed inter-task associations could be positive or negative. In details, each mono-directional association to  $t_k$  ( $\hat{Z}_{i \rightarrow k} - \hat{Z}_{k \rightarrow i} > 0$ ) accounts for three scenarios of bi-directional inter-task associations, including (1) ( $\hat{Z}_{i \rightarrow k} > 0$  &  $\hat{Z}_{k \rightarrow i} < 0$ ), (2) ( $\hat{Z}_{i \rightarrow k} > 0$  &  $\hat{Z}_{k \rightarrow i} > 0$ ), (3) ( $\hat{Z}_{i \rightarrow k} < 0$  &  $\hat{Z}_{k \rightarrow i} < 0$ ), where  $i \in \{w, z\}$ . Naturally, we desire positive inter-task associations from the auxiliaries to the primary, which reflects that the auxiliary task  $t_i$  boosts the primary task  $t_k$ . Thus, we select the first two scenarios with the top priority to determine two auxiliaries of a specific primary task  $t_k$ .

More importantly, since we also expect that  $Z_{k|w,z}^{(m)} \geq \text{Max} \left( Z_{k|w}^{(d)}, Z_{k|z}^{(d)} \right)$  at the same time, we calculate the maximum flow for the triad based on Ford-Fulkerson algorithm [20], where  $t_k$  is the sink node, one of its auxiliaries is the source node (e.g.,  $t_i$ ) and another is the intermedia node (e.g.,  $t_j$ ),  $i, j \in \{w, z\}$ . We directly regard mono-directed associations  $\{e_{i,j}\}$  as the flux between two tasks. Specifically,  $e_{i,k}$  is the flux from  $t_i$  to  $t_k$ , and  $e_{j,k}$  is the flux from  $t_j$  to  $t_k$ . Thus, we define the maximum input flux (e.g.,  $f_k^{\max}(i, j)$ ) of  $t_k$  given the source  $t_i$  and the sink  $t_j$  as follows,

$$f_{k|i,j}^{\max} = e_{i,k} + \min(e_{i,j}, e_{j,k}) \quad (3)$$

In the case that  $t_k$  has more than 2 auxiliaries, we apply the Ford-Fulkerson algorithm [20] to calculate the maximum flux. Finally, according to the value of  $f_k^{\max}(i, j)$  of multiple combinations from 2 to 5 auxiliaries in descending order, we determine the auxiliaries w.r.t.  $t_k$ .

## 2.4 Multi-task Graph Learning Model

The multi-task graph learning model contains a task-shared atom embedding module, a task-specific molecular embedding module, a primary task-centered gating module, and a multi-task predictor module (Fig. 2b–2c).



**Task-Shared Atom Embedding Module.** By adopting two-layer residual GCN (ResGCN), the task-shared atom embedding module learns atom embedding representation by aggregating neighboring atom features on molecule graphs, which are shared by both the primary task and its auxiliaries. According to chemical structure, each compound  $c$  is represented as a molecule graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  is a set of atoms and  $\mathcal{E}$  is a set of chemical bonds. Let  $\mathbf{A} \in \mathbb{R}^{N \times N}$  ( $N = |\mathcal{V}|$ ) be its adjacency matrix, in which  $a_{ij} = 1$  indicates the occurring bond ( $e_{ij} \in \mathcal{E}$ ) between atom  $i$  ( $v_i$ ) and atom  $j$  ( $v_j$ ), and  $a_{ij} = 0$  indicates no bond. Here, each node  $v_i$  (atom) is initially represented by a  $q$ -dimensional binary feature vector  $\mathbf{h}_i \in \mathbb{R}^q$ . As suggested in [21], the initial node features typically include the atom symbol, the number of adjacent atoms, the number of adjacent hydrogens, the implicit value of the atom, and the atom occurrence in an aromatic structure. For each atom  $v_i$  in the molecule graph  $\mathcal{G}$  of compound  $c$ , each layer of the ResGCN updates its features  $\mathbf{h}_i^c \in \mathbb{R}^{1 \times d}$  by aggregating the embeddings of its neighboring atoms and adding a residual connection from the previous layer [22]. The aggregation update rule for atom embedding on the  $k$ th layer in the matrix form is defined as follows:

$$\mathbf{H}^{(k+1)} = \sigma \left( \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(k)} \mathbf{W}^{(k)} \right) + \sigma \left( \tilde{\mathbf{H}}^{(k)} \right), k = 0, 1 \quad (4)$$

where  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ ,  $\mathbf{I}_N$  is the identity matrix,  $\mathbf{D} \in \mathbb{R}^{N \times N}$  is the degree matrix, in which diagonal elements are the degrees of each vertex and  $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ ,  $\mathbf{W}^{(k)}$  is the layer-wise trainable weight matrix, and  $\sigma(\cdot)$  denotes an activation function.

Note that the atom embedding vector  $\mathbf{h}_i^c$  (i.e., each row of  $\mathbf{H}$ ) is shared by all the tasks. In classical single-task learning, the molecular embedding  $\bar{\mathbf{h}}$  can be obtained by a popular readout, which is simply generated by averaging the task-shared atom embeddings of compound  $c$ . In contrast, the current multi-task learning requires task-specific molecular embedding, which shall be obtained in the next section.

**Task-Specific Molecular Embedding Module.** The task-specific molecular embedding module leverages parallel attention layers to learn task-specific molecular representations ( $\mathbf{h}_k, \mathbf{h}_i^{(k)}$ ) of compound  $c$ . The idea seems like a multi-head attention (i.e., a concatenation of parallel attentions). However, considering a different manner, we don't concatenate atom embeddings weighted by parallel task-specific attention layers, but take the weights on the same set of task-shared atom embeddings to differentiate atom embeddings w.r.t. tasks. Then, the task-specific molecular representations can be obtained by a classical readout on them.

Suppose that compound  $c$  contains  $N$  atoms, of which each is encoded into  $p$ -dimensional vectors  $\mathbf{h}_i^c \in \mathbb{R}^{p \times 1}$  by the task-shared atom embedding module. With regard to a specific task  $t_z$ , its attention weights are implemented by a forward-feed neural network as follows:

$$\mathbf{a}_z = \text{Sigmoid}(\mathbf{W}_z \cdot \mathbf{H} + \mathbf{b}_z), \sum_{i=1}^N a_z(i) = 1 \quad (5)$$

where  $\mathbf{W}_z \in \mathbb{R}^{1 \times p}$  is the learnable weight matrix and  $\mathbf{b}_z \in \mathbb{R}^{1 \times N}$  is the bias vector,  $\mathbf{H} \in \mathbb{R}^{p \times N}$  is the task-shared atom embedding matrix (stacked by  $\{\mathbf{h}_i^c \in \mathbb{R}^{p \times 1}\}$  column by column, a transposed form of that in the last section),  $\mathbf{a}_z \in \mathbb{R}^{1 \times N}$  is the atom weight vector, of which each element  $a_z(i)$  accounts for the  $i$ -th atom in the compound. Thus, the task-specific molecular embedding w.r.t.  $t_z$  (denoted as  $\mathbf{h}_z$ ) can be calculated in the following form:

$$\mathbf{h}_z = \sum_{i=1}^N a_z(i) \mathbf{h}_i^c \quad (6)$$

More importantly, task-specific atom weights facilitate finding crucial substructures w.r.t. tasks, where two bonding atoms are regarded as a crucial structure fragment if both of them have high weights (See Sect. 3.4).

**Primary Task-Centered Gating Module.** Under the paradigm ‘one primary, multiple auxiliaries’, the primary task-centered gating module learns how each auxiliary task contributes to the primary task and how these contributions are combined by a set of gating networks. Inspired by [23], each gating network  $G$  is simply composed of a single-layer feed-forward network and a Sigmoid activation function. It takes the task-shared average molecular embedding  $\bar{\mathbf{h}}$  of compound  $c$  as the input and outputs two scalar weights, of which one is for the primary task and another is for an auxiliary. The weighted embedding of the auxiliaries with regard to the primary task is taken as its contribution to the primary task. Sequentially, all the contributed embeddings of the primary task are summed up as their final embeddings (Fig. 2c). Let  $G_i$  be a gating network containing a fully connected layer  $FC_G^i$ , which accounts for each primary-center task pair be  $(t_k, t_i^{(k)})$  where  $t_k$  is the primary task,  $t_i^{(k)} \in \mathcal{T}$  is its  $i$ -th auxiliary, and  $i = 1, 2, \dots, \left| \left\{ t_i^{(k)} \right\} \right|$ . With regard to  $t_k$  and  $t_i^{(k)}$ , we suppose that  $w_k^{(i)}, w_i^{(k)}$  are their weights and  $\mathbf{h}_k, \mathbf{h}_i^{(k)}$  are their task-specific embeddings respectively. In addition, let  $\bar{\mathbf{h}} \in \mathbb{R}^{p \times 1}$  be the task-shared molecular embedding, which is simply generated by averaging the task-shared atom embeddings of compound  $c$  (i.e., a popular readout). Formally, the weights of the primary task and its auxiliary  $t_i^{(k)}$  are defined by a neural network as follows,

$$\left[ w_k^{(i)}, w_i^{(k)} \right] = \text{Softmax} \left( FC_G^i(\bar{\mathbf{h}}) \right) \quad (7)$$

Furthermore, with the contribution of  $t_i^{(k)}$  to  $t_k$ , its embedding  $\mathbf{h}_{i \rightarrow k}$  is defined as

$$\mathbf{h}_{i \rightarrow k} = w_k^{(i)} \mathbf{h}_k + w_i^{(k)} \mathbf{h}_i^{(k)}, w_k^{(i)} + w_i^{(k)} = 1 \quad (8)$$

Thus, the final embedding of the primary task passing through all the gating networks  $\{G_i\}$  is as follows:

$$\mathbf{h}_k^* = \sum_{i=1} \mathbf{h}_{i \rightarrow k} \quad (9)$$

Once the primary task-centered gating module is done,  $\mathbf{h}_k^*$  and  $\{\mathbf{h}_i^*\}$  are passed into task-specific towers (e.g., implemented by fully-connected neural networks) to predict task labels.

**Multi-task Predictor.** Either the primary task or its auxiliaries have individual predictors, which contain task-specific fully-connected neural networks to learn better task-specific nonlinear representation. We implement these neural networks (NNs) with the same architecture, which contains an input layer, a hidden layer, and an output layer. The predictor concerning the primary task  $t_k$  maps the contributed embeddings  $\mathbf{h}_k^*$  into the predicted primary task label  $\hat{y}_k$  (i.e.,  $\hat{y}_k = NN_k(\mathbf{h}_k^*)$ ), while those predictors accounting for the auxiliaries  $\{t_i^{(k)}\}$  maps  $\{\mathbf{h}_i^{(k)}\}$  into corresponding auxiliary task labels (i.e.,  $\{\hat{y}_i^{(k)} = NN_i^{(k)}(\mathbf{h}_i^{(k)})\}$ ). Furthermore, considering these tasks are of classification or regression, we use the Cross-Entropy loss for classification tasks and the Mean Squared Error loss for regression tasks when training multiple tasks together. When training MTGL-ADMET, the loss of multi-task learning is defined as follows,

$$\text{loss} = \sum_{c=1}^C \sum_{n=1}^{M_c} (-[p_c y_c \cdot \log \sigma(\hat{y}_c) + (1 - y_c) \cdot \log(1 - \sigma(\hat{y}_c))]) + \sum_{r=1}^R \sum_{n=1}^{M_r} (\hat{y}_r - y_r)^2 \quad (10)$$

here  $y_c$  and  $\hat{y}_c$  are the true label and the predictor value of compound  $c_n$  w.r.t. classification task  $t_c$  respectively,  $y_r$  is the true property value of  $c_n$  w.r.t. regression task  $t_r$ ,  $\hat{y}_r$  is the corresponding predicted value,  $C$  is the number of classification tasks and  $M_c$  is the number of compounds in classification tasks.  $R$  is the number of regression tasks and  $M_r$  is the number of compounds in regression tasks. Inspired by [24], to alleviate the imbalance of positive and negative samples in classification tasks, we utilize a weight  $p_c$  in the loss function, which is defined as the ratio of the number of negative samples to that of positive samples w.r.t. classification task  $t_c$ .

## 3 Experiments

### 3.1 Dataset and Setup

To evaluate our MTGL-ADMET, we built a dataset covering 24 endpoints (18 for classification and 6 for regression) from 8 publications. The dataset contains five Absorption [25, 26], two Distribution [27, 28], five Metabolism [24], two Excretion [29], eight Toxicity [24] and two ADMET-related Physicochemical properties [30, 31]. There are 43,291 drug-like compounds across 24 endpoint tasks in total, including 28,153 compounds in classification tasks and 16,545 in regression tasks, where a compound may have one or more endpoint labels (Table S1). Each node of the input drug/compound was initially represented by a 40-dimensional (40-d) binary atom feature vector, including atom symbol (16-d), degree (7-d), formal charge (1-d), radical electrons (1-d), hybridization (6-d), aromaticity (1-d), hydrogens (5-d), chirality (1-d) and chirality type (2-d), as suggested in [24]. Furthermore, the two-layer ResGCN encodes each compound into a 64-dimensional embedding vector. After that, through parallel attention blocks on atoms of each task, each compound is represented as a 64-dimensional embedding vector. In the meanwhile, the input feature of FC in the gating network is the

task-shared average molecular representation, which is also a 64-dimensional embedding vector and the output feature is a 2-dimensional embedding vector. In the  $n$ -th task predictor module, of which the input layer, the two-hidden layer, and the output layer contain 64, 128, 128, and 1 neurons respectively. In addition, we adopted an empirical setting for the parameters in the training, which assigns the batch size of 128, the epoch number of 200, the learning rate of 0.001, and selects Adam as the optimizer.

**Table 1.** Performance comparisons of MTGL-ADMET and baselines on 24 ADMET endpoint

No.	Endpoint	Metric	ST-GCN	ST-MGA	MT-GCN	MT-GCNAtt	MGA	MTGL-ADMET <sup>a</sup>
1	HIA	AUC	0.916 ± 0.054	<u>0.972 ± 0.014</u>	0.899 ± 0.057	0.953 ± 0.019	0.911 ± 0.034	<b>0.981 ± 0.011</b> (18)
2	OB	AUC	0.716 ± 0.035	0.710 ± 0.035	0.728 ± 0.031	0.726 ± 0.027	<u>0.745 ± 0.029</u>	<b>0.749 ± 0.022</b> (14, 24)
3	P-gp inhibitor	AUC	0.916 ± 0.012	<u>0.917 ± 0.006</u>	0.895 ± 0.014	0.907 ± 0.009	0.901 ± 0.010	<b>0.928 ± 0.008</b> (None)
4	P-gp substrates	AUC	<u>0.775 ± 0.034</u>	0.755 ± 0.014	0.733 ± 0.044	0.730 ± 0.034	0.719 ± 0.035	<b>0.801 ± 0.031</b> (18, 21)
5	Caco-2 permeability	R <sup>2</sup>	0.451 ± 0.033	<u>0.519 ± 0.014</u>	0.374 ± 0.022	0.404 ± 0.017	0.385 ± 0.031	<b>0.523 ± 0.025</b> (24)
6	PPB	R <sup>2</sup>	0.577 ± 0.028	0.585 ± 0.004	0.589 ± 0.036	<u>0.619 ± 0.025</u>	0.568 ± 0.038	<b>0.626 ± 0.029</b> (9)
7	BBB	AUC	0.956 ± 0.008	<u>0.959 ± 0.004</u>	0.945 ± 0.007	0.955 ± 0.009	0.956 ± 0.010	<b>0.973 ± 0.005</b> (23)
8	CYP1A2 inhibitor	AUC	0.932 ± 0.007	0.931 ± 0.013	0.914 ± 0.009	<u>0.941 ± 0.008</u>	0.940 ± 0.006	<b>0.952 ± 0.005</b> (9)
9	CYP2C19 inhibitor	AUC	0.774 ± 0.012	0.781 ± 0.008	0.775 ± 0.011	0.782 ± 0.011	<u>0.795 ± 0.019</u>	<b>0.804 ± 0.015</b> (12, 16)
10	CYP2C9 inhibitor	AUC	0.746 ± 0.016	0.764 ± 0.017	0.771 ± 0.016	0.782 ± 0.011	<b>0.798 ± 0.019</b>	<u>0.794 ± 0.013</u> (5, 6, 11, 16)
11	CYP2D6 inhibitor	AUC	0.848 ± 0.016	0.841 ± 0.022	0.839 ± 0.015	0.845 ± 0.015	<b>0.877 ± 0.017</b>	<u>0.869 ± 0.016</u> (5, 6)
12	CYP3A4 inhibitor	AUC	0.892 ± 0.006	<u>0.915 ± 0.006</u>	0.865 ± 0.007	0.896 ± 0.011	0.875 ± 0.006	<b>0.916 ± 0.007</b> (11)
13	Half life	AUC	<u>0.725 ± 0.011</u>	0.708 ± 0.024	0.688 ± 0.035	0.699 ± 0.028	0.707 ± 0.017	<b>0.727 ± 0.022</b> (14)
14	Clearance	AUC	0.723 ± 0.030	0.710 ± 0.015	0.686 ± 0.031	<u>0.755 ± 0.014</u>	0.740 ± 0.027	<b>0.779 ± 0.027</b> (12, 22, 24)
15	Hepatotoxicity	AUC	0.653 ± 0.040	0.669 ± 0.022	0.612 ± 0.039	0.640 ± 0.068	<b>0.713 ± 0.053</b>	<u>0.701 ± 0.036</u> (8, 10, 17)
16	Respiratory toxicity	AUC	0.842 ± 0.018	<b>0.872 ± 0.013</b>	0.810 ± 0.014	0.828 ± 0.015	0.828 ± 0.021	<u>0.859 ± 0.010</u> (None)
17	Cardiotoxicity-1	AUC	<u>0.707 ± 0.026</u>	0.703 ± 0.020	0.683 ± 0.028	0.696 ± 0.028	0.684 ± 0.023	<b>0.740 ± 0.023</b> (2, 16, 23)
18	Cardiotoxicity-5	AUC	0.620 ± 0.015	<u>0.637 ± 0.010</u>	0.626 ± 0.027	0.619 ± 0.015	0.623 ± 0.014	<b>0.641 ± 0.014</b> (9, 14, 19)
19	Cardiotoxicity-10	AUC	<u>0.627 ± 0.013</u>	0.611 ± 0.015	0.609 ± 0.022	0.613 ± 0.021	0.603 ± 0.026	<b>0.654 ± 0.010</b> (9, 10)
20	Cardiotoxicity-30	AUC	0.664 ± 0.036	0.653 ± 0.036	0.645 ± 0.036	0.687 ± 0.059	<u>0.709 ± 0.035</u>	<b>0.723 ± 0.029</b> (6, 11, 23)
21	LD50	R <sup>2</sup>	0.588 ± 0.018	<u>0.617 ± 0.018</u>	0.503 ± 0.017	0.502 ± 0.023	0.492 ± 0.029	<b>0.635 ± 0.015</b> (16, 22)
22	IGC50	R <sup>2</sup>	0.703 ± 0.055	<u>0.818 ± 0.021</u>	0.618 ± 0.027	0.744 ± 0.032	0.772 ± 0.021	<b>0.819 ± 0.008</b> (5, 19)
23	ESOL	R <sup>2</sup>	0.814 ± 0.030	<u>0.896 ± 0.013</u>	0.824 ± 0.030	0.872 ± 0.018	0.866 ± 0.020	<b>0.931 ± 0.038</b> (21, 22)
24	logD7.4	R <sup>2</sup>	0.759 ± 0.056	<u>0.904 ± 0.008</u>	0.770 ± 0.019	0.838 ± 0.016	0.838 ± 0.018	<b>0.915 ± 0.008</b> (1, 16)
	Average		0.747 ± 0.025	<u>0.769 ± 0.015</u>	0.725 ± 0.006	0.752 ± 0.022	0.752 ± 0.023	<b>0.793 ± 0.018</b>

<sup>a</sup>The numbers in the parenthesis means the auxiliary task numbers for each primary task in our model.

### 3.2 Comparisons with State-of-the-Art

We assessed our MTGL-ADMET by the comparison with three state-of-the-art multi-task learning models, which commonly use GNN but different MTL architectures under the paradigm ‘one model fits all tasks’. They are briefly summarized as follows:

MT-GCN [32]: It extends GCNs into the MTL architecture. It utilizes a two-layer GCN module with mean pooling and maximum pooling to generate task-shared molecular embeddings and a group of task-specific four-layer fully-connected neural networks to generate task-specific molecular embeddings. Its default values of parameters were used to train the model in the following experiments. Also, we denote its single-task form as ST-GCN.

MT-GCNAtt: We designed an extension of MT-GCN, MT-GCNAtt, by adding an extra attention block for each task after its GCN module. The parameters in the attention blocks are the same as those in our model while other parameters are directly taken from the original MT-GCN.

MGA [13]: Adopting a similar architecture to MT-GCN, it uses two-layer R-GCNs [33] (an extension of regular GCNs) to generate task-shared molecular embeddings and builds a task-specific attention layer followed by a three-layer fully-connected neural network for each task. We used all the default values of parameters to train the MGA in the following experiments. Meanwhile, we adopted its single-task form as ST-MGA.

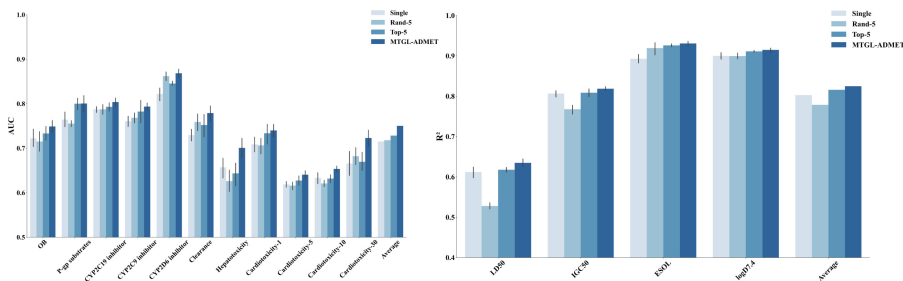
For a fair comparison, we utilized 10 independent experiments for all the methods. In each round of the experiments, the dataset was randomly split into a training set, a validation set, and a testing set by the ratio of 8:1:1 in terms of sample number. The validation set was used to select tasks. We run 10 repetitions under different random seeds and measured their performance by the average Area Under the Receiver Operating Characteristic Curve (AUC) for classification tasks, and the square determination coefficient ( $R^2$ ) for regression tasks respectively (Table 1). The greater, the better. We highlighted the result of best in bold and the second best in underline, and the numbers in brackets mean serial numbers of auxiliary tasks w.r.t. each primary task in our model.

The results show that MT-GCN has the worse performance, MT-GCNAtt and MGA exhibit similar performances, and our method achieves the best performance on average with significant improvements over these five approaches by 4.6%, 2.4%, 6.8%, 4.1%, and 4.1% respectively. Meanwhile, the performance of ST-GCN and ST-MGA is better than their MTL forms, MT-GCN and MGA on average. In terms of individual endpoints, ST-GCN achieves the second best over 4 tasks only, ST-MGA wins the best over 1 task and the second best over 10 tasks, MT-GCN achieves the worst on all tasks, MT-GCNAtt achieves the second best over 3 tasks, while MGA wins the best over 3 tasks and the second best over 3 tasks. In contrast, our MTGL-ADMET wins the best over 20 tasks and the second best over the remaining 4 tasks. Therefore, the comparison demonstrates the superiority of our MTGL-ADMET.

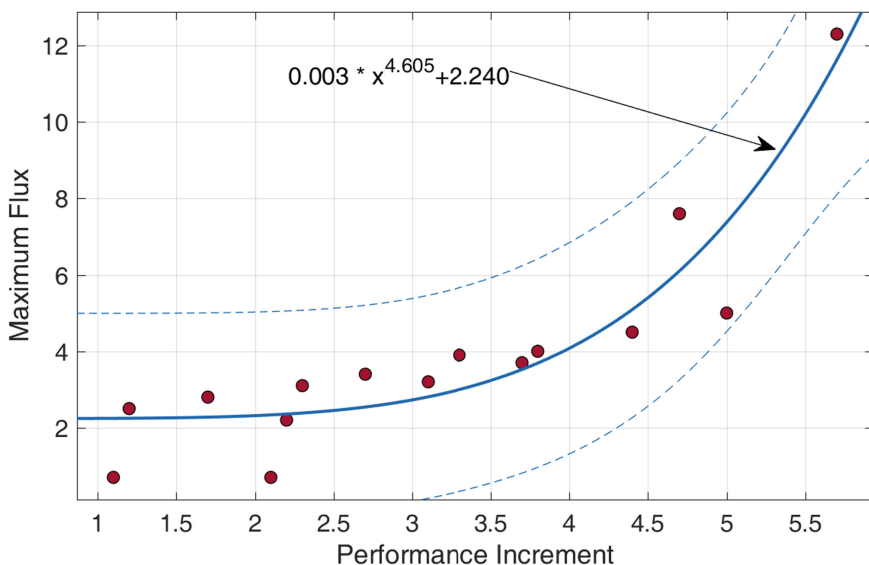
### 3.3 Ablation Studies

We evaluated the proposed selection of primary-specific auxiliary tasks by the comparison with three selection strategies, including an individual-task strategy and two designated multiple-task strategies (Fig. 3). The individual-task strategy, denoted as Single, trained predictors independently for individual ADMET endpoint tasks. The second strategy (denoted as Rand-5) randomly selected five auxiliaries for a specific task  $t_k$ . The third one (denoted as Top-5) selected at most top-5 positive auxiliary tasks of  $t_k$  according to the descending order of  $\{\hat{Z}_{i \rightarrow k}\}$  and  $\hat{Z}_{i \rightarrow k} > 0$  (i.e., the fluence of auxiliary  $t_i$  on  $t_k$ , see also Sect. 2.3). In contrast, our selection algorithm results in different numbers of auxiliary endpoints w.r.t. a specific endpoint. See the last column in Table 1.

Overall, our status theory-based selection outperforms these selection strategies and our MTGL-ADMET wins the best over all the tasks. Our method achieves the best performance on average with significant improvements over Single, Rand-5, and Top-5 by 3.16%, 3.60%, and 1.85% respectively. Especially, our selection algorithm remarkably improves the prediction on two classification tasks ('Hepatotoxicity', Cardiotoxicity-30) with 4.4% and 4.1% increments than the second best. More details can be found in Table S2.



**Fig. 3.** Comparison with auxiliary selection strategies. The left panel is for classification tasks while the right panel is for regression tasks. Compared with the MTGL-ADMET selection strategies, Single, Rand-5 randomly selected five auxiliaries, and Top-5 selected at most top-5 positive auxiliary tasks.



**Fig. 4.** Correlation between the maximum fluxes and the incremental performance. Each point represents a task group, which contains at least 2 auxiliary tasks. The maximum flux is calculated by the Ford-Fulkerson algorithm. The solid curve indicates a fitting while two dotted curves denote its 95% confidence bounds

In detail, there are 2 tasks, ‘P-gp inhibitor’ and ‘Respiratory toxicity’, having no appropriate auxiliary, 7 tasks having only one auxiliary, 9 tasks having only two auxiliaries, 5 tasks having three auxiliaries, 1 task having four auxiliaries (‘CYP2C9 inhibitor’). These results demonstrate that our selection algorithm relying on the status theory and the maximum flow can select approximate tasks adaptively. It significantly outperforms these strategies over all the endpoint tasks, no matter whether it is a classification task or a regression task.

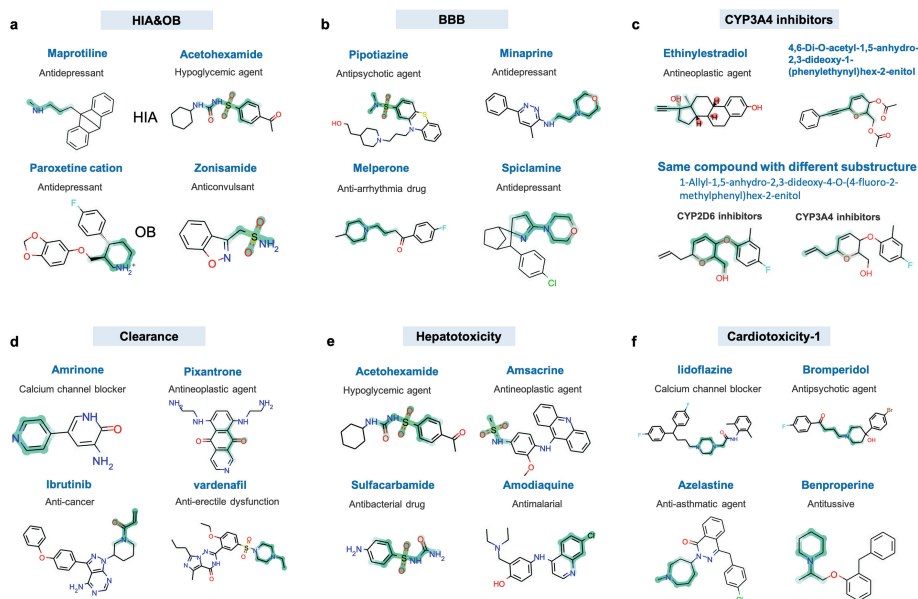
To investigate why our task selection algorithm strategy is effective, we first calculated the maximum flux over all the primary-specific task groups. Then, we measure the correlation between them and the incremental performance of the multi-task learning to the single-task learning on the testing dataset, i.e.  $Z_{k|w,z}^{(m)} - Z_k^{(s)}$ . We found a significant Spearman correlation ( $\gamma = 0.9562, \rho = 2.6089e - 08$ ) between the maximum flux and the increment (Fig. 4). The results demonstrate that the maximum flow on status-shaped triads can be a good indicator to select appropriate auxiliary tasks for a specific task. More results can refer to Fig. S3.

In addition, we investigated how the main components of MTGL-ADMET contribute to the prediction by ablation studies. We designed two variants of MTGL-ADMET (Fig. S4). The first one (denoted as w/o Att) eliminates the parallel attention block in the task-specific molecular embedding module. The second one (denoted as w/o Gate) lacks gate networks in the primary task-centered gate module. MTGL-ADMET significantly outperforms these variants in both classification and regression tasks. Specifically, compared to w/o Att and w/o Gate, MTGL-ADMET improves on average value by 3.1% and 1.1%. In detail, MTGL-ADMET improves the AUC value by 2.45% and 1.16% in classification tasks and the  $R^2$  value by 4.9% and 1.03% in regression tasks. Therefore, the results indicate that the parallel attention blocks and the gate networks play critical roles in predicting ADMET endpoints.

### 3.4 Case Study: Interpretability of MTGL-ADMET

Although deep learning is known as a black-box model, it is essential to understand how MTGL-ADMET makes a prediction and whether it can guide lead compound optimization in drug discovery. Since the task-specific molecular embedding module can learn task-specific atom importance by its task-specific attention layers, we decided that two bonding atoms are regarded as a crucial structure fragment if both of them have high attention weights. The weight of a bond is the average of the weights of its constituent atoms (highlighted in Fig. 5). One or more fragments form a crucial substructure, which is specific to endpoint tasks.

We selected seven endpoints as the case study, including HIA (A), OB (A), BBB (D), CYP3A4 and CYP2D6 inhibitors (M), Clearance (E), Hepatotoxicity (T), and Cardiotoxicity-1 (T), where A is for an absorption endpoint, D is for a distribution endpoint, M is for a metabolism endpoint, E is for an excretion endpoint and T is for a toxicity endpoint respectively.



**Fig. 5.** Cases study of crucial substructures. (a) two compounds of HIA and two of OB. (b) four compounds of BBB. (c) two CYP3A4 inhibitors and one inhibitor for CYP2D6 and CYP3A4. (d) four compounds of Clearance. (e) four compounds of Hepatotoxicity. (f) four compounds of Cardiotoxicity-1. The atoms and bonds of endpoint-specific critical substructures are highlighted in green. (Color figure online)

First, we picked up two *e* compounds (*Maprotiline* and *Acetohexamide*) having good HIAs and another two compounds (*Paroxetine cation* and *Zonisamide*) having good OBs. As shown in Fig. 5a, their crucial substructures indicated by our MTGL-ADMET involve *hydroxyl* and *amino*, of which all are commonly hydrophilic [34]. These results are consistent with the fact that drugs/compounds with good absorption properties (e.g., HIA and OB) have higher hydrophilicity [17].

Then, four compounds having good BBBs were investigated (Fig. 5b). Similarly, their highlighted substructures involve lipophilic functional groups, (i.e., *sulfonamide*, *morpholinyl* and *piperidine*), which are helpful to pass the blood-brain barrier [35–37].

After that, we investigated an important enzyme inhibitor, *Ethinylestradiol*, which belongs to the family of CYP3A4 inhibitors. Two compounds having high affinities with CYP enzymes were focused on. After an extra docking simulation (Autodock), we found that their highlighted substructures involve aromatic rings and hydrophobic fragments (Fig. 5c, upper panel), which are the key to the binding site in the pocket by contributing to non-covalent bonds (e.g., H-bonds and Pi-Pi bonds) [38]. Also, the result is consistent with the domain knowledge that CYP3A4 inhibitors usually have *furan ring*, *tertiary amine*, or *acetylene* substructures [38].



More importantly, to investigate whether a compound shows task-specific crucial substructures, we picked up a compound (*1-Allyl-1, 5-anhydro-2, 3-dideoxy-4-O-(4-fluoro-2-methylphenyl) hex-2-enitol*) inhibiting two kinds of CYP enzymes (CYP 2D6 and CYP 3A4) as the case study (Fig. 5c, lower panel). The results validate that its highlighted substructures are specific to two endpoints.

Furthermore, four compounds having good Clearances were selected (Fig. 5d). Their highlighted substructures, including *alkyl* and *halogen*, are lipophilic. The results are consistent with the knowledge that compounds having high lipophilicity tend to have high clearance [18,34]. Last, we paid attention to two toxicity endpoints, Hepatotoxicity, and Cardiotoxicity-1, which represent serious concerns in drug development and are the main reasons for a drug being withdrawn from the market [39]. *Acetohexamide* and *Amodiaquine* have highlighted substructures, *sulfonamide moiety* and *halogen atom* (Fig. 5e), while *Lidoflazine* and *Bromperidol* have highlighted substructures, a basic nitrogen center flanked by aromatic or hydrophobic groups (Fig. 5f). These two groups of substructures agree with the finding in [40] and [41] respectively.

In summary, the consistency of our findings with domain knowledge and literature demonstrates that MTGL-ADMET is an interpretable model, which can indicate compound substructures (or functional groups) significantly associated with ADMET endpoints. It would help reveal why a compound shows a specific ADMET property of interest.

## 4 Conclusions

In this paper, holding a new paradigm of MTL, ‘one primary, multiple auxiliaries’, we’ve proposed a multi-task graph learning framework for predicting various ADMET endpoints of drug-like small molecules (MTGL-ADMET). It contains two stages, auxiliary task selection, and primary-centered multi-task learning. The former stage builds a task association network by training individual and pairwise tasks and leverages both the status theory and the maximum flow in complex network science to collect appropriate tasks. The latter stage constructs a novel primary-centered multi-task graph learning model to train the primary task and its auxiliary tasks together. The model technically includes a task-shared atom embedding module, a task-specific molecular embedding module, a primary task-centered gating module, and a multi-task predictor. MTGL-ADMET can address two existing issues, including auxiliary selection and task-specific molecular substructure finding.

The comparison with state-of-the-art MTL-based models demonstrates the superiority of our MTGL-ADMET in terms of prediction performance. More elaborate experiments validate its contributions. First, it improves the selection algorithm of appropriate auxiliary tasks in the MTL by calculating the maximum flux of status theory-satisfied task triads as the initial estimator. Secondly, by the gating networks, it uncovers the contributions of auxiliary tasks to the primary task, which helps understand ADMET endpoint associations in a quantity manner. Thirdly, by task-shared atom embeddings and task-specific attention

scores, it obtains task-specific molecular embeddings with the highlight of crucial compound substructures specific to ADMET endpoints.

In summary, we believe that our study provides new insights into ADMET endpoint prediction and also can be borrowed for other multi-task learning problems (e.g., compound physic-chemical property prediction in drug discovery, object detection, autonomous vehicles, and recommendation systems). In the coming future, the integration of status theory and maximum flow techniques into the architecture of neural networks (e.g., to embed the task association network) would improve the finding of optimal auxiliary tasks.

**Acknowledgements.** The authors would like to thank anonymous reviewers for suggestions that improved the paper. This work is supported in part by the National Nature Science Foundation of China (Grant number 61872297), Shaanxi Province Key R&D Program (Grant number 2023-YBSF-114), and CAAI-Huawei MindSpore Open Fund (Grant Number CAAIXSJLJJ-2022-035A).

## References

1. Schneider, G.: Automating drug discovery. *Nat. Rev. Drug Discov.* **17**(2), 97–113 (2018)
2. Wouters, O.J., McKee, M., Luyten, J.: Estimated research and development investment needed to bring a new medicine to market, 2009–2018. *JAMA* **323**(9), 844–853 (2020)
3. Waring, M.J., Arrowsmith, J., Leach, A.R., et al.: An analysis of the attrition of drug candidates from four major pharmaceutical companies. *Nat. Rev. Drug Discov.* **14**(7), 475–486 (2015)
4. Schneider, P., Walters, W.P., Plowright, A.T., et al.: Rethinking drug design in the artificial intelligence era. *Nat. Rev. Drug Discov.* **19**(5), 353–364 (2020)
5. Jia, C.-Y., Li, J.-Y., Hao, G.-F., Yang, G.-F.: A drug-likeness toolbox facilitates ADMET study in drug discovery. *Drug Discov. Today* **25**(1), 248–258 (2020)
6. Wu, Z., Ramsundar, B., Feinberg, E.N., et al.: MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**(2), 513–530 (2018)
7. Shen, W.X., Zeng, X., Zhu, F., et al.: Out-of-the-box deep learning prediction of pharmaceutical properties by broadly learned knowledge-based molecular representations. *Nat. Mach. Intell.* **3**(4), 334–343 (2021)
8. Rong, Y., Bian, Y., Xu, T., et al.: Self-supervised graph transformer on large-scale molecular data. In: 33th Advances in Neural Information Processing Systems, pp. 12559–12571 (2020)
9. Chen, D., Gao, K., Nguyen, D.D., et al.: Algebraic graph-assisted bidirectional transformers for molecular property prediction. *Nat. Commun.* **12**(1), 3521 (2021)
10. Ruder, S.: An overview of multi-task learning in deep neural networks. [arXiv:1706.05098](https://arxiv.org/abs/1706.05098) (2017)
11. Bhatarai, B., Walters, W.P., Hop, C.E., et al.: Opportunities and challenges using artificial intelligence in ADME/Tox. *Nat. Mater.* **18**(5), 418–422 (2019)
12. Feinberg, E.N., Joshi, E., Pande, V.S., Cheng, A.C.: Improvement in ADMET prediction with multitask deep featurization. *J. Med. Chem.* **63**(16), 8835–8848 (2020)

13. Xiong, G., Wu, Z., Yi, J., et al.: ADMETlab 2.0: an integrated online platform for accurate and comprehensive predictions of ADMET properties. *Nucleic Acids Res.* **49**(W1), W5–W14 (2021)
14. Peng, Y., Lin, Y., Jing, X.Y., et al.: Enhanced graph isomorphism network for molecular ADMET properties prediction. *IEEE Access* **8**, 168344–168360 (2020)
15. Fifty, C., Amid, E., Zhao, Z., et al.: Efficiently identifying task groupings for multi-task learning. In: 34th Advances in Neural Information Processing Systems, pp. 27503–27516 (2021)
16. Dong, J., Li, S., Liu, G.: Binimetinib is a potent reversible and time-dependent inhibitor of cytochrome P450 1A2. *Chem. Res. Toxicol.* **34**(4), 1169–1174 (2021)
17. Khadka, P., Ro, J., Kim, H., et al.: Pharmaceutical particle technologies: an approach to improve drug solubility, dissolution and bioavailability. *Asian J. Pharm. Sci.* **9**(6), 304–316 (2014)
18. Johnson, T.W., Gallego, R.A., Edwards, M.P.: Lipophilic efficiency as an important metric in drug design. *J. Med. Chem.* **61**(15), 6401–6420 (2018)
19. Tang, J., Chang, Y., Aggarwal, C., et al.: A survey of signed network mining in social media. *ACM Comput. Surv. (CSUR)* **49**(3), 1–37 (2016)
20. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. *Can. J. Math.* **8**, 399–404 (1956)
21. Nguyen, T., Le, H., Quinn, T.P., et al.: GraphDTA: predicting drug-target binding affinity with graph neural networks. *Bioinformatics* **37**(8), 1140–1147 (2021)
22. Kipf, TN., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
23. Tang, H., Liu, J., Zhao, M., et al.: Progressive layered extraction (PLE): a novel multi-task learning (MTL) model for personalized recommendations. In: 14th ACM Conference on Recommender Systems, pp. 269–278 (2020)
24. Wu, Z., Jiang, D., Wang, J., et al.: Mining toxicity information from large amounts of toxicity data. *J. Med. Chem.* **64**(10), 6924–6936 (2021)
25. Yang, M., Chen, J., Xu, L., et al.: A novel adaptive ensemble classification framework for ADME prediction. *RSC Adv.* **8**(21), 11661–11683 (2018)
26. Wang, X., Liu, M., Zhang, L., et al.: Optimizing pharmacokinetic property prediction based on integrated datasets and a deep learning approach. *J. Chem. Inf. Model.* **60**(10), 4603–4613 (2020)
27. Wang, N.-N., Deng, Z.-K., Huang, C., et al.: ADME properties evaluation in drug discovery: prediction of plasma protein binding using NSGA-II combining PLS and consensus modeling. *Chemom. Intell. Lab. Syst.* **170**, 84–95 (2017)
28. Alsenan, S., Al-Turaiki, I., Hafez, A.: A deep learning approach to predict blood-brain barrier permeability. *PeerJ Comput. Sci.* **7**, e515 (2021)
29. Lombardo, F., Berellini, G., Obach, R.S.: Trend analysis of a database of intravenous pharmacokinetic parameters in humans for 1352 drug compounds. *Drug Metab. Dispos.* **46**(11), 1466 (2018)
30. Wang, J.-B., Cao, D.-S., Zhu, M.-F., et al.: In silico evaluation of logD<sub>7.4</sub> and comparison with other prediction methods. *J. Chemometr.* **29**(7), 389–398 (2015)
31. Delaney, J.S.: ESOL: estimating aqueous solubility directly from molecular structure. *J. Chem. Inf. Comput. Sci.* **44**(3), 1000–1005 (2004)
32. Montanari, F., Kuhnke, L., Ter Laak, A., et al.: Modeling physico-chemical ADMET endpoints with multitask graph convolutional networks. *Molecules* **25**(1), 44 (2019)

33. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-93417-4\\_38](https://doi.org/10.1007/978-3-319-93417-4_38)
34. Li, Y., Hsieh, C.-Y., Lu, R., et al.: An adaptive graph learning method for automated molecular interactions and properties predictions. *Nat. Mach. Intell.* **4**(7), 645–651 (2022)
35. Zhu, C., Li, X., Zhao, B., et al.: Discovery of aryl-piperidine derivatives as potential antipsychotic agents using molecular hybridization strategy. *Eur. J. Med. Chem.* **193**, 112214 (2020)
36. Lenci, E., Calugi, L., Trabocchi, A.: Occurrence of morpholine in central nervous system drug discovery. *ACS Chem. Neurosci.* **12**(3), 378–390 (2021)
37. Khaldan, A., Bouamrane, S., En-Nahli, F., et al.: Prediction of potential inhibitors of SARS-CoV-2 using 3D-QSAR, molecular docking modeling and ADMET properties. *Heliyon* **7**(3), e06603 (2021)
38. Beck, T.C., Beck, K.R., Morningstar, J., et al.: Descriptors of cytochrome inhibitors and useful machine learning based methods for the design of safer drugs. *Pharmaceuticals* **14**(5), 472 (2021)
39. Onakpoya, I.J., Heneghan, C.J., Aronson, J.K.: Post-marketing withdrawal of 462 medicinal products because of adverse drug reactions: a systematic review of the world literature. *BMC Med.* **14**(1), 10 (2016)
40. Liu, R., Yu, X., Wallqvist, A.: Data-driven identification of structural alerts for mitigating the risk of drug-induced human liver injuries. *J. Cheminformatics* **7**(1), 1–8 (2015). <https://doi.org/10.1186/s13321-015-0053-y>
41. Cavalluzzi, M.M., Imbrici, P., Gualdani, R., et al.: Human ether-à-go-go-related potassium channel: exploring SAR to improve drug design. *Drug Discov. Today* **25**(2), 344–366 (2020)



# CDGCN: Conditional de novo Drug Generative Model Using Graph Convolution Networks

Shikha Mallick<sup>1</sup>(✉)  and Sahely Bhadra<sup>2</sup> 

<sup>1</sup> Department of Computer Science and Engineering, Indian Institute of Technology, Palakkad, Palakkad 678557, Kerala, India

[its.shikhamallick@gmail.com](mailto:its.shikhamallick@gmail.com)

<sup>2</sup> Department of Data Science, Indian Institute of Technology, Palakkad, Palakkad 678557, Kerala, India

[sahely@iitpkd.ac.in](mailto:sahely@iitpkd.ac.in)

**Abstract.** *De novo* drug design is a crucial part of drug discovery which is a highly expensive and slow process. Many deep learning methods have been proposed to automate and accelerate it. However, most of the current state-of-the-art methods are limited to generating novel drugs specific to proteins that already have known drugs or limited to generating molecules which lack certain desirable drug-like properties like high binding affinity or low binding energy. We introduce our graph generative model, CDGCN (Conditional *de novo* drug generative model using Graph Convolution Networks), for *de novo* drug generation for novel proteins, which takes as input a protein sequence of amino acids and generates novel molecular structures having desirable drug-like properties. CDGCN generates desirable molecules for a protein using a sequential decoding scheme by learning the distribution of generation paths of its ligands. We show that CDGCN can quickly generate novel and chemically valid drug-like molecules which have a higher binding affinity with their target proteins as compared to the state-of-the-art methods. The best binding energy between a novel protein and its novel drug-like molecules generated by CDGCN was observed to be at least  $-7.3$  kcal/mol whereas for the state-of-the-art method it was observed to be  $-6.2$  kcal/mol.

**Availability and implementation:** Code and data are available at <https://github.com/mshik/CDGCN>.

**Keywords:** Graph convolution network · De novo drug design · Graph generation

## 1 Introduction

Drug discovery and development is an integral part of the healthcare system, with the objective to find better drugs for known diseases and effective drugs for new diseases. Usually behind a disease there is a target *protein* which is a type

---

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_7](https://doi.org/10.1007/978-3-031-29119-7_7).

of biological molecule and behind a drug there is a *lead* which is a biological molecule with drug-like properties [8,10,17,39] like high binding affinity with the target protein to suppress its function [10], low synthetic accessibility score [8], and others [17,39]. Finding optimum leads turns out to be the main part of drug discovery and is mainly done in two ways: i) Drug repurposing which is a process of finding leads from the set of known drugs which inhibit already known proteins similar to the target protein, and ii) *de novo* drug design which aims to discover novel drugs for known or novel target proteins. Drug repurposing requires detailed knowledge about the target proteins to find similar known proteins [35]. Moreover, it restricts the search space among known drugs which can hinder finding leads with better binding affinity. From now on, we mention proteins known to interact with some known drugs as known proteins, and proteins not known to interact with any known drug as novel proteins.

In this work, we focus on *de novo* drug design which is an iterative task and has multiple stages starting from i) understanding the target protein, ii) generating and optimizing leads having drug-like properties, iii) lab trial to obtain drugs from leads etc. The latter is a highly expensive and slow process and its success depends on the quality of leads found in stage ii. Efficiently finding optimum leads is not an easy task as one needs to search quite extensively among possible molecules which have drug-like properties and high binding affinity and the size of the drug-like molecule space is estimated to be  $10^{60}$ , and the number of synthesizable molecules is in the order of  $10^8$  [10,33].

In recent years, machine learning methods have been introduced for *de novo* drug design, many of which utilize deep generative models [10,14–16,22]. The primary idea behind these methods is to generate leads with the required properties of valid drugs. The set of generated leads are significantly small to be considered for further testing which reduces the search time. These generative models learn the distribution of molecules and then generate novel molecules by sampling from the learned distribution [40]. Popular deep generative models for *de novo* drug generation can be broadly divided into two categories: i) SMILES-based methods [10] which generate SMILES strings [44] of molecules and ii) graph-based methods [14–16,22] which generate graph structures of molecules. There are existing methods that generate novel leads for known proteins by learning to generate molecules similar to known drugs for those proteins. The conditional graph generative model by Li et al. [22] is designed to generate novel drugs for a predetermined set of known proteins on which it is trained. The sequential decoding scheme of this model is well suited for drug molecules of different sizes as it provides the flexibility to generate graphs of different sizes unlike the methods which generate graphs by generating fixed size adjacency matrices in one step [23]. However, this model is incapable of learning the rich structural and functional information of proteins, and therefore, fails to generate novel drugs for novel proteins. The recent SMILES-based generative model by Grechishnikova [10] addresses this issue by learning to translate protein sequence of amino acids to SMILES string of drugs. To the best of our knowledge, it is the only method that is designed to generate novel drugs for novel proteins. However, SMILES-based methods are limited to learning the SMILES syntax and

grammar, rather than directly learning the molecular structure, and hence they tend to generate less number of chemically valid molecules [22]. It is also shown that Grechishnikova [10] is comparatively much slower and requires a larger memory to generate the same number of drugs as compared to Li et al. [22].

In this work, we propose a conditional graph generative model to generate novel drug structures for novel proteins which satisfy drug-like properties and strongly bind to proteins. Our proposed model CDGCN (Conditional *de novo* drug generative model using Graph Convolution Networks) learns the relationship between protein sequences, and the structure of their known drugs, and then predicts the structure of novel drugs for novel proteins exploiting that learned relationship. CDGCN utilizes a sequential decoding scheme for generating desirable molecular graphs for a given protein by learning the distribution of generation paths for ligands of that protein. Novel proteins are more widely available as sequences of amino acids than their three-dimensional structures [10] and hence it is ideal to utilize protein sequences for training the model. The three-dimensional structures of some of the proteins for testing is obtained from Protein Data Bank (PDB) [1]. We show that CDGCN generates a smaller number of prescribed leads and while reducing search space it also prescribes completely unknown or novel leads which have better binding energy than known ligands of the tested proteins, namely, i) Nitric oxide synthase, inducible (PDB code 4NOS), ii) Interleukin-1 receptor-associated kinase 4 (PDB code 2NRU), iii) Tyrosine-protein phosphatase non-receptor type 1 (PDB code 1LQF), iv) Nitric oxide synthase, brain (PDB code 1K2R), and v) Integrin alpha-L (PDB code 1CQP). The contribution of this work is as follows:

- We developed the first graph generative network, CDGCN (Conditional *de novo* drug generative model using Graph Convolution Networks), for protein-specific *de novo* drug generation for both known and novel proteins.
- As technical contribution, we propose a non-trivial way to include protein function information into the graph generation process, which allowed us to generate novel drugs having high binding affinity to novel proteins.
- Our proposed method CDGCN achieved superior performance when compared to the state-of-the-art method by Grechishnikova [10] for the task of novel protein-specific *de novo* drug generation for five novel proteins such that the best binding energy between each protein and its generated molecules by CDGCN is at least  $-7.3$  kcal/mol whereas the same for the generated molecules by Grechishnikova [10] is  $-6.2$  kcal/mol.

To the best of our knowledge, there is no existing method that can directly generate a reasonable number of drug-like molecules using the information from novel proteins in a short amount of time. This is crucial to reduce the time and cost required in drug discovery for novel proteins.

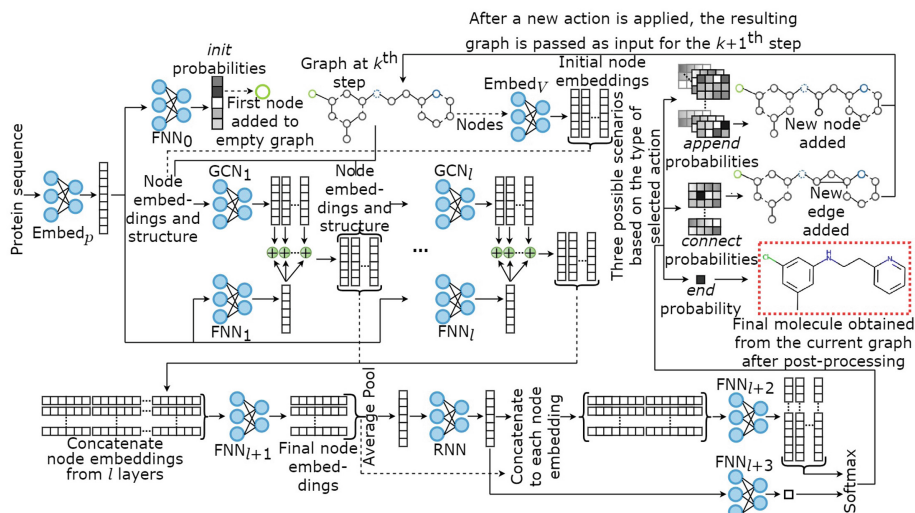
## 2 Materials and Methods

### 2.1 Molecular Graph Generation

A drug molecule is represented as a graph  $G = (V, E)$  where  $V$  is the set of nodes in  $G$  corresponding to the atoms in the drug molecule, and  $E$  is the set



of edges in  $G$  corresponding to the bonds in the drug molecule. Each node has an atom type from a fixed set  $\mathcal{A}$  of atom types and each edge has a bond type from a fixed set  $\mathcal{B}$  of bond types. CDGCN takes as input a protein as a sequence of amino acids and sequentially generates a molecular graph by adding a node or an edge incrementally. The graph generation process starts with an empty graph. At each generation step, CDGCN selects one out of four different types of actions to be performed on the intermediate graph. i) *init* action which adds the first node to the empty graph, ii) *append* action which attaches a new node with a new edge to an existing node in the intermediate graph, iii) *connect* action which adds a new edge between an existing node and the last appended node in the intermediate graph, and iv) *end* action which ends the sequential graph generation process. Figure 1 shows an overview of CDGCN. In CDGCN,  $\text{Embed}_p$  takes a protein sequence and gives its embedding which contains useful information about its function.  $\text{FNN}_0$  takes this protein embedding and gives probability scores for each atom type in  $\mathcal{A}$  for the *init* action.  $\text{FNN}_1, \dots, \text{FNN}_l$  also take the protein embedding and give task-specific protein embeddings. At each generation step,  $\text{Embed}_v$  takes the atom type of each node in the intermediate graph and gives its embedding.  $\text{GCN}_1$  takes the intermediate graph structure and the node embeddings and gives new node embeddings. The task-specific protein embedding is summed with the output node embedding for each  $\text{FNN}$  and  $\text{GCN}$  layer.  $\text{GCN}_2, \dots, \text{GCN}_l$  take the intermediate graph structure and the summed node embeddings from the previous layer. The outputs of the  $l$  different summations are concatenated and passed to  $\text{FNN}_{l+1}$  to obtain final node embeddings and then passed to an average pooling layer to obtain a graph-level embedding. This



**Fig. 1.** Overview of our generative model CDGCN. The three types of actions, i.e., *append*, *connect* and *end*, are shown separately for better visualization.



graph embedding is passed to RNN, which stores information from the previous intermediate graphs, and its output is concatenated with each node embedding from  $\text{FNN}_{l+1}$ . The concatenated outputs are passed to  $\text{FNN}_{l+2}$  to obtain a tensor of size  $|V| \times (|\mathcal{A}| \times |\mathcal{B}| + |\mathcal{B}|)$  for all possible *append* and *connect* actions. RNN output is passed to  $\text{FNN}_{l+3}$  to obtain a scalar for the *end* action.  $\text{FNN}_{l+2}$  and  $\text{FNN}_{l+3}$  outputs are passed through Softmax layer to obtain the final probability scores for the *append*, *connect* and *end* actions. An action is then sampled to be applied on the intermediate graph. After *append* or *connect* action, the new graph is given as input to  $\text{GCN}_1$  for the next generation step. After *end* action, the generation process stops and the current graph is post-processed to obtain the drug molecule. The inference algorithm of CDGCN is given in the supplementary.

## 2.2 Model Architecture

The learnable parameters of the model are denoted as  $\Theta$ .  $\text{Embed}_p$  is based on a pre-trained network from Dallago et al. [5] which was trained on protein sequences for the task of predicting protein function from its sequence.  $\text{FNN}_0$  is a fully connected feed-forward neural network which consists of i) dense layer, and ii) Softmax activation [7]. CDGCN is first pre-trained to generate chemically valid molecules without protein constraint. In this case, a learnable weight vector followed by Softmax activation is used instead of  $\text{Embed}_p$  and  $\text{FNN}_0$ .  $\text{Embed}_v$  is a matrix of dimension  $|\mathcal{A}| \times d_0$  consisting of learnable embeddings.  $\text{GCN}_1, \dots, \text{GCN}_l$  are Graph Convolution Networks (GCN) having i) Graph Convolution (GC) layer, ii) Batch Normalization (BN) layer [11, 13], and iii) Rectified Linear Unit (ReLU) activation [30], except  $\text{GCN}_l$  which consists of only GC layer. GC is computed following Wu et al. [46] for molecular graphs and has the following formulation.

$$\mathbf{h}_{(v)\text{GC}}^i = \mathbf{W}^i \mathbf{h}_{(v)\text{GC}}^{i-1} + \sum_{b \in \mathcal{B}} \Theta_b^i \sum_{u \in \mathcal{N}_b(v)} \mathbf{h}_{(u)\text{GC}}^{i-1} + \sum_{1 < d \leq D} \Theta_d^i \sum_{u \in \mathcal{N}^d(v)} \mathbf{h}_{(u)\text{GC}}^{i-1} \quad (1)$$

where  $\mathbf{h}_{(v)\text{GC}}^i$  is the  $i$ th layer node embedding of node  $v$ .  $\mathcal{N}_b(v)$  is the set of nodes which are directly connected to node  $v$  with edges having bond type  $b$  and  $\mathcal{N}^d(v)$  is the set of nodes at a path length of  $d$  from the node  $v$  in the graph  $G$ .  $\sum_{b \in \mathcal{B}} \Theta_b^i \sum_{u \in \mathcal{N}_b(v)} \mathbf{h}_{(u)\text{GC}}^{i-1}$  and  $\sum_{1 < d \leq D} \Theta_d^i \sum_{u \in \mathcal{N}^d(v)} \mathbf{h}_{(u)\text{GC}}^{i-1}$ , where  $D$  is the receptive field size, represent the information of the local and distant neighbourhoods of node  $v$  respectively.  $\mathbf{W}^i$ ,  $\Theta_b^i$  and  $\Theta_d^i$  are learnable parameters of the  $i$ th GC layer.  $\text{FNN}_1, \dots, \text{FNN}_l$  each consist of a single dense layer with output dimensions equal to those of the  $L$  GCN layers respectively.  $\text{FNN}_{l+1}$  consists of two dense layers each followed by a BN layer and ReLU activation. RNN is implemented using Gated Recurrent Unit (GRU) layers [4] as shown in the following equation.

$$\mathbf{h}_m^{\text{GRU}} = \text{GRU}_{\Theta}(\mathbf{h}_{m-1}^{\text{GRU}}, \mathbf{h}_{v^*} || \mathbf{h}_G) \quad (2)$$

where  $\mathbf{h}_m^{\text{GRU}}$  is the graph-level embedding for the  $m$ th generation step containing information from the previous  $m - 1$  intermediate graphs.  $\mathbf{h}_{v^*}$  is the node

embedding of the last appended node  $v^*$  in the current intermediate graph  $G$ .  $\text{FNN}_{l+2}$  consists of i) dense layer, ii) BN layer, iii) ReLU activation, iv) dense layer and v) exponential activation.  $\text{FNN}_{l+3}$  consisting of i) dense layer, and ii) exponential activation.  $\parallel$  is concatenation.

### 2.3 Loss Computation and Training

The objective for CDGCN is to maximize the log-likelihood of generating a molecular graph  $G$  by following a generation path  $\mathcal{T}$  given a protein  $P$ .  $\mathcal{T}$  is a sequence of  $m$  actions where the  $j$ th action  $t_j$  determines the evolution of the intermediate graph  $G_j$  at the  $j$ th generation step. The next action depends on all the previous intermediate graphs and the protein  $P$ . The log-likelihood is given as

$$\log(\mathcal{P}_\Theta(G, \mathcal{T}|P)) = \sum_{j=1}^m \log(\mathcal{P}_\Theta(t_{j+1}|G_j, \dots, G_1, P)) \quad (3)$$

where  $m$  is the number of actions in  $\mathcal{T}$  and  $G_j$  is the intermediate graph obtained after applying action  $t_j \in \mathcal{T}$ . The marginal likelihood is given as

$$\log(\mathcal{P}_\Theta(G|P)) = \log \sum_{\mathcal{T} \in S(G)} \mathcal{P}_\Theta(G, \mathcal{T}|P) \quad (4)$$

where  $S(G)$  is the set of all possible generation paths. However, for most molecules encountered during drug discovery, the marginal likelihood is intractable [22]. To resolve this issue, importance sampling is used to estimate a particular probability distribution by sampling from a different predefined distribution [6, 18], and hence the marginal likelihood from Eq. 4 becomes

$$\log(\mathcal{P}_\Theta(G|P)) = \log \mathbb{E}_{\mathcal{T} \sim \mathcal{Q}_\alpha(\mathcal{T}|G, P)} \left[ \frac{\mathcal{P}_\Theta(G, \mathcal{T}|P)}{\mathcal{Q}_\alpha(\mathcal{T}|G, P)} \right] \geq \log \frac{1}{K} \sum_{k=1}^K \frac{\mathcal{P}_\Theta(G, \mathcal{T}_k|P)}{\mathcal{Q}_\alpha(\mathcal{T}_k|G, P)} \quad (5)$$

where  $\mathcal{Q}_\alpha(\mathcal{T}|G, P)$  is a predetermined distribution of generation paths in which the actions at each step are either based on canonical atom ordering [45] with probability  $\alpha$  or based on random ordering with probability  $1 - \alpha$  and  $K$  denotes the number of different generation paths possible to generate  $G$ . The detailed algorithm to compute  $\mathcal{Q}_\alpha(\mathcal{T}|G, P)$  is given in the supplementary. The training of CDGCN is done in mini-batches of size  $N$  and during the training, the following negative log-likelihood loss is being minimized

$$\hat{\mathcal{L}}(\Theta) = -\frac{1}{N} \sum_{i=1}^N \log \frac{1}{K} \sum_{k=1}^K \frac{\mathcal{P}_\Theta(G_i, \mathcal{T}_{i_k}|P_i)}{\mathcal{Q}_\alpha(\mathcal{T}_{i_k}|G_i, P_i)} \quad (6)$$

## 3 Results and Discussion

This section reports the empirical evaluation and comparison of CDGCN with state-of-the-art benchmark methods for the quality of the generated structures

for both the cases of *known* and *novel* proteins, i.e., i) the fraction of generated structures being chemically valid and novel [22], ii) the fraction of generated structures having drug-like properties [10], iii) the fraction of correctly generated structures for known proteins [22] and iv) the best binding affinity achieved by the generated structures [10]. Quantitative efficiency of CDGCN is shown by comparing the time required for generating the structures.

### 3.1 Datasets

Two real biological databases, i) ChEMBL [28] and ii) BindingDB [27] were used for empirical evaluation. ChEMBL is a manually curated public database of approximately 2.3 million molecules. Around  $10^6$  molecules that satisfy desired physicochemical properties (Sect. 3.4) were extracted and used for the pre-training of CDGCN and Li et al. [22]. BindingDB is a public database of experimentally measured binding energies between proteins and ligands [27]. All experiments here use the version of the BindingDB database constructed by Grechishnikova [10] where each protein is presented as FASTA sequences [26] and each ligand is presented as SMILES strings [44]. This was used for the fine-tuning of the pre-trained CDGCN and Li et al. [22], as well as for the training of Grechishnikova [10]. The BindingDB dataset is divided into five cross-validation folds where the validation folds are further split in half to obtain validation data for hyperparameter tuning and separate test data for evaluation of the trained models. The pairwise sequence similarity, measured using the Needleman-Wunsch global alignment algorithm from the EMBOSS package [36], between the proteins in the training dataset and the proteins in the test/validation dataset for each fold is kept less than 20% so that the test dataset has proteins which are truly novel for the generative model to prove its generalization. The majority of protein sequences share less than 40% similarity within the training dataset and the test dataset to make them diverse enough to train and evaluate the models. The histogram plots for the pairwise sequence similarities are given in the supplementary. Proteins in training data are known proteins whereas proteins in validation/test data are novel proteins. The models were trained, validated and tested by the ligands present in the BindingDB dataset, hence they were removed from the modified ChEMBL dataset to avoid data leakage during evaluation. In the raw dataset, each drug molecule is in its canonical SMILES format [45] which is used for obtaining the node descriptors and edge descriptors using RDKit [20]. The entire dataset, including both the modified ChEMBL and BindingDB together, has 65 unique atom types listed in the supplementary and four unique bond types (single, double, triple, and aromatic).

### 3.2 Baselines

For the task of generating novel drugs for known proteins, CDGCN was compared against the state-of-the-art i) graph-based method by Li et al. [22], and ii) SMILES-based method by Grechishnikova [10]. For the task of generating novel drugs for novel proteins, CDGCN has been compared with the latter.

### 3.3 Implementation Details

CDGCN is implemented in MXNet 1.7.0 [3] on a system with a single NVIDIA GeForce RTX 2080 Ti GPU with 8 GB memory. Training was done with batch size 8 selected using five-fold cross-validation. Batch sizes higher than 8 could not be tested due to memory constraint. It is however, worthwhile to train with higher batch sizes. All other hyperparameters and optimizer for CDGCN follow Li et al. [22]. Pre-training of CDGCN was done on the ChEMBL dataset. Finetuning of CDGCN was done on the BindingDB datasets by passing pairs of proteins and their known ligands to the model. The hyperparameters  $\alpha$  and  $K$  from Eq. 5 are selected as 0.8 and 5 respectively. A suitable choice for  $\alpha$  and  $K$  is crucial to generate unique and novel chemically valid molecules. The hyperparameter sensitivity study is in the supplementary. Finetuning lasted on average for 10 epochs with early stopping [34] for the five different folds. Li et al. [22] and Grechishnikova [10] were trained following their original setup, however, the batch size was restricted to 8 for Li et al. [22] and the number of epochs was restricted to 6000 for Grechishnikova [10] due to memory constraint.

### 3.4 Evaluation Metrics

CDGCN is designed to speed up the drug discovery process, hence we evaluate its ability to quickly generate high quality molecules. We report the time required to generate proposed structures by each of the three models, i.e., CDGCN and the two baselines, along with the chemical validity of generated molecules, which is checked using RDKit [20], and novelty of generated molecules using exact string matching algorithm on the molecules in the dataset.

Along with having a good binding affinity with the target protein, a lead should also have desirable physicochemical properties [10]. Lipinski’s rule of five is a widely used rule of thumb to test whether the generated molecules have certain physicochemical properties found in orally active drugs for humans [21, 25, 32], like i) an octanol-water partition coefficient  $\log P$  that does not exceed 5, ii) molecular weight within 500 daltons, iii) no more than 5 hydrogen bond donors (H-bond donors), and iv) no more than 10 hydrogen bond acceptors (H-bond acceptors) [9, 24]. Molecules satisfying properties like i) no more than 10 rotatable bonds and ii) no more than  $140 \text{ \AA}^2$  of topological polar surface area (TPSA) are predicted to have good oral bioavailability [41]. The quantitative estimate of drug-likeness (QED) is another property widely used to select appropriate molecules during the early stages of drug discovery [2]. It ranges from 0 to 1 with higher values indicating highly drug-like. The synthetic accessibility score (SAS) is crucial in deciding whether a drug-like molecule is easy to synthesize [8]. We report these properties for all three models.

Following Li et al. [22], we report the percentage of “binding/active” predictions ( $R_c$ ) among all generated molecules and their proteins using binary random forest classifiers. The positive class for these classifiers consisted of 100 known

ligands for each known protein. The negative class consisted of 100 ligands not known to interact with that known protein, however, it might be noisy. Higher  $R_c$  is better. ECFP6 fingerprints were used as the features for the ligands [37]. We selected classifiers which had accuracy of atleast 95% for evaluation.

Leads must strongly bind to a protein to inhibit its function. We performed molecular docking to evaluate binding energies of generated molecules. We generated three-dimensional conformers for both the generated molecules and the known ligands using OpenBabel package [31]. Molecular docking is computationally heavy, therefore we selected five novel proteins with available binding sites and docked their Protein Data Bank (PDB) [1] structures with the generated molecules and their known ligands using SMINA software with default settings [19]. Jupyter Dock software [29] was used for integrating SMINA into python for parallelizing the heavy computation of the molecular docking for multiple molecules. We report binding energies for the generated molecules and the known ligands for each of the five proteins.

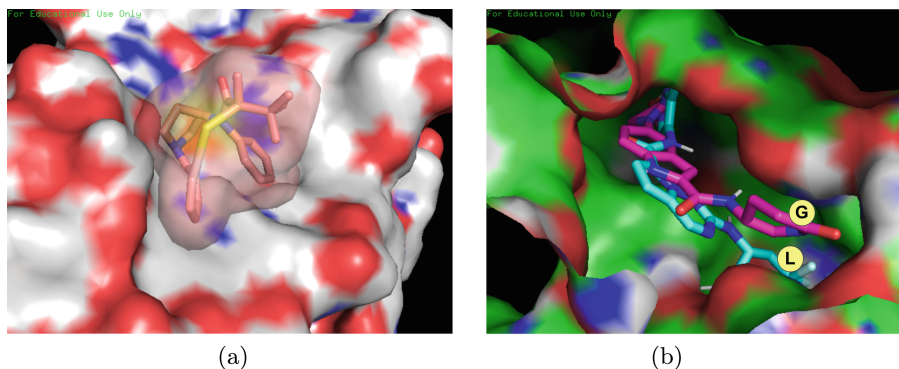
### 3.5 Experimental Evaluation

Different sets of molecules were generated by CDGCN and Li et al. [22] for each protein by repeated sampling as the generative processes here are probabilistic whereas a deterministic set of molecules was generated for each protein in the case of Grechishnikova [10]. Duplicate molecules were discarded from the sets of generated molecules. Henceforth  $S_N$  will denote the set of  $N$  generated molecules. The generation of  $S_N$  by Grechishnikova [10] was done by setting the beam size hyperparameter equal to  $N$ . To maintain consistency for evaluation with Grechishnikova [10], the molecular structures generated by both the graph-based methods were converted to canonical SMILES strings using RDKit [20]. Each experiment was repeated 10 times and the mean and standard deviations are reported. Paired t-tests for each evaluation metric were performed for statistical significance. The values in bold for multiple models indicate that the difference is not significant between those models.

Table 1 reports the i) average time in seconds required to generate the proposed structures, ii) average percentage of generated molecules that are chemically valid, novel and satisfy the constraints of all the physicochemical properties, and iii) average  $R_c$  values described in this section. Molecules generated by both the graph-based methods outperform those of Grechishnikova [10] in terms of their chemical validity, novelty and desirable drug-like properties by a large margin. This makes CDGCN computationally efficient for generating novel drugs for novel proteins which can help enlarge their sets of drugs. In the case of known proteins, Li et al. [22] is better than CDGCN for some cases but the values of CDGCN are comparable. It is important to note that more than 85% of the molecules generated by CDGCN for both the known and the novel proteins satisfy SAS which means that CDGCN tends to generate molecules that are easy to synthesize which can further help in speeding up the drug

discovery process. For the  $R_c$  values for known proteins, CDGCN is superior to Li et al. [22] but Grechishnikova [10] is superior to both of them. This is because for known proteins, Grechishnikova [10] with simple SMILES strings is easy to learn but can lead to over-fitting which reflects in Table 1 and the percentage of novel molecules generated by Grechishnikova [10] is very less as compared to the graph-based methods.

Table 2 shows the best binding energies and the percentage of generated molecules having binding energy lower than  $-7.0$  kcal/mol for each protein. Binding energy of  $-6.0$  kcal/mol is considered as minimum threshold for any molecule to be used for drug development [12]. In all the cases, CDGCN is superior to Grechishnikova [10]. For each of the five novel proteins, CDGCN generated molecules that gave lower binding energy than their known ligands. This shows that CDGCN is optimized to generate novel molecules which can effectively bind with a protein. Two sample molecular docking output images using PyMol [38] are provided in Fig. 2.



**Fig. 2.** (a) Docked novel protein from test dataset named Integrin alpha-L (PDB code 1CQP) having highest binding energy of  $-7.3$  kcal/mol with its generated novel molecule by CDGCN. (b) Docked protein named Vascular endothelial growth factor receptor 2 (PDB code 1Y6A) having lowest binding energy of  $-11.1$  kcal/mol with one of its known ligand (L) named ZD4190 [43] and  $-11.2$  kcal/mol with one of its generated novel molecule (G) by CDGCN.

The Tanimoto similarity score, following Grechishnikova [10], shows  $\sim 13\%$  of the generated molecules from all the three models-Li et al. [22], Grechishnikova [10] and CDGCN are structurally similar to the molecules in the dataset and  $\sim 42\%$  of the generated molecules from all the three models differ significantly from the molecules in the dataset. We also tested the relevance of  $\text{Embed}_p$  from Fig. 1 by computing binding energies of the five novel proteins from Table 2

**Table 1.** Qualitative and quantitative analysis of the three models-Li et al. [22], Grechishnikova [10], and CDGCN, for generating chemically valid, novel and drug-like molecules for the known and the novel proteins. Better values are in bold. ‘-’ indicates samples not generated due to memory constraint.

	Protein type	Model	Average across all proteins		
			$S_{10}$	$S_{100}$	$S_{1000}$
Time required to generate structures (seconds)	known	Li et al. [22]	<b>15.5 ± 0.77</b>	<b>28.8 ± 1.42</b>	<b>117.0 ± 4.63</b>
		Grechishnikova [10]	41.5 ± 3.30	121.4 ± 8.57	-
		CDGCN	17.8 ± 1.62	31.0 ± 4.05	131.1 ± 4.87
	novel	Grechishnikova [10]	42.1 ± 2.10	190.0 ± 12.21	-
		CDGCN	<b>20.7 ± 0.91</b>	<b>35.1 ± 1.47</b>	141.7 ± 5.72
Valid (%)	known	Li et al. [22]	<b>96.3 ± 1.55</b>	<b>95.2 ± 0.57</b>	<b>95.4 ± 0.21</b>
		Grechishnikova [10]	89.0 ± 4.34	77.3 ± 6.44	-
		CDGCN	95.3 ± 2.49	95.0 ± 0.66	94.9 ± 0.37
	novel	Grechishnikova [10]	85.7 ± 4.15	63.9 ± 2.98	-
		CDGCN	<b>94.5 ± 0.63</b>	<b>94.1 ± 0.86</b>	94.0 ± 0.51
Novel (%)	known	Li et al. [22]	<b>99.7 ± 0.22</b>	<b>99.7 ± 0.10</b>	<b>99.7 ± 0.03</b>
		Grechishnikova [10]	76.8 ± 0.20	78.0 ± 0.13	-
		CDGCN	<b>99.7 ± 0.27</b>	<b>99.7 ± 0.11</b>	99.6 ± 0.03
	novel	Grechishnikova [10]	82.3 ± 0.32	84.1 ± 0.09	-
		CDGCN	<b>99.8 ± 0.48</b>	<b>99.9 ± 0.08</b>	99.8 ± 0.05
$< 5 \log P$	known	Li et al. [22]	78.0 ± 4.79	76.9 ± 2.17	76.9 ± 1.65
		Grechishnikova [10]	63.9 ± 6.02	48.5 ± 9.50	-
		CDGCN	<b>78.8 ± 2.65</b>	<b>77.5 ± 3.69</b>	<b>77.1 ± 2.92</b>
	novel	Grechishnikova [10]	55.3 ± 9.88	31.1 ± 4.30	-
		CDGCN	<b>77.4 ± 4.22</b>	<b>76.9 ± 2.94</b>	77.0 ± 2.60
$< 500$ Molecular weight (Da)	known	Li et al. [22]	<b>79.0 ± 3.99</b>	<b>77.9 ± 2.16</b>	<b>77.9 ± 2.19</b>
		Grechishnikova [10]	60.4 ± 14.02	43.6 ± 8.33	-
		CDGCN	76.9 ± 4.31	75.8 ± 3.93	75.5 ± 2.76
	novel	Grechishnikova [10]	77.3 ± 5.10	42.1 ± 4.62	-
		CDGCN	<b>71.9 ± 4.35</b>	<b>70.9 ± 2.07</b>	71.0 ± 2.10
$< 5$ Number of H-bond donors	known	Li et al. [22]	<b>91.0 ± 2.92</b>	<b>88.4 ± 1.17</b>	<b>88.7 ± 0.68</b>
		Grechishnikova [10]	66.6 ± 9.77	50.6 ± 7.98	-
		CDGCN	88.9 ± 5.47	87.6 ± 1.67	87.5 ± 0.73
	novel	Grechishnikova [10]	76.5 ± 5.29	42.7 ± 4.25	-
		CDGCN	<b>85.0 ± 4.05</b>	<b>84.0 ± 1.60</b>	84.1 ± 1.56
$< 10$ Number of H-bond acceptors	known	Li et al. [22]	<b>93.0 ± 2.92</b>	<b>90.4 ± 1.14</b>	<b>90.7 ± 0.47</b>
		Grechishnikova [10]	79.8 ± 6.47	60.8 ± 8.74	-
		CDGCN	90.5 ± 4.36	89.3 ± 2.04	89.2 ± 1.89
	novel	Grechishnikova [10]	77.8 ± 5.23	43.8 ± 4.35	-
		CDGCN	<b>89.5 ± 2.69</b>	<b>88.1 ± 1.66</b>	87.8 ± 1.02

(continued)

Table 1. (continued)

	Protein type	Model	Average across all proteins		
			$S_{10}$	$S_{100}$	$S_{1000}$
< 10 Number of rotatable bonds	known	Li et al. [22]	<b>83.4 ± 5.64</b>	<b>81.7 ± 1.94</b>	<b>81.6 ± 1.60</b>
		Grechishnikova [10]	67.6 ± 12.6	50.1 ± 8.13	–
		CDGCN	79.5 ± 3.39	79.2 ± 3.33	78.7 ± 2.08
	novel	Grechishnikova [10]	77.4 ± 5.38	43.5 ± 4.14	–
		CDGCN	<b>78.2 ± 2.89</b>	<b>74.7 ± 1.74</b>	74.8 ± 1.76
< 140 TPSA	known	Li et al. [22]	<b>89.3 ± 3.44</b>	<b>86.7 ± 1.04</b>	<b>87.0 ± 0.99</b>
		Grechishnikova [10]	66.7 ± 9.68	49.3 ± 7.43	–
		CDGCN	86.5 ± 5.12	85.4 ± 2.05	85.1 ± 1.56
	novel	Grechishnikova [10]	77.2 ± 4.67	43.5 ± 4.47	–
		CDGCN	<b>83.5 ± 3.48</b>	<b>82.0 ± 2.00</b>	82.0 ± 1.76
QED	known	Li et al. [22]	<b>0.58 ± 0.02</b>	<b>0.57 ± 0.02</b>	<b>0.57 ± 0.01</b>
		Grechishnikova [10]	0.44 ± 0.08	0.34 ± 0.06	–
		CDGCN	0.57 ± 0.01	<b>0.57 ± 0.02</b>	<b>0.57 ± 0.02</b>
	novel	Grechishnikova [10]	0.49 ± 0.06	0.31 ± 0.04	–
		CDGCN	<b>0.54 ± 0.02</b>	<b>0.54 ± 0.02</b>	<b>0.54 ± 0.02</b>
< 6 SAS	known	Li et al. [22]	<b>93.1 ± 2.88</b>	<b>90.4 ± 1.16</b>	<b>90.6 ± 0.41</b>
		Grechishnikova [10]	82.5 ± 5.81	63.4 ± 8.59	–
		CDGCN	91.1 ± 4.64	89.8 ± 1.09	89.6 ± 0.79
	novel	Grechishnikova [10]	77.8 ± 5.23	44.4 ± 4.37	–
		CDGCN	<b>89.5 ± 2.96</b>	<b>88.2 ± 1.67</b>	87.8 ± 0.96
$R_c$	known	Li et al. [22]	57.4 ± 3.30	42.1 ± 1.02	40.1 ± 0.92
		Grechishnikova [10]	<b>85.2 ± 2.28</b>	<b>77.9 ± 2.19</b>	–
		CDGCN	62.1 ± 1.84	46.7 ± 1.25	43.9 ± 0.94

with i) 100 molecules generated by Li et al. [22] for the known proteins, ii) 100 molecules generated by the pretrained CDGCN without Embed<sub>p</sub>, and iii) 100 molecules generated for each novel protein by the finetuned CDGCN with Embed<sub>p</sub>. We found that the finetuned CDGCN with Embed<sub>p</sub> gave the best binding energies for all five novel proteins among all the three models. Detailed ablation study is provided in the supplementary.



**Table 2.** Binding energies of known ligands and generated molecules for CDGCN and Grechishnikova [10] for each of the five novel proteins. ‘Best’ shows the best binding energy for the docked molecules. The PDB codes of the PDB structures used for molecular docking for each protein is also reported for reference. Better values are in bold. Underlined values are better than the values for the known ligand. ‘–’ indicates values not reported due to memory constraint.

PDB code			Best	$\leq -7.0$ kcal/mol (%)
4nos		Known ligands	-7.9	46.2
	$S_{10}$	Grechishnikova [10]	-6.2	0.0
		CDGCN	<b><math>-7.3 \pm 0.24</math></b>	<b><math>17.0 \pm 10.00</math></b>
	$S_{100}$	Grechishnikova [10]	-7.6	2.8
		CDGCN	<b><math>-8.0 \pm 0.38</math></b>	<b><math>12.1 \pm 2.90</math></b>
	$S_{1000}$	Grechishnikova [10]	–	–
CDGCN		<u><math>-8.6 \pm 0.12</math></u>	$12.4 \pm 0.70$	
2nru		Known ligands	-8.8	93.5
	$S_{10}$	Grechishnikova [10]	-7.9	<b>90.0</b>
		CDGCN	<b><math>-8.3 \pm 0.37</math></b>	$59.3 \pm 18.60$
	$S_{100}$	Grechishnikova [10]	-8.1	52.4
		CDGCN	<b><math>-8.9 \pm 0.27</math></b>	<b><math>58.9 \pm 4.30</math></b>
	$S_{1000}$	Grechishnikova [10]	–	–
CDGCN		<u><math>-10.1 \pm 0.45</math></u>	$59.5 \pm 1.70$	
1lqf		Known ligands	-11.0	99.4
	$S_{10}$	Grechishnikova [10]	-9.3	80.0
		CDGCN	<b><math>-9.5 \pm 0.26</math></b>	<b><math>95.7 \pm 5.30</math></b>
	$S_{100}$	Grechishnikova [10]	-10.1	88.0
		CDGCN	<b><math>-10.2 \pm 0.12</math></b>	<b><math>92.8 \pm 2.20</math></b>
	$S_{1000}$	Grechishnikova [10]	–	–
CDGCN		<u><math>-11.1 \pm 0.21</math></u>	$94.8 \pm 0.40$	
1k2r		Known ligands	-9.0	94.2
	$S_{10}$	Grechishnikova [10]	-7.8	<b>91.3</b>
		CDGCN	<b><math>-8.3 \pm 0.29</math></b>	$63.2 \pm 4.10$
	$S_{100}$	Grechishnikova [10]	-8.2	56.1
		CDGCN	<b><math>-9.1 \pm 0.24</math></b>	<b><math>64.6 \pm 2.10</math></b>
	$S_{1000}$	Grechishnikova [10]	–	–
CDGCN		<u><math>-10.2 \pm 0.17</math></u>	$66.8 \pm 0.80$	
1cqp		Known ligands	-9.6	98.6
	$S_{10}$	Grechishnikova [10]	-7.9	<b>90.5</b>
		CDGCN	<b><math>-8.1 \pm 0.26</math></b>	$69.5 \pm 15.20$
	$S_{100}$	Grechishnikova [10]	-9.2	69.3
		CDGCN	<b><math>-9.8 \pm 0.32</math></b>	<b><math>78.0 \pm 2.50</math></b>
	$S_{1000}$	Grechishnikova [10]	–	–
CDGCN		<u><math>-10.1 \pm 0.39</math></u>	$79.7 \pm 1.20$	

## 4 Conclusion

In this work, the first deep conditional graph generative network, CDGCN, for novel protein-specific *de novo* drug generation was introduced. Computational experiments demonstrated the superiority of CDGCN as compared to the state-of-the-art method by Grechishnikova [10] in terms of binding energies of generated molecules with the corresponding proteins, percentages of valid, novel, and diverse molecules, drug-likeness, and synthetic accessibility. In CDGCN, the encoder used to encode graphs into learned embeddings utilizes graph convolu-

tion networks adapted for molecular graphs [46]. As part of our future work, the graph encoding scheme of CDGCN will be improved upon by exploring methods that provide more informative representations of molecular graphs [42, 47].

## References

1. Berman, H.M., et al.: The protein data bank. *Nucleic Acids Res.* **28**(1), 235–242 (2000). <https://doi.org/10.1093/nar/28.1.235>
2. Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. *Nat. Chem.* **4**(2), 90–98 (2012). <https://doi.org/10.1038/nchem.1243>
3. Chen, T., et al.: MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems. arXiv e-prints [arXiv:1512.01274](https://arxiv.org/abs/1512.01274), December 2015. <https://ui.adsabs.harvard.edu/abs/2015arXiv151201274C>
4. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555 (2014). <http://arxiv.org/abs/1412.3555>
5. Dallago, C., et al.: Learned embeddings from deep learning to visualize and predict protein sets. *Curr. Protoc.* **1**(5), e113 (2021). <https://doi.org/10.1002/cpz1.113>
6. van Dijk, H.K., Kloek, T.: Experiments with some alternatives for simple importance sampling in Monte Carlo integration. *Econometric Institute Archives* 272281, Erasmus University Rotterdam, May 1983. <https://doi.org/10.22004/ag.econ.272281>
7. Dunne, R., Campbell, N.: On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation functions. In: *Proceedings of the 8th Australasian Conference on Neural Networks*, pp. 181–185 (1997). <https://doi.org/10.1002/221561>
8. Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminformatics* **1**(1), 8 (2009). <https://doi.org/10.1186/1758-2946-1-8>
9. Ghose, A.K., Viswanadhan, V.N., Wendoloski, J.J.: A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. 1. A qualitative and quantitative characterization of known drug databases. *J. Comb. Chem.* **1**(1), 55–68 (1999). <https://doi.org/10.1021/cc9800071>
10. Grechishnikova, D.: Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Sci. Rep.* **11**(1), 321 (2021). <https://doi.org/10.1038/s41598-020-79682-4>
11. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: *Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908*, pp. 630–645. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46493-0\\_38](https://doi.org/10.1007/978-3-319-46493-0_38)
12. Hiremath, S., et al.: In silico docking analysis revealed the potential of phytochemicals present in *Phyllanthus amarus* and *Andrographis paniculata*, used in Ayurveda medicine in inhibiting SARS-CoV-2. *3 Biotech* **11**(2), 1–18 (2021). <https://doi.org/10.1007/s13205-020-02578-7>
13. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015*, vol. 37, pp. 448–456. *JMLR.org* (2015). <https://doi.org/10.5555/3045118.3045167>

14. Jin, W., Barzilay, D., Jaakkola, T.: Multi-objective molecule generation using interpretable substructures. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 4849–4859. PMLR, 13–18 July 2020. <https://proceedings.mlr.press/v119/jin20b.html>
15. Jin, W., Barzilay, R., Jaakkola, T.: Junction tree variational autoencoder for molecular graph generation. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2323–2332. PMLR, 10–15 July 2018. <https://proceedings.mlr.press/v80/jin18a.html>
16. Jin, W., Barzilay, R., Jaakkola, T.: Hierarchical generation of molecular graphs using structural motifs. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020. JMLR.org (2020). <https://doi.org/10.5555/3524938.3525387>
17. Kadurin, A., Nikolenko, S., Khrabrov, K., Aliper, A., Zhavoronkov, A.: druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Mol. Pharm.* **14**(9), 3098–3104 (2017). pMID: 28703000. <https://doi.org/10.1021/acs.molpharmaceut.7b00346>
18. Kloek, T., van Dijk, H.K.: Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica* **46**(1), 1–19 (1978). <http://www.jstor.org/stable/1913641>
19. Koes, D.R., Baumgartner, M.P., Camacho, C.J.: Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *J. Chem. Inf. Model.* **53**(8), 1893–1904 (2013). <https://doi.org/10.1021/ci300604z>
20. Landrum, G., et al.: rdkit/rdkit: 2020\_03\_1 (q1 2020) release, March 2020. <https://doi.org/10.5281/zenodo.3732262>. <http://www.rdkit.org>
21. Leeson, P.D., Springthorpe, B.: The influence of drug-like concepts on decision-making in medicinal chemistry. *Nat. Rev. Drug Discov.* **6**(11), 881–890 (2007). <https://doi.org/10.1038/nrd2445>
22. Li, Y., Zhang, L., Liu, Z.: Multi-objective de novo drug design with conditional graph generative model. *J. Cheminformatics* **10**(1), 1–24 (2018). <https://doi.org/10.1186/s13321-018-0287-6>
23. Liao, R.: Graph neural networks: graph generation. In: Wu, L., Cui, P., Pei, J., Zhao, L. (eds.) Graph Neural Networks: Foundations, Frontiers, and Applications, pp. 225–250. Springer, Singapore (2022). [https://doi.org/10.1007/978-981-16-6054-2\\_11](https://doi.org/10.1007/978-981-16-6054-2_11)
24. Lipinski, C.A., Lombardo, F., Dominy, B.W., Feeney, P.J.: Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **46**(1–3), 3–26 (2001). [https://doi.org/10.1016/s0169-409x\(00\)00129-0](https://doi.org/10.1016/s0169-409x(00)00129-0)
25. Lipinski, C.A.: Lead- and drug-like compounds: the rule-of-five revolution. *Drug Discov. Today Technol.* **1**(4), 337–341 (2004). <https://doi.org/10.1016/j.ddtec.2004.11.007>
26. Lipman, D.J., Pearson, W.R.: Rapid and sensitive protein similarity searches. *Science* **227**(4693), 1435–1441 (1985). <https://doi.org/10.1126/science.2983426>
27. Liu, T., Lin, Y., Wen, X., Jorissen, R.N., Gilson, M.K.: BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Res.* **35**(suppl\_1), D198–D201 (2006). <https://doi.org/10.1093/nar/gkl999>
28. Mendez, D., et al.: ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.* **47**(D1), D930–D940 (2018). <https://doi.org/10.1093/nar/gky1075>
29. Moreno, A.J.R.: Angelruizmoreno/jupyter\_dock: v0.2.5, September 2021. <https://doi.org/10.5281/zenodo.5514956>

30. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML 2010, Madison, WI, USA, pp. 807–814. Omnipress (2010). <https://doi.org/10.5555/3104322.3104425>
31. O’Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., Hutchison, G.R.: Open babel: an open chemical toolbox. *J. Cheminformatics* **3**(1), 33 (2011). <https://doi.org/10.1186/1758-2946-3-33>
32. Oprea, T.I., Davis, A.M., Teague, S.J., Leeson, P.D.: Is there a difference between leads and drugs? A historical perspective. *J. Chem. Inf. Comput. Sci.* **41**(5), 1308–1315 (2001). <https://doi.org/10.1021/ci010366a>
33. Polishchuk, P.G., Madzhidov, T.I., Varnek, A.: Estimation of the size of drug-like chemical space based on GDB-17 data. *J. Comput. Aided Mol. Des.* **27**(8), 675–679 (2013). <https://doi.org/10.1007/s10822-013-9672-4>
34. Prechelt, L.: Automatic early stopping using cross validation: quantifying the criteria. *Neural Netw.* **11**(4), 761–767 (1998). [https://doi.org/10.1016/S0893-6080\(98\)00010-0](https://doi.org/10.1016/S0893-6080(98)00010-0)
35. Pushpakom, S., et al.: Drug repurposing: progress, challenges and recommendations. *Nat. Rev. Drug Discovery* **18**(1), 41–58 (2019). <https://doi.org/10.1038/nrd.2018.168>
36. Rice, P., Longden, I., Bleasby, A.: EMBOSS: the European molecular biology open software suite. *Trends Genet.* **16**(6), 276–277 (2000). [https://doi.org/10.1016/s0168-9525\(00\)02024-2](https://doi.org/10.1016/s0168-9525(00)02024-2)
37. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**(5), 742–754 (2010). <https://doi.org/10.1021/ci100050t>
38. The PyMOL molecular graphics system, version 1.8 Schrödinger, LLC, November 2015. <https://pymol.org/2/>
39. Vamathevan, J., et al.: Applications of machine learning in drug discovery and development. *Nat. Rev. Drug Discov.* **18**(6), 463–477 (2019). <https://doi.org/10.1038/s41573-019-0024-5>
40. Vanhaelen, Q., Lin, Y.C., Zhavoronkov, A.: The advent of generative chemistry. *ACS Med. Chem. Lett.* **11**(8), 1496–1505 (2020). <https://doi.org/10.1021/acsmchemlett.0c00088>
41. Veber, D.F., Johnson, S.R., Cheng, H.Y., Smith, B.R., Ward, K.W., Kopple, K.D.: Molecular properties that influence the oral bioavailability of drug candidates. *J. Med. Chem.* **45**(12), 2615–2623 (2002). <https://doi.org/10.1021/jm020017n>
42. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2018). <https://doi.org/10.48550/arXiv.1710.10903>
43. Wedge, S., et al.: ZD4190: an orally active inhibitor of vascular endothelial growth factor signaling with broad-spectrum antitumor efficacy. *Cancer Res.* **60**(4), 970–975 (2000). <http://cancerres.aacrjournals.org/cgi/content/full/60/4/970>
44. Weininger, D.: SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28**(1), 31–36 (1988). <https://doi.org/10.1021/ci00057a005>
45. Weininger, D., Weininger, A., Weininger, J.L.: SMILES. 2. Algorithm for generation of unique SMILES notation. *J. Chem. Inf. Comput. Sci.* **29**(2), 97–101 (1989). <https://doi.org/10.1021/ci00062a008>
46. Wu, Z., et al.: MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018). <https://doi.org/10.1039/C7SC02664A>
47. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019). <https://openreview.net/forum?id=ryGs6iA5Km>



# Percolate: An Exponential Family JIVE Model to Design DNA-Based Predictors of Drug Response

Soufiane M. C. Mourragui<sup>1,2</sup>, Marco Loog<sup>2,3</sup>, Mirrelijin van Nee<sup>4</sup>,  
Mark A van de Wiel<sup>4,5</sup>, Marcel J. T. Reinders<sup>2,6</sup>(✉),  
and Lodewyk F. A. Wessels<sup>1,2</sup>(✉)

<sup>1</sup> Computational Cancer Biology, Division of Molecular Carcinogenesis, Oncode Institute, The Netherlands Cancer Institute, Amsterdam, The Netherlands

[l.wessels@nki.nl](mailto:l.wessels@nki.nl)

<sup>2</sup> Faculty of EEMCS, Delft University of Technology, Delft, The Netherlands

[m.j.t.reinders@tudelft.nl](mailto:m.j.t.reinders@tudelft.nl)

<sup>3</sup> Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

<sup>4</sup> Department of Epidemiology and Biostatistics, Amsterdam Public Health research institute, Amsterdam University medical centers, Amsterdam, The Netherlands

<sup>5</sup> MRC Biostatistics Unit, University of Cambridge, Cambridge, UK

<sup>6</sup> Leiden Computational Biology Center, Leiden University Medical Center, Leiden, The Netherlands

**Abstract. Motivation:** Anti-cancer drugs may elicit resistance or sensitivity through mechanisms which involve several genomic layers. Nevertheless, we have demonstrated that gene expression contains most of the predictive capacity compared to the remaining omic data types. Unfortunately, this comes at a price: gene expression biomarkers are often hard to interpret and show poor robustness.

**Results:** To capture the best of both worlds, i.e. the accuracy of gene expression and the robustness of other genomic levels, such as mutations, copy-number or methylation, we developed Percolate, a computational approach which extracts the joint signal between gene expression and the other omic data types. We developed an out-of-sample extension of Percolate which allows predictions on unseen samples without the necessity to recompute the joint signal on all data. We employed Percolate to extract the joint signal between gene expression and either mutations, copy-number or methylation, and used the out-of sample extension to perform response prediction on unseen samples. We showed that the joint signal recapitulates, and sometimes exceeds, the predictive performance achieved with each data type individually. Importantly, molecular signatures created by Percolate do not require gene expression to be evaluated, rendering them suitable to clinical applications where only one data type is available.

**Availability:** Percolate is available as a [Python 3.7 package](#) and the scripts to reproduce the results are available [here](#).

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_8](https://doi.org/10.1007/978-3-031-29119-7_8).

© The Author(s) 2023

H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 120–138, 2023.

[https://doi.org/10.1007/978-3-031-29119-7\\_8](https://doi.org/10.1007/978-3-031-29119-7_8)

## 1 Introduction

Over the course of their lifespan, human cells accumulate molecular alterations that result in the modification of cell behavior [27]. When aggregated at the tissue level, these alterations can compromise tissue homeostasis, in turn clinically impacting a patient [13]. Understanding the combined effect of these alterations is key to designing bespoke lines of treatment [28,33]. These molecular alterations occur at different genomic levels and are recorded using different technologies, collectively referred to as “omics” technologies. Each of these omic measurements offers only partial information regarding the compromised tissue. Aggregating different omic measurements, an analysis known as multi-omics integration, is therefore necessary to generate a comprehensive picture of the molecular features underlying a cancerous lesion [5,20].

Owing to their high versatility, cell lines offer a cost-effective model system for drug response modelling [8]. Specifically, large scale consortia have industriously subjected a large number of cell lines to hundreds of different compounds, yielding valuable drug response measurements [12,16,32]. A key challenge resides in combining these response measurements with multi-omics data to study mechanisms of resistance and sensitivity [24]. Existing approaches focus on combining all omics data types and can be ordered based on the stage of the analysis at which the integration is performed [6]. At one extreme, early integration approaches [4,19] first aggregate all features from all data types to process them all simultaneously. At the other extreme, late integration approaches first compute a representation of each data type individually, and subsequently combine these representations [11,26,36]. Several other methods can be positioned along this ordering, and differ by the analysis stage during which the grouping of data types is performed [41]. Although promising and encouraging, these methods do not take into account the quality of the data types and do not explicitly model their topology [2], i.e., how the data types relate to each other regarding information content and capacity to predict drug response. In particular, it has been observed that, although it has traditionally been the least clinically actionable data type, gene expression consistently prevails over other data types [9] and provides similar performance as early-integration approaches [1], obviating the need for complex integration strategies.

In order to maintain the predictive power of gene expression data, while exploiting the robustness of the most actionable data types, we present Percolate, an unsupervised multi-omics integration framework. Percolate sets itself apart from other integration approaches as it aims to eliminate gene expression measurements from the final predictor, rather than integrating it with all other data types. This is achieved by extracting the joint signal between gene expression and the other data types in an iterative fashion. First the joint signal between gene expression and Data type 1 (e.g. mutations) is extracted. Then the remaining signal (not shared with Data type 1) is employed to extract the joint signal of gene expression with Data type 2 (e.g. copy number data). This procedure is repeated for all omics data types. In this way, the gene expression signal is “percolated” down the other omics data types, ideally extracting all

predictive signal from the gene expression data. Technically, Percolate employs a popular framework, called JIVE [11,26], which breaks down paired datasets into joint and individual signals. We first extended JIVE to non-Gaussian noise models employing GLM-PCA [7]. Specifically, we used an alternative optimization, the decomposition of saturated parameters [21], which we theoretically proved to be competitive with the original formulation. Finally, we developed an out-of-sample extension for JIVE, useful when only one of the two data types is available.

We first show that comparing gene expression to other data types individually recovers a known topology of multi-omics data. We then show that the information shared between individual omic data types and gene expression increases drug response predictive performance for the individual omic data types. Finally, reconstructing the joint signal solely from mutation, copy-number and methylation, we show that the signatures derived from “percolating” gene expression down these data types recapitulate the drug response predictive performance of these data types.

## 2 Methods

### 2.1 Trade-off Between Robust and Predictive Types

We consider four data types: mutations (MUT), copy number aberrations (CNA), methylation (METH) and gene expression (GE). MUT and CNA directly measure genetic aberrations and therefore rely on DNA measurements. Due to several biological and technological factors, these measurements are highly robust and suffer from little technical artefacts. On the other end of the spectrum, GE measures RNA abundance, a process known for exhibiting large biological variability and prone to technical artefacts. Between these two extremes, methylation offers an intermediate level of robustness. However, when it comes to drug response prediction, the order is reversed: GE offers, on average, a better predictive performance than METH, and significantly outperforms MUT and CNA [1,8,17]. This leads to a trade-off between robustness and predictive ability (Fig. 1A) with MUT and CNA being the most robust and least predictive and GE being the most predictive and least robust, with METH rating at the intermediate level in terms of robustness and predictive capacity.

### 2.2 Exponential Family Distribution

Our integrated approach is inspired by AJIVE [11], a computational approach which takes as input two paired datasets and computes a joint and a data-specific signals. AJIVE is an extension of the JIVE model [26], which we selected, among other extensions [35,37], for its computational tractability and its mathematical formulation which is amenable to the derivation we propose. JIVE, AJIVE, and derivations thereof, critically rely on Principal Component Analysis (PCA) which assumes a Gaussian noise model on the data [22,39]. To extend this

**Table 1. Exponential family distributions.** Gaussian distribution is assumed to have unit variance. The dispersion parameter  $r$  is fixed for the Negative Binomial.

Data type	Family distribution
Copy-number aberration (CNA)	Log-Normal or Gamma
Gene expression (GE)	Negative-Binomial
Methylation (METH)	Beta
Mutation (MUT)	Bernoulli

framework to non-Gaussian settings, we make use of a generalized formulation that can deal with a wider class of parametric distribution models, i.e., the so-called exponential family [31].

**Definition 2.1** (Exponential family distribution). *Let  $\mathcal{X} \subset \mathbb{R}^p$ , we say that a random vector  $Z \in \mathcal{X}$  follows an **exponential family distribution** if its probability density function  $f$  can be written as*

$$\forall z \in \mathcal{X}, \quad f(z|\theta) = h(z) \exp\left(\eta(\theta)^T T(z) - A(\theta)\right). \quad (1)$$

$T : \mathcal{X} \rightarrow \mathbb{R}^q$  ( $q > 0$ ) is called the **sufficient statistics**,  $\theta \in \mathbb{R}^q$  the **exponential parameter**,  $\eta : \mathbb{R}^q \rightarrow \mathbb{R}^q$  the **natural parametrization**,  $A : \mathbb{R}^q \rightarrow \mathbb{R}$  the **log-partition function** and  $h : \mathcal{X} \rightarrow \mathbb{R}^+$  the base measure.

The exponential family encompasses a broad set of distributions (Supp. Table 1), including the Gaussian distribution with unit variance, the Poisson, the Bernoulli, the Beta or the Gamma distributions. Practically, the functions  $A$ ,  $T$  and  $\eta$  are modelling choices which can be tuned for any specific application.

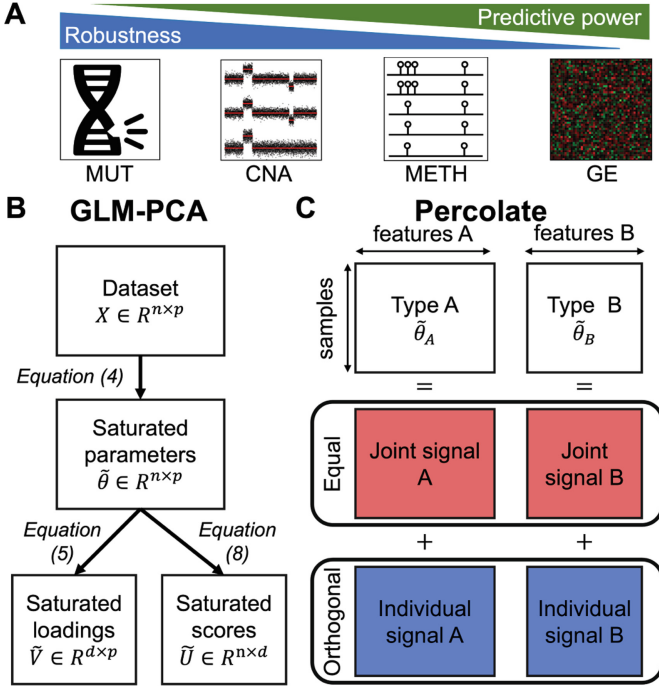
### 2.3 Saturated Model Parameters

For this section, we consider a data matrix  $X \in \mathbb{R}^{n \times p}$ , with  $n$  (resp.  $p$ ) the number of samples (resp. features). We model this data using an exponential family distribution  $\mathcal{E} = (T, A, \eta)$  (Definition 2.1), which choice is motivated by prior knowledge. For instance, if the data is known to be binary, one would turn to  $\mathcal{E}$  defined by the Bernoulli distribution, while another data distribution would lead to a different choice of functions (Supp. Table 1). We denote by  $q$  the dimensionality of  $T$  and  $A$  output space.

**Definition 2.2** (Negative log-likelihood). *We define the **negative log-likelihood**, denoted  $\mathcal{L}$ , as follows:*

$$\forall \theta \in \mathbb{R}^{n \times p \times q}, \quad \mathcal{L}(\theta; X, \mathcal{E}) = \sum_{i=1}^n \sum_{j=1}^p A(\theta_{i,j}) - \eta(\theta_{i,j})^T T(X_{i,j}). \quad (2)$$





**Fig. 1. Dissecting multi-omics topology using Percolate bridges the gap between predictive and robust data types.** (A) Trade-off between robust data types (MUT, CNA) and predictive types (METH, GE). (B) Workflow of our implementation of GLM-PCA, which relies on the projection of saturated parameters. (C) Workflow of Percolate, which extends JIVE to non-Gaussian settings by comparing the low-rank structures of saturated parameter matrices.

**Definition 2.3** (Saturated parameters). *We define the saturated parameters  $\tilde{\Theta}(X, \mathcal{E}) \in \mathbb{R}^{n \times p \times q}$  as the minimizers of  $\mathcal{L}$ , i.e.,*

$$\tilde{\Theta}(X, \mathcal{E}) = \underset{\Theta \in \mathbb{R}^{n \times p \times q}}{\operatorname{argmin}} \mathcal{L}(\Theta; X, \mathcal{E}). \tag{3}$$

The saturated parameters correspond to single-sample maximum likelihood estimates. This quantity, which will be the pillar of our approach to GLM-PCA (Sect. 2.4), can be computed as follows.

**Proposition 2.4** (Computation of saturated parameters). *Assume that  $A$  and  $\nu$  are differentiable with invertible differentials. Then, denoting  $J$  as the Jacobian of a function:*

$$\tilde{\Theta}(X, \mathcal{E}) = \eta^{-1} \circ (J_{A \circ \eta^{-1}})^{-1} \circ T(X) \hat{=} g^{-1}(X), \tag{4}$$

using an element-wise operation on all elements of  $X$ .

*Proof.* We refer the reader to the Supplementary Material (Sect. 4) for the proof.  $\blacksquare$

Proposition 2.4 shows that the saturated parameters correspond to a dual representation of the data motivated by prior knowledge on the data-distribution. We will exploit this representation ‘*a la PCA*’ to find the main sources of variations in a framework called GLM-PCA.

## 2.4 Generalized Linear Model Principal Component Analysis (GLM-PCA)

JIVE is based on Principal Component Analysis (PCA), which admits three equivalent definitions: maximization of projected variance, minimization of reconstruction error and maximization of a Gaussian likelihood with unit-variance. This latter definition can be restrictive for non-Gaussian data and we therefore set out to replace PCA by an extension called **GLM-PCA** [7]. In these methods, the Gaussian likelihood is replaced by an exponential family distribution. The original approach from *Collins et al.* [7] minimizes a negative log-likelihood using an SVD-like decomposition for the exponential parameters, yielding three different matrices. Refinements of this idea, which solve a similar optimization problem, have been proposed in the literature [23, 25] and offer competitive routines for the computation of these three matrices. Another take on this problem, which relies on the projection of saturated parameters, has recently been developed by *Landgraf et al.* [21]. This approach offers the advantage of a simpler single-matrix optimization instead of concomitantly optimizing on three. Furthermore, the out-of-sample extension relies on a matrix multiplication and is thus computationally fast. These two approaches therefore aim at finding the same decomposition through different computational routines. We here present these two approaches and prove that the latter offers a similar or better minimizer for the negative log-likelihood, which, to the best of our knowledge, had not been established.

### 2.4.1 Two Formulations of GLM-PCA

**Definition 2.5** (SVD-type [7]). *SVD-type GLM-PCA computes three matrices,  $U_{SVD} \in \mathbb{R}^{n \times d}$ ,  $V_{SVD} \in \mathbb{R}^{p \times d}$  and  $\Sigma_{SVD} \in \mathbb{R}^{d \times d}$  (diagonal), alongside a vector  $\mu_{SVD} \in \mathbb{R}^p$  defined as*

$$U_{SVD}, V_{SVD}, \Sigma_{SVD}, \mu_{SVD} \hat{=} \underset{\substack{U, V, \Sigma, \mu \\ V^T V = U^T U = I_d}}{\operatorname{argmin}} \mathcal{L}(U \Sigma V^T + 1_n \mu^T; X, \mathcal{E}) \quad (5)$$

**Definition 2.6** (Projection of saturated parameters [21]). *GLM-PCA by projection of saturated parameters computes one matrix,  $V_{SP} \in \mathbb{R}^{p \times d}$  alongside a vector  $\mu_{SP} \in \mathbb{R}^p$  defined as*

$$V_{SP}, \mu_{SP} \hat{=} \underset{\substack{V \in \mathbb{R}^{p \times d}, \mu \in \mathbb{R}^p \\ V^T V = I_d}}{\operatorname{argmin}} \mathcal{L}\left(\left(\tilde{\Theta}(X; \mathcal{E}) - 1_n \mu^T\right) V V^T + 1_n \mu^T; X, \mathcal{E}\right), \quad (6)$$

The loading matrices ( $V_{SVD}$  and  $V_{SP}$ ) and the score matrix ( $U_{SVD}$ ) have orthogonal constraints, which is similar to PCA where scores are by construction uncorrelated.

### 2.4.2 Equivalence of the Formulations

We here show that the projection of saturated parameters provides a competitive minimization when compared to the SVD-type decomposition. The main result is based on Supp. Lemma 5.1 and we refer the reader to the Supplementary Material (Sect. 5) for a complete proof.

**Theorem 2.7.** *Let us define  $U_{SVD}, V_{SVD}, \Sigma_{SVD}$  and  $\mu_{SVD}$  as in Definition 2.5, and  $V_{SP}, \mu_{SP}$  as in Definition 2.6. The likelihood resulting from the two optimization processes satisfies*

$$\mathcal{L}(U_{SVD}\Sigma_{SVD}V_{SVD}^T + 1_n\mu_{SVD}^T) \geq \mathcal{L}\left(\left(\tilde{\Theta} - 1_n\mu_{SP}^T\right)V_{SP}V_{SP}^T + 1_n\mu_{SP}^T\right), \quad (7)$$

where the dependencies on  $X$  and  $\mathcal{E}$  for  $\mathcal{L}$  and  $\tilde{\Theta}$  were removed for verbosity's sake.

Theorem 2.7 shows that, although the two approaches compute the same decomposition, the one obtained from saturated parameters yields a lower or equal negative log-likelihood. It is also worth noting that the SVD-like optimization is usually performed by alternate optimization [40] and the initialization can play a major role in the convergence. The projection of saturated parameters only requires one minimization round, and is thus faster and less prone to initialization effects. Using the decomposition of saturated parameters, however, comes at a price: there is an infinity of solutions, all equal up to a unitary transformation. In order to obtain sample scores that are uncorrelated, we proceed as follows.

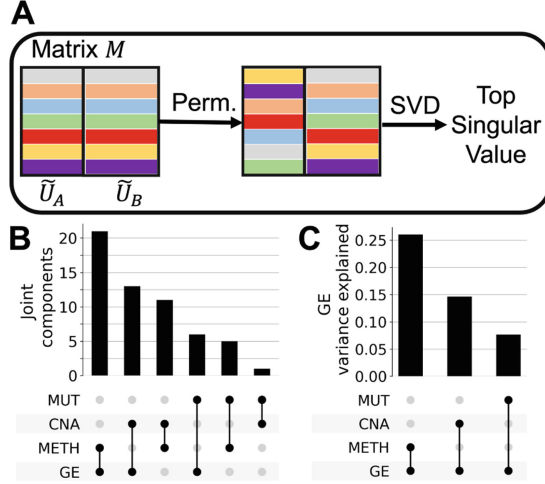
**Definition 2.8** (Sample scores). *Let  $V_{SP}$  and  $\mu_{SP}$  be defined as in Definition 2.6 and assume that  $\text{rank}\left(\tilde{\Theta} - 1_n^T\mu_{SP}\right) \geq d$ . Then  $\text{rank}\left[\left(\tilde{\Theta} - 1_n^T\mu_{SP}\right)V_{SP}V_{SP}^T\right] = d$  and we define  $U_{SP}, \Sigma_{SP}$  and  $W_{SP}$  as the unique rank- $d$  SVD decomposition of the saturated parameters, i.e.*

$$U_{SP}\Sigma_{SP}W_{SP}^T = \left(\tilde{\Theta} - 1_n^T\mu_{SP}\right)V_{SP}V_{SP}^T. \quad (8)$$

It is worth noting that the equality in Eq. 8 is not an approximation and this second SVD does not entail any loss of information. It is a pure computational maneuver to whiten the obtained scores.

### 2.4.3 Hyper-parameter Optimization

The solution of Eq. (6) is an optimization problem with a Stiefel-manifold constraint, which we solved by using recent advances in auto-differentiation [30]



**Fig. 2. Assessing the number of joint components.** (A) Schematic of the sample-level permutations we perform to estimate the number of joint components. (B) Venn-diagram of the number of joint components obtained using the permutation scheme. (C) Ratio of variance explained for the GE saturated parameters matrix after projection on the joint components.

and optimization on Riemannian manifolds [29]. We modelled the functions  $A$ ,  $T$  and the negative log-likelihood using PyTorch; stochastic gradient descent (SGD) on the Stiefel-manifold was performed using McTorch. Such a formulation allows to employ a large variety of exponential family distributions without the need for heavy and potentially cumbersome Lagrangian computations. Our optimization scheme relies on four hyper-parameters: number of factors (or principal components), learning rate, number of epochs and batch size. To determine them, we compute the Akaike Information Criterion (AIC) of the complete data for various values of  $d$  and different hyper-parameters [3]. For a GLM-PCA model with  $d$  PCs, the AIC corresponds to the sum of the data log-likelihood and the number of model parameters, which we estimate as the dimensionality of the Stiefel manifold  $\{V \in \mathbb{R}^{d \times p} | VV^T = I_d\}$ , equal to  $pd - d(d+1)/2$ . Among all trained models, we select the one which harbors the smallest AIC.

## 2.5 Comparison of GLM-PCA Directions by Percolate

**Setting:** We consider two datasets  $X_A \in \mathbb{R}^{n \times p_A}$  and  $X_B \in \mathbb{R}^{n \times p_B}$  with paired samples (rows) but potentially different features. We first perform GLM-PCA independently on  $X_A$  and  $X_B$  using two different exponential family distributions, yielding  $d_A$  and  $d_B$  factors, respectively denoted as  $\tilde{V}_A$  and  $\tilde{V}_B$ . We furthermore denote by  $\tilde{\Theta}_A$  and  $\tilde{\Theta}_B$  the saturated parameters of datasets A and B respectively, and  $\tilde{\mu}_A$  and  $\tilde{\mu}_B$  the intercept terms. Using the decomposition presented in Definition 2.8, we furthermore define  $\tilde{U}_A, \Sigma_A, \tilde{W}_A$  and  $\tilde{U}_B, \Sigma_B, \tilde{W}_B$ .

**Definition 2.9.** To compare the two sets of samples scores,  $\tilde{U}_A$  and  $\tilde{U}_B$ , we aggregate them in a matrix  $\mathbf{M}$ , which we decompose by SVD:

$$\mathbf{M} = \begin{bmatrix} \tilde{U}_A & \tilde{U}_B \end{bmatrix} = U_M \Sigma_M V_M^T . \quad (9)$$

The top left-singular vectors correspond to sample scores which are highly correlated between  $\tilde{U}_A$  and  $\tilde{U}_B$ , since both of these two matrices are consisting, by construction, of uncorrelated factors. Following the same intuition as in AJIVE, these can be understood as the **joint signal**, motivating the following definition.

**Definition 2.10** (Joint and individual signals). Let  $r_J < \min(d_A, d_B)$ , we define the **joint signal** as the matrix  $\tilde{U}_J \in \mathbb{R}^{n \times r_J}$  with the top  $r_J$  left-singular values of  $\mathbf{M}$ . We furthermore denote by  $\Sigma_J$  the diagonal matrix with the top  $r_J$  singular values of  $\mathbf{M}$ .

We define the **individual signal** of  $A$  (resp.  $B$ ), denoted as  $\tilde{U}_I^A$  (resp.  $\tilde{U}_I^B$ ), as the signal from  $\tilde{U}_I^A$  (resp.  $\tilde{U}_I^B$ ) not present in  $\tilde{U}_A$  (resp.  $\tilde{U}_B$ ), formally:

$$\begin{aligned} \tilde{U}_I^A \& = \left( I_n - \tilde{U}_J \tilde{U}_J^T \right) \tilde{U}_A \\ \tilde{U}_I^B \& = \left( I_n - \tilde{U}_J \tilde{U}_J^T \right) \tilde{U}_B . \end{aligned} \quad (10)$$

We call the complete process **Percolate**, and a summarised workflow can be found in Fig. 1B-C.

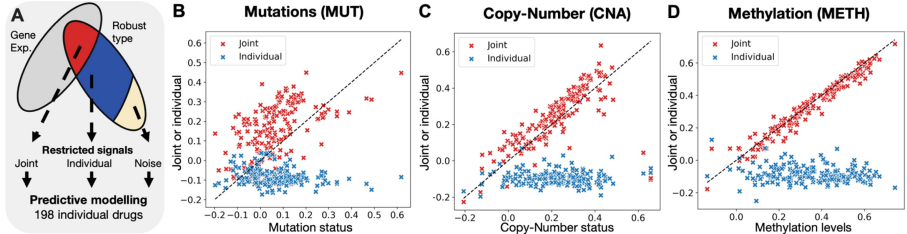
In order to set the number of joint components  $r_J$ , we employ a sample-level permutation scheme. We first independently permute the rows of  $\tilde{U}_A$  and  $\tilde{U}_B$ , which we then aggregate as in Eq. (9) to obtain the singular values. We perform 100 such permutations independently and retrieve the first singular value for each. Finally, we set  $r_J$  as the number of elements in  $\Sigma_M$  over one standard deviation from the mean of the permuted singular values (Fig. 2A).

## 2.6 Projector of Joint Signal

AJIVE does not provide an out-of-sample extension, and we here propose a derivation thereof by rewriting the matrix  $U_J$  as a function of the saturated parameters.

**Theorem 2.11.** Let's decompose the matrix  $V_M$  as  $V_M = [V_{M,A}^T \ V_{M,B}^T]^T$  such that  $V_{M,A}^T$  contains the first  $d_A$  columns of  $V_M^T$  and  $V_{M,B}^T$  the last  $d_B$  ones, we obtain:

$$\begin{aligned} \tilde{U}_J &= \tilde{U}_{J,A} + \tilde{U}_{J,B} \\ \text{with } \begin{cases} \tilde{U}_{J,A} &= \left( \tilde{\Theta}_A - 1_n \tilde{\mu}_A^T \right) \tilde{V}_A^T \tilde{V}_A W_A \Sigma_A^{-1} V_{M,A} \Sigma_J^{-1} . \\ \tilde{U}_{J,B} &= \left( \tilde{\Theta}_B - 1_n \tilde{\mu}_B^T \right) \tilde{V}_B^T \tilde{V}_B W_B \Sigma_B^{-1} V_{M,B} \Sigma_J^{-1} \end{cases} \end{aligned} \quad (11)$$



**Fig. 3.** The joint signal between robust and gene expression contains most of the predictive signal. (A) Workflow of our approach. (B) Predictive performance for MUT when using Percolate between MUT and GE. Each point corresponds to a single drug, with the x-axis corresponding to the predictive performance obtained using the original mutation data, and the y-axis by either the joint (red) or the individual (blue) signals. (C) Predictive performance for CNA, similarly displayed as in B. (D) Predictive performance for METH, similarly displayed as in B.

*Proof.* We refer the reader to the Supplementary Material (Sect. 6) for the complete proof. ■

The formulation of  $\tilde{U}_J$  presented in Equation (11) highlights the additive contribution of both dataset to the joint signal. At test time, both views are therefore required to estimate the joint signal. To tackle the issue of missing data-view, we propose a nearest-neighbor imputation of the unknown joint-term. Let’s consider, without loss of generality, that only the view  $A$  is available. The joint signal has been computed using the two data matrices  $X_A$  and  $X_B$ , yielding  $\tilde{U}_{J,A}$  and  $\tilde{U}_{J,B}$ . The second term contains  $r_J$  terms, and we train  $r_J$  corresponding k-Nearest-Neighbors (kNN) regressors. The test dataset  $Y_A \in \mathbb{R}^{m \times p_A}$  can be projected on the joint signal by replacing the saturated parameter  $\tilde{\Theta}_A$  in Eq. 11 with the saturated parameter of the test data. We then estimate the second term by means of the  $r_J$  kNN regression models. Adding these two terms yields an estimate of the joint signal.

## 2.7 Drug Response Prediction

We assess the predictive performance of a dataset by employing ElasticNet [42], which has been shown, inspite of its relative simplicity, to outperform more complex non-linear models when it comes to drug response prediction [8, 17, 38]. For a given dataset, we perform nested cross-validation as follows. First, datasets are stratified into 10 groups of equal size. For each group (10%), we employ a 3-fold cross-validation grid search on the remaining 90% to determine the optimal ElasticNet hyper-parameters ( $\ell_1$ -ratio and penalization). We then fit this optimal ElasticNet model on the 90% to predict the class labels on the 10%. Repeating this procedure, we obtain one cross-validated estimate per sample and we define the **predictive performance** as the Pearson correlation between these estimates and the actual values.

## 2.8 Data Download, Modelling and Processing

We consider four data types in our analysis (Table 1) which we modelled using different exponential family distributions (Supp. Material). The GDSC data was accessed on January 2020 from CellModel Passport [16]. For GE, MUT and CNA, we restricted to protein coding genes known to be frequently mutated in cancer, referred to as the **mini-cancer genome** [15]. GE was corrected for library size using TMM normalization [34] and mutations were restricted to non-silent.

## 3 Results

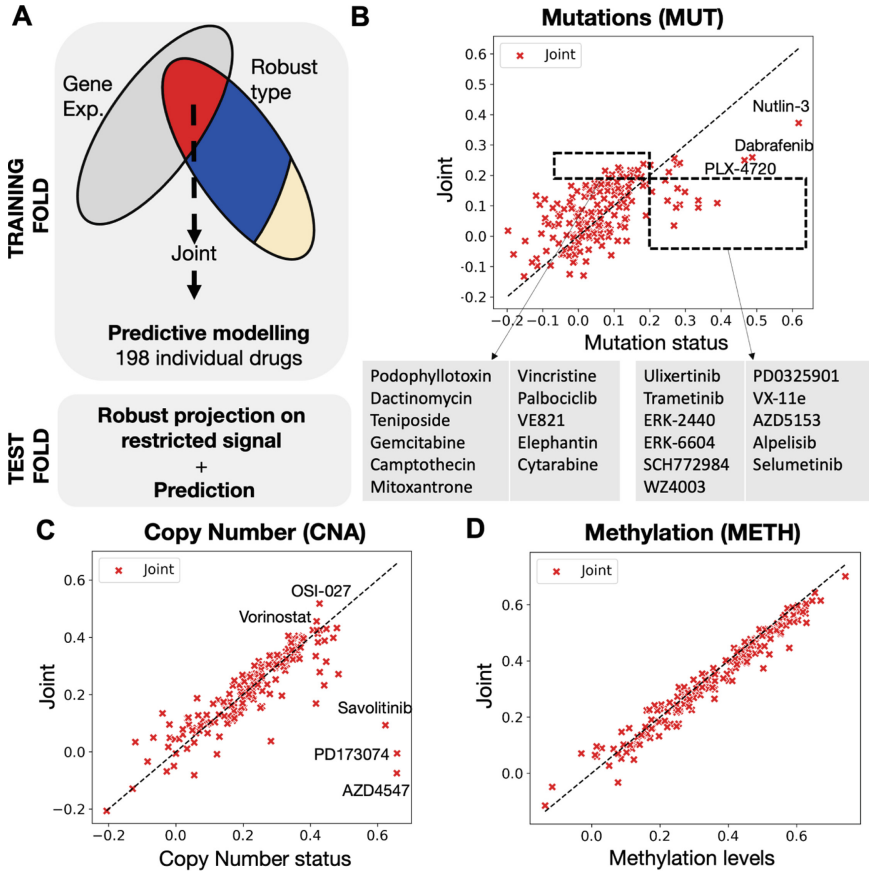
### 3.1 The Breakdown of the Joint Signals Highlights the Topology of Multi-omics Data

To compare data types, we employ Percolate using the distributions defined in Table 1, and a number of PCs set using the procedure presented in Subsect. 2.4 (Supp. Figure 2). For each comparison, setting the number of joint components is a crucial step, as it defines the threshold between the joint and individual signals. For that purpose, we used a sample level permutation test (Fig. 2A, Subsect. 2.5).

We observe that GE shares 21 joint components with METH, 13 with CNA and only 6 with MUT, which is coherent with the gradient put forward in Fig. 1. We furthermore observe that MUT is consistently the data type with the least number of joint components (Fig. 2B), highlighting the weakness of the signal coming from MUT data, corroborating previous measured topologies of multi-omics data [2]. To measure the strength of the underlying joint signals, we computed the proportion of GE variance explained by the joint directions (Fig. 2C), computed as the ratio between the joint signal variance and the variance of the GE's saturated parameters matrix. We observe that the joint signal between GE and METH explains 26% of GE variance, while this figures drops to 14% and 7% for CNA and MUT, respectively. These observations highlight the existence of a joint signal, of which the predictive performance can be interrogated.

### 3.2 Robust Signal Predictive of Drug Response Is Concentrated in the Joint Part

We then investigated the relevance of the joint and individual signals when it comes to drug response prediction. Considering one robust data type at a time (MUT, CNA or METH), we first decomposed the original robust data type into a signal joint with GE and an individual signal specific to the robust data type. We then computed, for 195 drugs (Methods), the predictive performance for these two signals and compared it to the original robust data (Fig. 3A, Subsect. 2.7). To ensure a proper comparison between joint, individual and cell-view, the cross-validation was performed using the same folds for all datasets. As ElasticNet has been shown in the literature to outperform other more advanced algorithms for this particular task [8, 17, 38], we restricted our comparison to



**Fig. 4. Robust-type-based signatures created from Percolate recapitulate drug response.** (A) Schematic of the cross validation experiment. (B) Results for MUT with a special zoom on drugs predictive for joint but not robust (left) and for robust but not joint (right). (C) Results for CNA. (D) Results for METH.

this regression method. Such experimental design has the advantage to properly assess the effect of Percolate, as no additional performance can be gained from the regression model.

We first analyzed the results obtained between MUT and GE data (Fig. 3B). We observe that for most drugs, the predictive performance of the joint signal exceeds the predictive performance of the original robust signal, except for a number of drugs of which the response is quite well predicted based on MUT only. This set includes the drugs Nutlin-3, Dabrafenib, and PLX-4720. In contrast, the individual signal shows no predictive performance (Pearson correlation below 0) for most drugs, indicating an absence of drug response related signal in the individual portion. We then turned to CNA where the choice of distribution was unclear, with, to the best of our knowledge, no clear precedent on how to

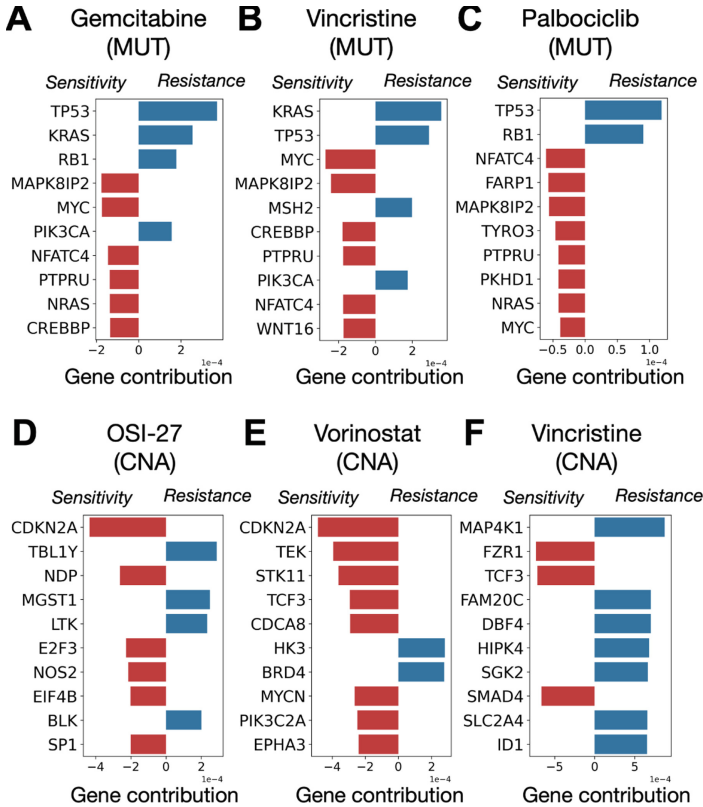


model such data. Due to the observed behavior of CNA data, we opted for two possible distributions: Log-normal and Gamma distributions (Supp. Table 1). We observe that the joint signal computed using a Gamma-distribution yields better performances than the log-normal model (Supp. Figure 3A-B). When using a Gamma distribution, a conclusion similar to the MUT data can be reached with the majority of drugs predicted well with the joint signal except three drug, AZD4547, PD173074 and Savolitinib (Fig. 3C). This advocates for using the Gamma distribution for analyzing CNA data and shows that the joint signal presents an increased performance while the individual signal is not predictive. Finally, we studied the drug response performance obtained after decomposing METH using GE (Fig. 3D). We observe that the joint signal presents a similar predictive performance as the original methylation data. The individual signal is, again, not predictive. These results highlight the potential of restricting predictors to the joint signal for robust data types.

### 3.3 Out-of-sample Extension Recapitulates the Predictive Performance of Robust Signal

In order to compute the joint signal between one robust data type and GE, one needs to have access to both modalities. However, the purpose is to become independent of non-robust GE measurements. In order to study whether the joint signal could be estimated without access to gene expression, when the predictor is applied to a test case, we exploited our out-of-sample extension (Subsect. 2.6). We employed this algorithm to compute the drug response predictive performance of the joint signal estimated using the robust data alone (Fig. 4A). Dividing the data in ten independent folds, we performed a cross-validation estimation as follows. For each train-test division of the data, we trained a Percolate instance on the 90% of the data, the training set containing GE and the robust data type. The resulting joint information was then used to train an ElasticNet model to predict drug response. The remaining 10% (test data) were then used to first estimate the joint signal, solely based on the robust data (Subsect. 2.6). This joint signal was then used as input into the ElasticNet model to predict the response on this test set. Finally, we computed the predictive performance as indicated in Subsect. 2.7.

When analyzing results for MUT (Fig. 4B), we first observe a clear drop in performance for the joint signal compared to the previous results (Fig. 3B). This suggests that the GE portion of the joint signal (Eq. 11) contains a significant portion of predictive signal, which is less well captured by our out-of-sample extension. Nonetheless, we observe that 11 drugs show a predictive performance above 0.2 for joint but not for the robust data. In contrast, 11 drugs show the opposite effect, including seven which target the MAPK pathway – MEK (Trametinib, PD0325901, Selumetinib) and ERK (ERK2440, ERK6604, Ulixertinib, SCH772984). BRAF inhibitors Dabrafenib and PLX-4720 also show a drop in performance. This suggests that constitutive activation of the MAPK pathway is not recapitulated by the joint signal. Nonetheless, the joint signal generated by Percolate helps increase performance for several poorly predictive



**Fig. 5. Study of joint signals contributing to improved performance.** For each drug, we report the top 10 largest gene regression coefficients from the joint signal, in absolute values. We first analysed the joint biomarkers created from MUT data for Gemcitabine (A), Vincristine (B) and Palbociclib (C). We then turned to CNA-based signatures for OSI-27 (D), Vorinostat (E) and Vincristine (F).

drugs and is therefore of interest to study various response mechanisms. We then turned to CNA (Fig. 4C) and observe a modest decrease in predictive performance compared to the performance on the original CNA profiles. Three drugs show a spectacular drop as the response can not be predicted by the joint signal – Savolitinib (cMET), PD173074 (FGFR) and AZD4547 (FGFR). In contrast, three drugs show improved performance for the joint signal – OSI-027 (mTOR), Navitoclax (HDAC) and Vincristine (tubulin). Finally, we repeated the experiment for METH (Fig. 4D) and observe that predictive performances of the joint signal is remarkably comparable to the predictive performance on the original METH data, with most drugs falling showing less than 2% relative performance difference (Supp. Figure 4C). Taken together, these results show that the joint signal recapitulates the drug response performance abilities of DNA-based measurements.

### 3.4 Study of Genes Contributing to the Joint Signals

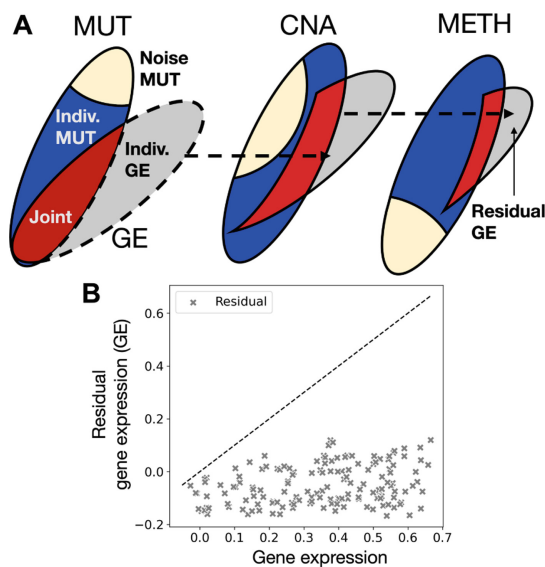
We then set out to study the underlying mechanisms associated with the predictors derived from the robust data types (Subsect. 3.3) which also lead to improved performance. For a given drug, we trained an ElasticNet model on the joint signal, yielding one regression coefficient per joint component. Using the relationship from Eq. 11, we obtain a regression coefficient for each gene. A positive coefficient indicates that larger values of the saturated parameters, caused by a mutation or amplification of the supporting gene, are associated with resistance. In contrast, a negative coefficient indicates that larger values of the saturated parameters are associated with sensitivity.

For MUT, we studied the mode of action of three drugs for which the joint signal performs well (Fig. 4B): Gemcitabine (Fig. 5A), Vincristine (Fig. 5B) and Palbociclib (Fig. 5C). We observe that TP53 mutation status is associated with resistance to three drugs, concordant with earlier observations showing that TP53 mutant are more resistant to chemotherapy [14]. Resistance to Gemcitabine and Vincristine is also associated with KRAS and PI3KCA mutations, known for their proliferative potential [10, 18]. Interestingly, mutations in MYC and MAPK8IP2 are associated with sensitivity to these three drugs. Three other drugs show a drop in predictive performance on the joint signal as compared to the original signal: Nutlin-3, Dabrafenib and PLX-4720 (Fig. 4B). We observe that the known targets of these drugs exhibit a large coefficient: TP53 for Nultin-3 (known resistance biomarker) and BRAF for Dabrafenib and PLX-4720 (Supp. Figure 5). These three drugs highlight a limitation of our approach: GLM-PCA generates scores which aggregates the contributions of several genes. Highly-specific drugs, like Nutlin-3 (Mdm2-inhibitor) or BRAF/MEK-inhibitors not only target a specific protein, but mutations in the target are excellent response predictors. Such cases do not benefit from the GLM-PCA aggregation as a single feature alone is predictive.

Next we turned to CNA where three drugs: OSI-27 (Fig. 5D), Vorinostat (Fig. 5E) and Vincristine (Fig. 5F), which all showed increased performance when the joint signal is employed as compared to the original CNA data. For both OSI-27 (mTORC1) and Vorinostat (HDAC), we observe that amplification of CDKN2A (p16) is associated with sensitivity. P16 acts as a tumor-suppressor by slowing down the early progression of the cell-cycle and its loss is here associated with resistance for these two drugs. Finally, Vincristine’s predictor shows that MAP4K1’s amplification as a predictor of resistance. Such result is coherent with what we observed for MUT (Fig. 5B) where mutations on KRAS were associated with resistance.

### 3.5 Iterative Application of Percolate Deprives Gene Expression from Predictive Power

Finally, we questioned whether some signal predictive of drug response is still present in gene expression. To this end, we studied the GE signal after it has been



**Fig. 6. The signal joint with DNA-based measurements deprives gene expression from any predictive power. (A)** Schematic of our iterative procedure to remove from GE any signal joint with robust data type. **(B)** Predictive performance of the resulting residual gene expression compared to the predictive performance of the complete gene expression.

stripped of all the signal it shares with MUT, METH or CNA. To remove all signal associated with robust data types from GE, we used Percolate iteratively on GE, starting with the *least* predictive data type (MUT), followed by CNA and ending with the *most* predictive data type (METH) (Fig. 6A). Specifically, we first “percolate” GE through MUT to obtain an individual GE signal (not shared with MUT), which is then percolated through CNA to obtain a second GE individual signal, which is then finally percolated through METH, resulting in the individual GE signal we denote as *residual gene expression*. We finally assessed the predictive performance of this residual gene expression and compared it to the predictive performance of the original GE (Fig. 6B, Subsect. 2.7). We observe that no drug reaches a Pearson correlation above 0.16, indicative of a complete lack of predictive performance in the residual GE. This shows that removing the signal joint with DNA-based measurements deprives gene expression from any predictive ability.

## 4 Discussion

Designing multi-omics predictors of drug response has highlighted the existence of a trade-off between robust and predictive data types. To study this trade-off, we developed Percolate, a method which decomposes a pair of data types

into a joint and an individual signal. After showing that the strength of the joint signal recapitulates the known topology between data types, we showed that the joint signal contains more predictive power than any robust data type alone. Exploiting our out-of-sample extension, we showed that the joint signal, computed from robust data types alone, recapitulates most of the predictive performance of each original robust signal. Finally, we showed that the gene expression signal predictive of drug response is fully captured by robust data types through Percolate.

Although encouraging, our results display certain limitations that could inspire future methodological improvements. A key direction lies in the drop of performance between Fig. 4 and Fig. 5, caused by the out-of-sample extension. We theoretically decomposed the joint signal (Theorem 2.7) and presented an approach to approximate, using the robust type, the contribution from gene expression. We believe that this step can be improved in two ways: either by increasing the sample-size, thereby expanding the pool of potential anchors, or through the design of novel regression approaches. Another important improvement would be to extend this methodology to unpaired (single-cell) multi-omic measurements where characterizing the joint signal between omic datasets is a critical step.

Technically, Percolate extends JIVE in two different ways. First, by using GLM-PCA instead of PCA, we tailor the dimensionality reduction step to the specific data under consideration. Second, we developed an out-of-sample extension which allows to estimate the joint signal, even in the absence of one data-modality. For our analysis, we made use of standard distributions from the exponential family: Negative Binomial, Gamma, Beta or Bernoulli. Our implementation of GLM-PCA is versatile and any exponential family distribution can be employed in our framework, provided it can be auto-differentiated by PyTorch. Employing more complex distribution, like the inverse-gamma for copy-number is a fruitful avenue to improve on our methodology.

**Acknowledgements and Funding.** We wish to thank Stavros Makrodimitris (TU Delft) for useful discussions, and the RHPC of the NKI for the computing infrastructure. This work was funded by ZonMw TOP grant COMPUTE CANCER (40-00812-98-16012).

## References

1. Aben, N., et al.: TANDEM: a two-stage approach to maximize interpretability of drug response models based on multiple molecular data types. *Bioinformatics* **32**(17), i413–i420 (2016)
2. Aben, N., et al.: ITOP: inferring the topology of omics data. *Bioinformatics* **34**(17), i988–i996 (2018)
3. Akaike, H.: A new look at the statistical model identification. *IEEE Trans. Autom. Control* **19**(6), 716–723 (1974)
4. Argelaguet, R., et al.: Multi-omics factor analysis—a framework for unsupervised integration of multi-omics data sets. *Mol. Syst. Biol.* **14**(6), 1–13 (2018)

5. Bersanelli, M., et al.: Methods for the integration of multi-omics data: mathematical aspects. *BMC Bioinform.* **17**(2), 167–177 (2016)
6. Cantini, L., et al.: Benchmarking joint multi-omics dimensionality reduction approaches for the study of cancer. *Nat. Commun.* **12**(1), 1–12 (2021)
7. Collins, M., et al.: A generalization of principal component analysis to the exponential family. *NeurIPS* **14**, 1–8 (2002)
8. Costello, J.C., et al.: A community effort to assess and improve drug sensitivity prediction algorithms. *Nat. Biotechnol.* **32**(12), 1202–1212 (2014)
9. Dempster, J.M., et al.: Gene expression has more power for predicting in vitro cancer cell vulnerabilities than genomics. *BioRxiv* **1**, 1–42 (2020)
10. Eklund, E.A., et al.: KRAS mutations impact clinical outcome in metastatic non-small cell lung cancer department of surgery. *Cancer* **14**, 2063 (2022)
11. Feng, Q., et al.: Angle-based joint and individual variation explained. *J. Multivar. Anal.* **166**, 241–265 (2018)
12. Ghandi, M., et al.: Next-generation characterization of the cancer cell line encyclopedia. *Nature* **569**(7757), 503–508 (2019)
13. Hanahan, D., et al.: Hallmarks of cancer: the next generation. *Cell* **144**(5), 646–674 (2011)
14. Hientz, K., et al.: The role of p53 in cancer drug resistance and targeted chemotherapy. *Oncotarget* **8**(5), 8921–8946 (2017)
15. Hoogstraat, M., et al.: Genomic and transcriptomic plasticity in treatment-Naïve ovarian cancer. *Genome Res.* **24**(2), 200–211 (2014)
16. Iorio, F., et al.: A landscape of pharmacogenomic interactions in cancer. *Cell* **166**, 740–754 (2016)
17. Jang, I.S., et al.: Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. *Pac. Symp. Biocomput.* **23**, 1–7 (2013)
18. Kim, S.T., et al.: Impact of KRAS mutations on clinical outcomes in pancreatic cancer patients treated with first-line gemcitabine-based chemotherapy. *Mol. Cancer Ther.* **10**(10), 1993–1999 (2011)
19. Kim, Y., et al.: WON-PARAFAC: a genomic data integration method to identify interpretable factors for predicting drug-sensitivity in-vivo, pp. 1–30
20. Kristensen, V.N., et al.: Kristensen - Principles and methods of integrative genomic analyses in cancer.pdf. *Nat. Rev. Cancer.* **14**, 299–313 (2014)
21. Landgraf, A.J., et al.: Dimensionality reduction for binary data through the projection of natural parameters. *J. Multivar. Anal.* **180**, 2020 (1999)
22. Lawrence, N.: Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *J. Mach. Learn. Res.* **6**, 1783–1816 (2005)
23. Li, J., et al.: Simple exponential family PCA. *IEEE Trans. Neural Netw. Learn. Syst.* **24**(3), 485–497 (2013)
24. Li, Y., et al.: A review on machine learning principles for multi-view biological data integration. *Brief. Bioinform.* **19**(2), 325–340 (2018)
25. Liu, L.T., et al.: ePCA: high dimensional exponential family PCA. *Ann. Appl. Statist.* **12**(4), 2121–2150 (2018)
26. Lock, E.F., et al.: Joint and individual variation explained (JIVE) for integrated analysis of multiple data types. *Ann. Appl. Statist.* **7**(1), 523–542 (2013)
27. Martincorena, I., et al.: Somatic mutation in cancer and normal cells. *Science* **349**(6255), 1483–1489 (2015)
28. McLeod, H.L.: Cancer pharmacogenomics: early promise, but concerted effort needed. *Science* **340**(6127), 1563–1566 (2013)
29. Meghwanshi, M., et al.: McTorch, a manifold optimization library for deep learning, pp. 1–5 (2018)

30. Paske, A., et al.: Automatic differentiation in prose. In: NeurIPS (2017)
31. Pitman, E.J.: Sufficient statistics and intrinsic accuracy. *Math. Proc. Cambridge Philos. Soc.* **32**(4), 567–579 (1936)
32. Rees, M.G., et al.: Correlating chemical sensitivity and basal gene expression reveals mechanism of action. *Nat. Chem. Biol.* **12**(2), 109–116 (2016)
33. Relling, M.V., et al.: Pharmacogenomics in the clinic. *Nature* **526**(7573), 343–350 (2015)
34. Robinson, M.D., et al.: edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26**(1), 139–140 (2009)
35. Sagonas, C., et al.: Robust joint and individual variance explained. *CVPR* **5739–5748**, 2017 (2017)
36. Sharifi-Noghabi, H., et al.: MOLI: multi-omics late integration with deep neural networks for drug response prediction. *Bioinformatics* **35**(14), i501–i509 (2019)
37. Shu, H., et al.: D-CCA: a decomposition-based canonical correlation analysis for high-dimensional datasets. *J. Am. Stat. Assoc.* **115**(529), 292–306 (2020)
38. Smith, A.M., et al.: Standard machine learning approaches outperform deep representation learning on phenotype prediction from transcriptomics data. *BMC Bioinform.* **21**(1), 1–18 (2020)
39. Tipping, M.E., et al.: Probabilistic principal component analysis. *J. Roy. Stat. Soc. B* **61**(3), 611–622 (1999)
40. Townes, F.W., Hicks, S.C., Aryee, M.J., Irizarry, R.A.: Feature selection and dimension reduction for single-cell RNA-SEQ based on a multinomial model. *Genome Biol.* **20**(1), 1–16 (2019)
41. Wang, B., et al.: Similarity network fusion for aggregating data types on a genomic scale. *Nat. Methods* **11**(3), 333–337 (2014)
42. Zou, H., et al.: Regularization and variable selection via the elastic net. *Hui. J. Statist. Soc. Ser. B* **67**(2), 301–320 (2005)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





# Translation Rate Prediction and Regulatory Motif Discovery with Multi-task Learning

Weizhong Zheng<sup>1</sup>, John H.C. Fong<sup>1</sup>, Yuk Kei Wan<sup>1</sup>, Athena H.Y. Chu<sup>1,6</sup>,  
Yuanhua Huang<sup>1,3,5</sup>, Alan S.L. Wong<sup>1,4,6</sup>, and Joshua W.K. Ho<sup>1,2</sup>(✉)

- <sup>1</sup> School of Biomedical Sciences, Li Ka Shing Faculty of Medicine, The University of Hong Kong, Hong Kong SAR, China  
jwkho@hku.hk
- <sup>2</sup> Laboratory of Data Discovery for Health (D<sup>2</sup>4H) Limited, Hong Kong Science Park, Hong Kong SAR, China
- <sup>3</sup> Department of Statistics and Actuarial Science, The University of Hong Kong, Hong Kong SAR, China
- <sup>4</sup> Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China
- <sup>5</sup> Center for Translational Stem Cell Biology, Hong Kong Science and Technology Park, Hong Kong SAR, China
- <sup>6</sup> Centre for Oncology and Immunology, Hong Kong Science Park, Hong Kong SAR, China

**Abstract.** Many studies have found that sequence in the 5' untranslated regions (UTRs) impacts the translation rate of an mRNA, but the regulatory grammar that underpins this translation regulation remains elusive. Deep learning methods deployed to analyse massive sequencing datasets offer new solutions to motif discovery. However, existing works focused on extracting sequence motifs in individual datasets, which may not be generalisable to other datasets from the same cell type. We hypothesise that motifs that are genuinely involved in controlling translation rate are the ones that can be extracted from diverse datasets generated by different experimental techniques. In order to reveal more generalised cis-regulatory motifs for RNA translation, we develop a multi-task translation rate predictor, *MTtrans*, to integrate information from multiple datasets. Compared to single-task models, *MTtrans* reaches a higher prediction accuracy in all the benchmarked datasets generated by various experimental techniques. We show that features learnt in human samples are directly transferable to another dataset in yeast systems, demonstrating its robustness in identifying evolutionarily conserved sequence motifs. Furthermore, our newly generated experimental data corroborated the effect of most of the identified motifs based on *MTtrans* trained using multiple public datasets, further demonstrating the utility of *MTtrans* for discovering generalisable motifs. *MTtrans* effectively integrates biological insights from diverse experiments and allows robust extraction of translation-associated sequence motifs in 5'UTR.

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-29119-7\\_9](https://doi.org/10.1007/978-3-031-29119-7_9).



**Keywords:** sequence modeling · multi-task learning · motif discovery · eukaryotic translation · explainable AI

## 1 Introduction

For eukaryotic translation systems, the primary point of regulation of translation occurs at the initiation stage [14, 24, 29]. The 5' untranslated regions (5'UTR) encode many important sequence features, collectively shaping the initiation efficiency [3, 9]. While the molecular mechanism of translation regulation has been studied for decades, it remains challenging to predict how well the 5'UTR of a transcript impacts its translation. The discovery of predictive sequence motifs in the 5'UTR and how the interplay of these elements exert their regulatory effects remain an important research area in molecular biology.

There is a rapid evolution in methods, both experimental and computational, to identify the regulatory code of translation control from the high throughput sequencing-based translation rate profiling. The most common sequencing-based techniques include Ribosome Profiling (RP) [13], Fluorescence-Activated Cell Sorting (FACS) screening coupled with deep sequencing, and Massively Parallel Reporter Assays (MPRA) [7, 25]. Previous hypothesis-driven methods required one to firstly have a set of candidate features and test them in the RP dataset or FACS-screening dataset afterwards [6, 9, 18, 21]. Deep learning, an alternative approach to discover the regulatory grammar in a data-driven manner [1, 4, 7, 16, 31], can capture the rich pattern of translation regulatory motifs in 5'UTRs from the ultra-high throughput MPRA datasets [15, 25].

However, deep neural networks can capture unintended rule that relies on dataset-specific covariates, also called the 'short-cut' [8, 10]. Even though seemingly successful in one dataset, short-cut features may fail to generalise to slightly different circumstances. Karollus et al. have found that the sequence motifs discovered from the MPRA dataset can hardly generalize to endogenous datasets [15], which is likely caused by the fact that these experimental techniques probe different facets of the translation regulation system. This poses a challenge to distinguish actual translation-related motifs from the false positives caused by dataset-specific artefacts [10].

To resolve the issue, we propose a multi-task translation rate prediction model *MTtrans* which can gather insight from various datasets for more accurate prediction and co-optimize a set of translation regulatory features that generalise across techniques. The fundamental assumption of our work is that *robust sequence features* in 5'UTR that predict translation rate across multiple datasets generated by different experimental techniques are more likely to be actual regulatory elements for mRNA translation. In this work, we demonstrated that our model could outperform existing methods in both synthetic sequences and endogenous sequences. Further, our model identifies sequence features that are generalisable to an independent dataset collected from yeast. We determined what combinations of tasks would lead to the most robust predictive models and derive sequence motifs. As further validation, we conducted an independent FACS screening experiment to validate the identified sequence motifs.

## 2 Methods

### 2.1 *MTtrans* Model

Our multi-task learning model *MTtrans* applies the hard parameter sharing to encode the information from several task-specific inputs. The model consists of a shared encoder  $f_e$  to embed sequences into a shared space and the task-specific towers  $f_w^t$  to capture the variance of the translation rate for the  $t$ -th task. Suppose a total of  $T$  tasks which are sequence libraries from any techniques, are integrated to train the model and the data for task  $t$  is denoted as  $\{x_i^t, y_i^t\}$  with paired translation rate  $y_i^t, t = \{1, \dots, T\}, i = \{1, \dots, N^t\}$ .

**The Shared Encoder.** We stacked four 1d-convolution layers as the main building block for the shared encoder. Raw sequences will first be one-hot encoded, ending up a 4 dimensional input  $x_i^t$  for *conv1*. Noted we don't include any pooling operation in the encoder following the convolution layer for a better sense of the location. Instead, Batch Normalisation (BN) and drop-out operation is performed after each convolution layer to stabilise the activation and have a more robust model performance. The hidden unit at layer  $l$  is computed by  $h_i^{t,l} = \text{ReLU}(W^l h_i^{t,l-1} + b^l)$  and the batch-normalized activation by  $a_i^{t,l} = \text{BN}(h_i^{t,l})$ .

**The Task Specific Tower.** The number of towers corresponds to the number of tasks selected. The  $t$ -th tower  $f_w^t$  consists of a 2-layer Gated Recurrent Unit (GRU) and an output dense layer. The hidden size of the GRU is set to 80 and the hidden state of the last time point is taken to connect to the output layer. Taken together, the translation rate is predicted by  $\hat{y}_i^t = f_w^t(a_i^{t,L}) = f_w^t(f_e(x_i^t))$  wherein  $a_i^{t,L}$  is the activation from the last convolution layer.

**Model Training.** One of the biggest differences in training our multi-task model is that only the  $t$ -th towers will be updated at one time. Technically, there is a task switch activating the corresponding tower for each mini-batch sampled from  $\mathcal{X}^t$ , while the shared encoder receives gradient for all the tasks. Each instance in  $\mathcal{X}^t$  is a pair of sequences  $x_i^t$  and the translation rate  $y_i^t$ , which is defined by the mean ribosome loading for MPRA tasks [25] and the translation efficiency for RP tasks [2, 12, 28]. The cost is quantified by the naive mean square error weighted by  $\lambda^t$  for  $t \in \{1, \dots, T\}$ :

$$\mathcal{L}^t = \lambda^t \sum_{i \in B} [y_i^t - f_w^t(f_e(x_i^t))]^2$$

A learning rate scheduler is applied to wrap the *Adam* optimizer so  $f_w^t$  and  $f_e$  will be updated in a rate that is boosted in the beginning and then goes through a dramatic decay. The small learning rate in the late training phase restricts the parameters in a narrow range. Therefore, each  $f_w^t$  docks to a similar  $\theta_{f_e}$  and

stabilise the models. The learning rate  $\epsilon$  is mediated by the largest dimension of the model  $d$ , two constants  $\tau_1, \tau_2$  and a step-renewing variable  $\delta$ , following by  $\delta = \delta + \min(\tau_2, 2\delta)$ .

$$\begin{aligned}\theta_{f_w}^* &= \theta_{f_w}^t - \epsilon \nabla_{\theta_{f_w}^t} \mathcal{L}^t \\ \theta_{f_e}^* &= \theta_{f_e} - \epsilon \sum_{t \in T} \nabla_{\theta_{f_e}} \mathcal{L}^t \\ \epsilon &= d^{-\frac{1}{2}} \times \min(\delta^{-\frac{1}{2}}, \delta * (\tau_1^{-\frac{3}{2}}))\end{aligned}$$

**Model Evaluation.** All the datasets used in our study were split into train, validation and test set. For all the MPRA tasks, the same train-test splitting was kept as in Sample et al. [25] so that the model performance was directly comparable. The remaining dataset was randomly split into training and validation set in a ratio of 9:1. The training process was terminated when the validation loss converged. For the RP tasks, the entire dataset will be split into train, validation, test set in a ratio of 8:1:1. We set up 10 runs of experiments with a different random seed each time to minimize the influence of data splitting. Similarly, the prediction accuracy was calculated by averaging the results on the test set by different random seeds using the Spearman correlation coefficient.

**Model Transfer.** Transfer learning is a useful deep learning technique to exploit the knowledge learned from related datasets to a new, and usually smaller, dataset [20, 22]. It could also be used to evaluate how informative are the learned features when fixing the feature extractor [30]. We relied a lot on the transferred performance to compare feature generalisability. All four convolutional layers are frozen when transferring *MTtrans* 3M to the yeast dataset. We randomly re-initialised a tower module, including 2 GRU layers and an output layer, and only updated parameters in these layers. The same MSE loss was used as the objective function. To transfer to our FACS-seq dataset for classification, in addition to the above-mentioned parameter fixing and tower re-initialization, we wrapped the output neuron with a sigmoid function and changed the objective function to binary cross-entropy. For FramePooling and Optimus models, we also fixed all of their convolutional layers and re-initialised the last two dense layers.

## 2.2 Extraction of Sequence Motifs from Convolutional Filters

The hidden features formulated within the convolution neuron represent the local sequence combinations that are useful for predicting the translation rate. Here in this study, we use a technique called Maximum Activation Seqlet [1, 25]. This technique can reveal the short sequence segments the convolutional filters are detecting, and then these segments are used to construct the Position Weight Matrices (PWMs) [1, 25]. Here we first segment an input sequence  $x_i^t$  into several subsequences (Seqlet)  $x_{i,j}^t$  at the same length as the receptive field of the target

neuron  $c$ . We searched for Seqlet from  $X^t$  that have the highest activation values for convolutional filter  $c$  and bagged them into  $\mathcal{P}_c$ .

$$\mathcal{P}_c = \operatorname{argmax}_{\{x_{i,j}^t \in \mathcal{X}^t\}} a_i^{t,l}$$

Piling up the short sequences in  $\mathcal{P}_c$ , we can calculate the frequency of nucleotide at each position, resulting in the Position Weight Matrices(PWMs). From here, the convolutional filters were converted into PWMs and were visualised as sequence logo using python package *logomaker version 0.8*.

### 2.3 Motif Similarity Comparison

Important translation regulatory elements can be captured independently by models trained with different datasets. In order to identify which motifs are significantly similar to annotated RBP binding motifs, motifs similarity analysis was performed using *tomtom* from the tool-kit *meme-suite version 5.4.1*. The query motifs are explained from the models in the way we described and target motifs are labelled by RNACompete [23] which is a built-in database in *meme-suite version 5.4.1*. Only motif pairs satisfying  $FDR \leq 0.05$  and  $E - value \leq 10$  at the same time will be noted as significantly similar.

### 2.4 Motifs Matching

To detect the motif occurrence using neuron activation, we first encoded the 5'UTRs by *MTtrans* to extract the feature map of the shared encoder. Then we filtered the sequences that can uniquely activate the motifs. For each convolutional filter, we ranked the 5'UTR sequences by the activated values in the feature map and kept the top-ranked sequences. Each sequence can only be assigned to one filter. When one sequence ranks top for multiple filters, the filters with the highest activation values will be assigned to it, and it will be added to the uniquely activating set for each convolution filter (channel). In the FACS-seq datasets, the effect of a motif converted from a filter is defined as the mean of log count in the uniquely activating set minus the average read count of the library. If their direction is the same, the motif effect is consistent in the new library.

We can also skip the shared encoder to match the motifs by using the 256 PWMs derived from the convolutional filters. Each matrix is scanning a 9bp long sequence window, whose matching score is produced by the inner product between the flattened PWM and one-hot sequence. We scanned along the query sequence in the stride of 1 bp and we only kept the maximal matching value across windows of every position. A UTR is considered to contain the motif when this collapsed score goes above a defined threshold. Because the score distribution differs from motif to motif, we set the activation threshold to the quantiles of their score distribution.

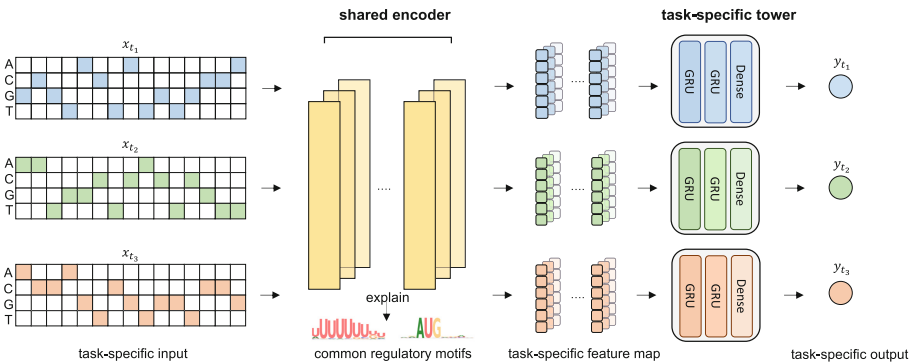
## 2.5 Building Logistic Regression and Random Forest Model on PWM-Derived Scores

The PWMs generate a positive real number which reflects the presence and strength of the motifs when scanning the input sequence. We could obtain 256 PWM scores for each sequence, which formulate the feature set to describe a sequence. The feature vector is standard scaled across all 1,052 UTRs from the two classes of our in-house FACS-seq dataset. We implemented the model training using *scikit-learn* package version 1.1.3. The training set and test set were then split in a ratio of 9:1 with different random seeds. The performance metric F1-score and AUROC score were evaluated in the test set for the Logistic Regression (LR) and Random Forest (RF) model, as well as the three deep learning baselines.

## 2.6 Identification of Important Motifs

Leveraging the white-box nature of the RF and LR model, we can obtain feature importance for 256 motifs and assign their regulation direction with the coefficient of LR. We first select the top half of the motifs ranking by the RF importance score. Motifs remain the same sign for all MPRA tasks that were first filtered (45 positives and 150 negatives), where we further identified 25 negative motifs by their intersection over negative LR coefficients. Because of fewer positive motifs left, we identified them from the top half of important features, resulting in 29 positive motifs.

## 3 Results



**Fig. 1.** The model architecture of our multi-task learning method *MTtrans*. The model consists of a shared encoder and several task-specific towers. We use the shared CNN encoder (bright yellow) to extract features and transform task-specific input  $x_t$  into task-specific feature map. The task specific tower takes in the feature map and predicts the translation rate  $y_t$  for the corresponding task. Blue, green and orange color stand for different tasks. (Color figure online)

### 3.1 *MTtrans* Learns the Shared Patterns from Multiple Experimental Systems

*MTtrans* is developed as a multi-task learning method to account for the variance of translation rate given by different experimental systems. We adapted the canonical hard parameter sharing architecture which consists of a shared encoder (bottom) and several task-specific upper layers (towers) (Fig. 1).

We can break down the forward propagation process of our neural network into two parts: sequence feature extraction and translation rate regression. The shared encoder was mainly made up of four 1D convolution layers. It can scan from the 5' to 3' end of the 5'UTR to detect sequence features like the scanning mechanism of the translation initiation. One important reason for using convolutional neural networks is that a variety of methods for deciphering convolutional neural networks have been developed in recent years [4, 11, 26], allowing for the visualisation of the regulatory signal encoded in the different layers. The towers will learn to reorient the extracted pattern for each task and make the regression from sequence pattern to translation rate (Fig. 1). The tower module is built up with a two-layer Gated Recurrent Unit network (GRU) to deal with the various input lengths and a dense layer to project the GRU memory to the output value, which is the predicted translation rate.

### 3.2 *MTtrans* Better Coordinates MPRA Tasks and Improves Translation Rate Prediction

We started with the MPRA datasets, in which the translation rate was measured by mean ribosome loading (MRL), to evaluate the effectiveness of our *MTtrans* model. We included two published models *OptimusN* [25] and *FramePooling* [15] for comparison. Another baseline called *mixing* was trained with a dataset merging all the MPRA libraries. The involvement of *mixing* was to account for the benefit of being trained from a larger integrated dataset.

Our multi-task method can predict the translation rate more accurately than the alternatives in all MPRA tasks, regardless of input length or sequence origin (Fig. 2a,b). In the largest MPRA dataset (fixed length random UTRs, short for MPRA-U), our model can consistently outperform the state-of-art result, reaching  $r^2 = 0.947$  in translation rate prediction ( $p=0.0037$  compared to Frame-Pooling, one-sided unpaired T-test, Fig. 2c & Supplementary Table 2). Interestingly, model *mixing* showed a performance loss on the fixed length human UTRs (task *MPRA-H* for *MTtrans*, the left panel of Fig. 2b), suggesting that it could be suboptimal to simply mix the datasets, probably due to covariates such as batch effect and shift in in sequence composition between random synthetic sequences and natural sequences. When modelling 5'UTR sequences with varying lengths, it turned out that using more data together with GRU layers can boost the performance strikingly. Thus, with the designed task-specific towers, our method could alleviate the conflict and effectively coordinate datasets with different study designs, benefiting all the tasks.

### 3.3 *MTtrans* is Robust Across Replicates

*MTtrans* has achieved high accuracy on the MPRA-U task, but it's reasonable to question whether it is over-fitted on the MPRA-U task and captured some unwanted batch effect. To answer that, we collected sequences having two experimental replicates from MPRA-U and found that the translation rate predicted by *MTtrans* is more consistent with that measured in replicate-2 (Fig. 2e). The absolute prediction error of *MTtrans* mostly fell in  $\pm 1MRL$  and had the smallest mean. The consistency, measured by the  $r^2$  in replicate-2, also showed the advantage of our multi-task method over the single-task baselines. Interestingly, despite a higher accuracy than *FramePooling* in replicate-1 (Fig. 2a), *OptimusN* made more errors than *FramePooling* in replicate-2. We further asked whether the failure cases by *MTtrans* were caused by the potential incorrect signal in the data. We correlated the error in replicate-1 with the variance of measurement and found a weak correlation (Fig. 2f). There are sequences strongly deviated from our prediction but highly consistent between replicates, serving as interesting cases for further investigation. Overall, training with multiple datasets can correct the model to become more robust between replicates.

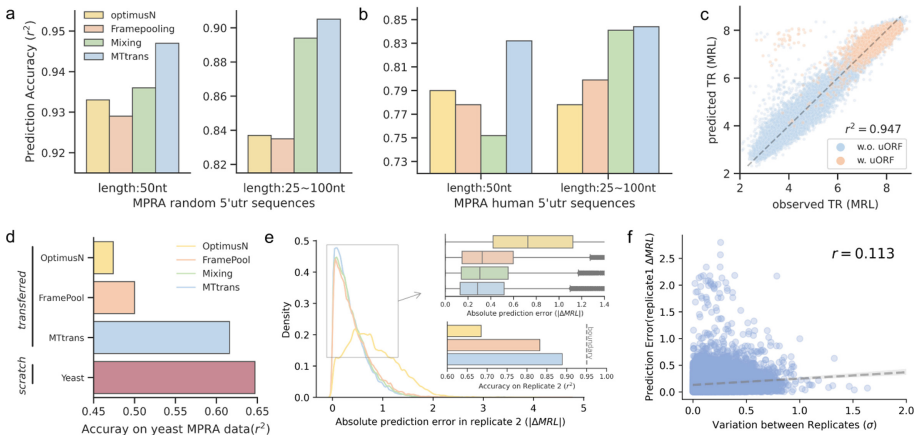
### 3.4 *MTtrans* Learns More Transferable Sequence Features

To test whether the shared encoder of *MTtrans* captures a more universal feature set, we fixed the encoder and then transferred the model to a new MPRA random 5'UTR dataset (MPRA-Yeast) generated in *Saccharomyces cerevisiae*. The performance of the transferred model with the fixed encoder can imply how general these learned features are [30].

*MTtrans* has learnt more transferable motifs than the single-task models (*OptimusN*, *FramePooling*) significantly. However, *MTtrans* was not on par with the model trained *from scratch* (box coloured with mauve in Fig. 2d), which showed the evolutionary divergence between the two translation systems despite a considerable convergence they shares. This performance gap may imply the existence of yeast-specific patterns shaped by the evolution process that can not be filled by learning more human data. The result may suggest that *MTtrans* has captured more generalisable sequence features in 5'UTR for translation initiation. Although not comparable with the yeast-oriented model, there is a significant improvement over the single-task models. Apart from seeing more sequence composition during training, using more datasets may also prevent the encoder from capturing dataset-specific patterns as it will hamper the other tasks and increase the overall loss during optimization. Overall, using the shared encoder to gather different tasks is a good strategy for learning more transferable features and may lead to genuine regulatory motifs.

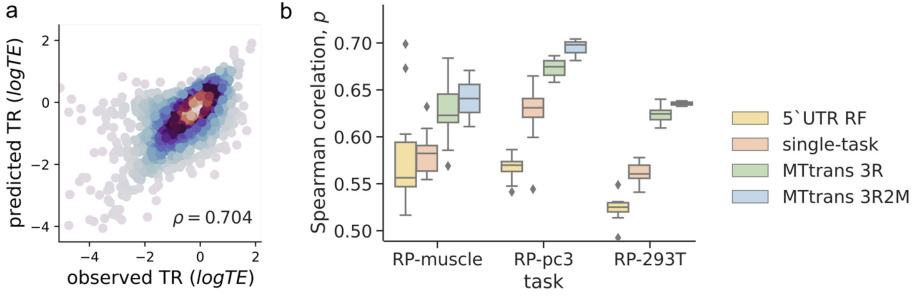
### 3.5 *MTtrans* Better Predicts the Translation Rate of Endogenous Transcripts in Human Cell Lines

We next turned to the human endogenous 5'UTRs in their natural genomic context, whose translation rate is typically assessed via ribosome profiling (RP).



**Fig. 2.** MTtrans make accurate and robust translation rate prediction by combining the signal from different task. **a** Performance for predicting mean ribosome loading (MRL) on fixed length random synthetic sequences MPRA dataset (left) and also a varying length synthetic sequences dataset (right). Prediction performance was measured by the  $r^2$  between the observed mrl value and the predicted in the held-out test set ( $n = 20,000$  for fixed length and  $n = 7,600$  for varying length). OptimusN [25], FramePooling [15] and the counterpart MTtrans model simply trained by mixing all the dataset were compared by evaluating them on the same test set. **b** Performance for predicting mean ribosome loading (MRL) on the MPRA dataset of human 5'utr sequences (left) and also a varying length synthetic sequences dataset (right). Prediction performance was measured by the  $r^2$  between the observed mrl value and the predicted in the held-out test set ( $n = 25,000$  for fixed length and  $n = 7,600$  varying length). OptimusN [25], FramePooling [15] and the counterpart MTtrans model simply trained by mixing all the dataset were compared by evaluating them on the same test set. **c** the scatter plot showing the performance for task fixed length random UTRs (MPRA-U). A high fidelity of  $r^2$  at 0.947 is reached. Sequences with uORF was colored with blue and sequences without uORF with orange respectively. **d** Transferring encoder trained from human MPRA datasets to MPRA-Yeast dataset. All the parameters in encoder are fixed for *OptimusN*, *FramePool* and *MTtrans*. While for model *Yeast*, the entire model is trained from scratch in MPRA-Yeast. **e** Robustness shown by the absolute prediction error in technical replicate. Absolute prediction error is calculated by subtracting the model prediction for MPRA-task to the label from MPRA-U replicate-2. The larger density plot showed the overall spread of error and the value ranging from 0 to 1.4  $|\Delta MRL|$  magnified in the upper right panel. The upper right box plot highlighting the densest region to display the mean and quantile of the error distribution. The bottom right bar plot label the consistency of model prediction for replicate-1 and observed value of replicate-2 by  $r^2$ . **f** The absolute prediction error is weakly correlated to with the variation observed between two replicates. Prediction error is calculated in replicate-1. Variation between replicates is calculated by firstly normalizing MRL to  $\sigma = 1$  for each replicate and then the absolute different of the two normalized MRL so that the diff of MRL by a unit of  $\sigma$ .





**Fig. 3.** *MTtrans* supports a flexible task combination strategy to make effective information sharing and promote the prediction for the noisy ribosome profiling data. **a** Observed log translation Efficiency (TE) against the log TE predicted by *MTtrans* 3R on the cell line PC-3 dataset. Color shows an increasing dot density from blue to red. **b** Model performance in three ribosome profiling datasets. Model 5'UTR RF was from the *MTtrans* 3R here was trained by the three RP datasets. The red box indicates a single-task counterpart. The blue box indicates the strategy *2M3R*. The *MTtrans* model was trained with two MPRA tasks and three RP tasks.

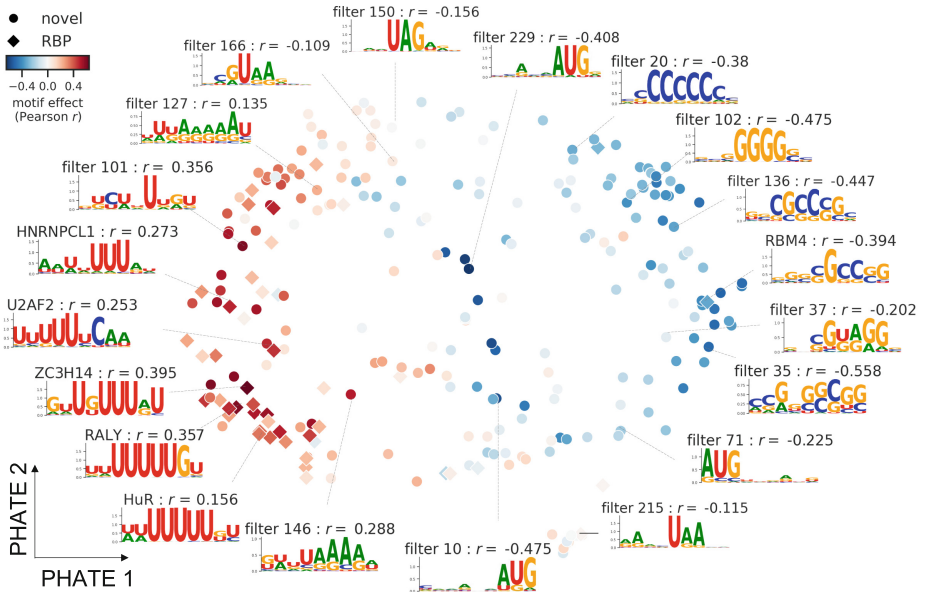
We selected three sequencing libraries from human HEK293T cell line [2], PC-3 cell line [12] and muscle tissue [28]. With the same datasets, Cao et al. built a random forest regressor (which is named 5'UTR RF in the later text) with thousands of manually crafted features to predict the translation efficiency (TE) of each transcript [6].

*MTtrans* greatly surpasses other methods in the noisy RP data (Fig. 3a&b). Deep learning-based algorithms seem to be more suitable in modelling RP when we compare *single-task* with *5'UTR RF*, which may be explained by the learned regulatory features not included in the RF model. A consistent performance gain was conferred by *MTtrans* in all cell lines. The improvement was not only observed in small source tasks like *muscle* but also held true for tasks of larger size (*i.e.* *PC-3*, Fig. 3a). Notably, adding two human 5'UTR MPRA datasets further improved the performance of all three RP tasks, suggesting beneficial information sharing from the MPRA sequence features.

### 3.6 Discovery of 5'UTR Sequence Motifs from the Deeper Layer of Shared Encoder

We next interpreted the convolutional filters in the shared encoder to see what features *MTtrans* has learned. There are studies that only explain the parameters of the first convolution layer [1, 31], but one could argue that features extracted from higher layers confer more abstraction and are more likely to represent the regulatory grammar [4, 7].

By taking the feature map from each layer for fitting random forest models, we found out that the deeper layers confer more information. (Supplementary Fig. 2a). The sequence features derived from the last layer of *MTtrans* 3M can reconstruct many biologically meaningful motifs. 43 motifs are significantly similar to the RBP binding motifs annotated by RNAcompete [23], indicating that



**Fig. 4.** Sequence motifs discovered by *MTtrans*. A total of 256 motifs are projected into PHATE space by reducing the last layer feature map input from all sequences to 2 dimension. The sequence logos of convolutional filters were generated from the position weight matrices using Maximal Activation Seqlet. Colour for the dots denotes the regulatory effect estimated in dataset MPRA-H, which is the Pearson correlation between motifs activity and translation rate. *Diamonds* indicate motifs that are significantly similar to known RBP in [23].

*MTtrans* captured the biologically meaningful elements to some degree (Fig. 4). For example, HuR have been proved to bind to 5'UTR; PABR4 has high poly-U affinity [5]; RMB4 is known to suppress the cap-dependent initiation process [19].

The upstream start codon (uAUG) and the Kozak consensus are well characterised regulatory signals that can alter the translation initiation [14, 17, 29]. AUG is present in a substantial fraction of the found motifs. 56 of the 67 uAUG detecting motifs, the majority of which include guanine right to the uAUG, are negatively correlated to the translation rate. *MTtrans* also re-discovered upstream stop codon UGA and UAA in filter 150 and filter 215.

In conclusion, by interpreting the learned shared encoder, we can rediscover many genuine regulatory signals, suggesting the advantage of *MTtrans* as a motif discovery tool.

### 3.7 The Discovered Regulatory Motifs can be Experimentally Validated

To validate whether the discovered motifs remain predictive in datasets generated by a different experimental technology, we generated an in-house FACS-seq library to measure the translation rate of 8,000 newly designed 5'UTRs (Fig. 5a).

When using neuron activation to detect the presence of motifs in a sequence, *MTtrans* motifs manifested an overall consistency (59% of all 256 channels) in regulatory effect across MPRA datasets and FACS-seq dataset ( $p=0.0066$ , binomial test with  $k = 153$   $n = 256$ , Supplementary Fig. 6b).

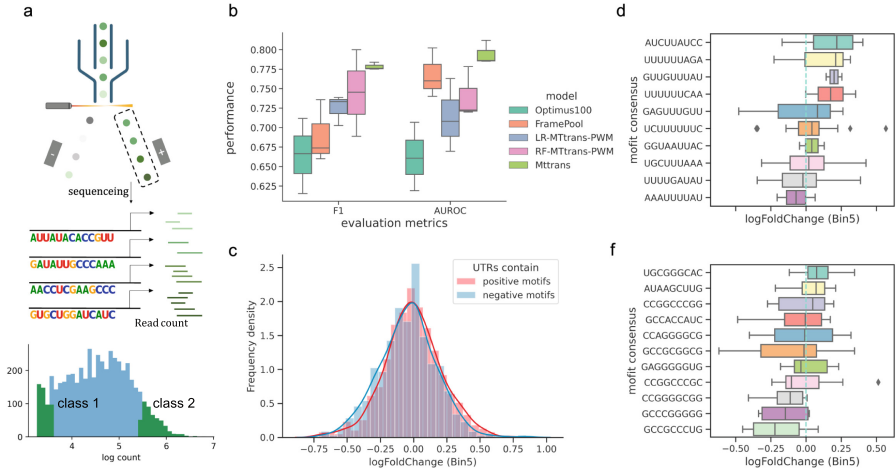
To test the extent to which the discovered motifs are predictive without the specific neural network, we used the PWMs derived from the shared encoder to score the strength of motifs. The scores of the 256 motifs were used as features to train logistic regression and random forest models to classify the top and bottom 10% UTRs of our FACS-seq dataset (Fig. 5ab). Using *MTtrans* motifs, LR and RF models succeeded in separating the two classes (average F1 0.714 & 0.740, average AUROC 0.725 & 0.745, respectively) and outperformed the deep neural network baselines *Optimus100* and *FramePool100* in F1-score. These results showed that the discovered motifs alone are robust and highly informative in modeling translation regulation.

Inspecting the learned parameters of LR and RF models can give us insights into the role of each motif in the FACS-seq dataset and we further identified a subset of important features (29 positive and 25 negative). We next tested these identified motifs in another independent FACS-seq dataset to evaluate their robustness. We curated another public FACS-seq library, GSE176581 [6], whose translation rate was represented by the enrichment of cells in the high GFP fluorescence bin (log fold change of Bin5). For motif detection, a motif is considered to occur in the sequence when its PWM matching score is higher than the defined quantile threshold of the score distribution (see Method). Collectively, UTRs with more positive motifs have higher translation rate than UTRs with more negative motifs (Fig. 5c,  $p$ -value =  $7.65 \times 10^{-4}$  for the threshold of 90% quantile, one-sided unpaired T-test), and the significance also holds for different threshold (Supplementary Fig. 6), indicating a robust regulatory direction of these motif features. For individual motifs, we specifically checked out sequences that only contained one motif and with a low average score on the opponent set (Fig. 5d and f). Only 10 positive and 11 negative motifs left meet the above criteria, and the majority of them maintained the same regulatory effect on the translation rate.

Overall, using two FACS-seq datasets, we demonstrated that the *MTtrans*-derived motifs are transferable to a new experimental system and can distinguish between low TR class and high TR class without the neural network. Two subsets of features, identified by PWM-based models, show a robust regulatory direction across two FACS-seq libraries. These results supported that *MTtrans* can indeed combine different datasets to yield biologically meaningful motifs.

## 4 Discussion

In this study, we demonstrate that our multi-task learning model *MTtrans*, is effective to predict mRNA translation rate using 5'UTR sequences for diverse experimental data. *MTtrans* works by treating each dataset as an individual task so that useful information across tasks can be underlined repeatedly. Importantly, we found that this framework enables the extraction of highly robust



**Fig. 5.** FACS experiment to validate the discovered motifs. **a** cells expressing EGFP coupled with designed 5'UTR are sorted by their EGFP fluorescence intensity, screened for high EGFP expression and then sequenced to identify the delivered 5'UTRs. The read count of the 5'UTRs can thus approximate its translation rate. 5'UTRs with the lowest translation rate were grouped as the negative class (class 1) and the positive class (class 2) for the highest one. **b** Result of binary cross-entropy for the above two classes. The box shows the performance of models for different train-test splitting random seeds. LR is the logistic regression built on the features scored by MTtrans-explained PWMs. RF stands for the random forest model built using the same features. **c** The distribution of translation rate (log fold-change in Bin5) for 5'UTRs on another FACS-seq library GSE176581. UTRs that enriched with the 29 identified positive motifs (red curve and bins), has higher translation rate than those containing the 25 negative motif sets (blue curve and bins) ( $p$ -value =  $7.65 \times 10^{-4}$  for threshold of 90% quantile, one-side unpaired T-test). **d** The translation rate of sequences from GSE176581 that contain only one motif among the selected positive feature set. **f** The translation rate of the sequence from GSE176581 that contains only one motif among the selected negative feature set.

sequence motifs that predict the increase or decrease of the translation rate. This is achieved by the hard-sharing architecture in our multi-task model.

Previous studies on transcription factor motif discovery usually can extract predictive features from the first convolutional layer [1, 31]. Koo et al. even propose an ambiguous pooling to enforce the motifs detection finished in the first layer [16]. Interestingly, in the context of translation rate prediction based on 5'UTR, it is necessary to look beyond the first layer to extract features. Furthermore, we found that the improvement does not result from the longer receptive field (Supplementary Fig. 2). As the deeper features are assembled from the lower layer patterns by convolution filters, translation rate prediction may thereby require better arrangement of the basal motifs to sense the sequence context. This might partially explain how the CNN model surpasses the  $k$ -mer-

linear model [25] because  $k$ -mer models lose the sequential relationship of the features despite a broader  $k$ -mer coverage. Future work should explore how to better extract features from different layers of neural networks for the prediction of transcription factor binding and translation rate.

A core premise of our work is that more robust and transferable features can be extracted from a model learned from diverse data sets generated from different experimental techniques and cell types. Although we have yielded robust regulatory motifs with maximally activated Seqlet (Fig. 4), there is still rich information in the model that is not fully explained, such as the motif interaction. Recent studies analyse the motif interaction in an instance-based way. Ziga et al. identified the cooperative TF binding interaction by permuting the spacing of two candidate motifs [4, 5]. Preserving the reading frame position of the extracted feature is necessary for the model to correctly identify the in-frame or out-of-frame motifs [15]. For *MTtrans*, we choose to employ the GRU layers in the task-specific tower because it is effective at handling the spatial arrangement of the motif detected by the last convolutional layer. Visualisation of the GRU layer, however, usually requires an extra architectural plugin like attention [27]. Although the task-specific syntax explanation of the GRU layers is not the focus of this work, future work can reveal how the essential elements are coordinated to predict the translation rate.

Overall, *MTtrans* provides a solution to extract the robust translation regulatory elements in the 5'UTR from a collection of related yet noisy systems. We expect this multi-task framework can be extended to learning the technique-invariant determinant for other sequence modelling questions to reveal the biologically meaningful signal from the sequence.

**Acknowledgements.** We thank Dr. Sung Chul Kwon for his helpful suggestions on transcript filtering and Dr. Chen Qiao for model training. We also thank Xinyi Lin and Yiming Chao for their feedback on data visualisation and pipeline testing.

This work was supported in part by AIR@InnoHK administered by Innovation and Technology Commission and the National Natural Science Foundation of China Excellent Young Scientists Fund (32022089). The work is also supported by the Centre for Oncology and Immunology Limited under the Health@InnoHK Initiative funded by the Innovation and Technology Commission, The Government of Hong Kong SAR, China.

**Availability of Data and Materials.** The code to re-implement *MTtrans* can be access from <https://github.com/holab-hku/MTtrans> and the FACS library is also available from Gene Expression Omnibus (GEO) under the accession of GSE201766.

## A Appendix

There is one additional file containing supplementary methods, supplementary Tables 1–2 and supplementary Figs. 1–8.

## References

1. Alipanahi, B., Delong, A., Weirauch, M.T., Frey, B.J.: Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* **33**(8), 831–838 (2015)
2. Andreev, D.E., et al.: Translation of 5′ leaders is pervasive in genes resistant to eif2 repression. *Elife* **4**, e03971 (2015)
3. Araujo, P.R., et al.: Before it gets started: regulating translation at the 5′ UTR. *Comp. Func. Genomics* **2012** (2012)
4. Avsec, Ž, et al.: Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nat. Genet.* **53**(3), 354–366 (2021)
5. Baltz, A.G., et al.: The mRNA-bound proteome and its global occupancy profile on protein-coding transcripts. *Mol Cell* **46**(5), 674–690 (2012)
6. Cao, J., et al.: High-throughput 5′ UTR engineering for enhanced protein production in non-viral gene therapies. *Nat. Commun.* **12**(1), 1–10 (2021)
7. Cuperus, J.T., et al.: Deep learning of the regulatory grammar of yeast 5′ untranslated regions from 500,000 random sequences. *Genome Res.* **27**(12), 2015–2024 (2017)
8. DeGrave, A.J., Janizek, J.D., Lee, S.I.: Ai for radiographic Covid-19 detection selects shortcuts over signal. *Nat. Mach. Intell.* **3**(7), 610–619 (2021)
9. Dvir, S., et al.: Deciphering the rules by which 5′-UTR sequences affect protein expression in yeast. *Proc. Natl. Acad. Sci.* **110**(30), E2792–E2801 (2013)
10. Geirhos, R., et al.: Shortcut learning in deep neural networks. *Nat. Mach. Intell.* **2**(11), 665–673 (2020)
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587 (2014)
12. Hsieh, A.C., et al.: The translational landscape of mTOR signalling steers cancer initiation and metastasis. *Nature* **485**(7396), 55–61 (2012)
13. Ingolia, N.T., Ghaemmaghami, S., Newman, J.R., Weissman, J.S.: Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. *Science* **324**(5924), 218–223 (2009)
14. Jackson, R.J., Hellen, C.U., Pestova, T.V.: The mechanism of eukaryotic translation initiation and principles of its regulation. *Nat. Rev. Mol. Cell Biol.* **11**(2), 113–127 (2010)
15. Karollus, A., Avsec, Ž, Gagneur, J.: Predicting mean ribosome load for 5′UTR of any length using deep learning. *PLoS Comput. Biol.* **17**(5), e1008982 (2021)
16. Koo, P.K., Eddy, S.R.: Representation learning of genomic sequence motifs with convolutional neural networks. *PLoS Comput. Biol.* **15**(12), e1007560 (2019)
17. Kozak, M.: An analysis of 5′-noncoding sequences from 699 vertebrate messenger RNAs. *Nucl. Acids Res.* **15**(20), 8125–8148 (1987)
18. Li, J.J., Chew, G.L., Biggin, M.D.: Quantitative principles of cis-translational control by general mRNA sequence features in eukaryotes. *Genome Biol.* **20**(1), 1–24 (2019)
19. Lin, J.C., Hsu, M., Tarn, W.Y.: Cell stress modulates the function of splicing regulatory protein RBM4 in translation control. *Proc. Natl. Acad. Sci.* **104**(7), 2235–2240 (2007)
20. Lotfollahi, M., et al.: Mapping single-cell data to reference atlases by transfer learning. *Nat. Biotechnol.* **40**(1), 121–130 (2022)

21. Noderer, W.L., et al.: Quantitative analysis of mammalian translation initiation sites by FACS-seq. *Mol. Syst. Biol.* **10**(8), 748 (2014)
22. Novakovsky, G., Saraswat, M., Fornes, O., Mostafavi, S., Wasserman, W.W.: Biologically relevant transfer learning improves transcription factor binding prediction. *Genome Biol.* **22**(1), 1–25 (2021)
23. Ray, D., et al.: A compendium of RNA-binding motifs for decoding gene regulation. *Nature* **499**(7457), 172–177 (2013)
24. Riba, A., et al.: Protein synthesis rates and ribosome occupancies reveal determinants of translation elongation rates. *Proc. Natl. Acad. Sci.* **116**(30), 15023–15032 (2019)
25. Sample, P.J., et al.: Human 5′ UTR design and variant effect prediction from a massively parallel translation assay. *Nat. Biotechnol.* **37**(7), 803–809 (2019)
26. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: *International Conference on Machine Learning*, pp. 3145–3153. PMLR (2017)
27. Vaswani, A., et al.: Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017)
28. Wein, N., et al.: Translation from a DMD exon 5 IRES results in a functional dystrophin isoform that attenuates dystrophinopathy in humans and mice. *Nat. Med.* **20**(9), 992–1000 (2014)
29. Weinberg, D.E., Shah, P., Eichhorn, S.W., Hussmann, J.A., Plotkin, J.B., Bartel, D.P.: Improved ribosome-footprint and mRNA measurements provide insights into dynamics and regulation of yeast translation. *Cell Rep.* **14**(7), 1787–1799 (2016)
30. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? arXiv preprint [arXiv:1411.1792](https://arxiv.org/abs/1411.1792) (2014)
31. Zeng, H., Edwards, M.D., Liu, G., Gifford, D.K.: Convolutional neural network architectures for predicting DNA-protein binding. *Bioinformatics* **32**(12), i121–i127 (2016)



# Computing Shortest Hyperpaths for Pathway Inference in Cellular Reaction Networks

Spencer Krieger<sup>1</sup>(✉) and John Kececioglu<sup>2</sup>

<sup>1</sup> Computational Biology Department, Carnegie Mellon University,  
Pittsburgh, PA 15213, USA

`skrieger@andrew.cmu.edu`

<sup>2</sup> Department of Computer Science, The University of Arizona,  
Tucson, AZ 85721, USA

`kece@cs.arizona.edu`

**Abstract.** Signaling and metabolic pathways, which consist of a series of reactions producing target molecules from source compounds, are cornerstones of cellular biology. The cellular reaction networks containing such pathways can be precisely modeled by *directed hypergraphs*, where each reaction corresponds to a hyperedge, directed from its set of reactants to its set of products. Given such a network represented by a directed hypergraph, inferring the most likely set of reactions that produce a given target from a given set of sources corresponds to finding a *shortest hyperpath*, which is NP-complete. The best methods currently available for shortest hyperpaths either offer no guarantee of optimality, or exclude hyperpaths containing cycles even though cycles are abundant in real biological pathways.

We derive a novel *graph-theoretic characterization* of hyperpaths, leveraged in a new formulation of the general shortest hyperpath problem as an integer linear program that for the first time handles hyperpaths containing cycles, and present a novel *cutting-plane algorithm* that can solve this integer program to optimality in practice. This represents a major advance over the best prior exact algorithm, which was limited to acyclic hyperpaths (and hence fails to find a solution for the many biological instances where all hyperpaths are in fact cyclic). In comprehensive experiments over thousands of instances from the standard NCI-PID and **Reactome** databases, we demonstrate that our cutting-plane algorithm quickly finds an *optimal hyperpath*, with a median running-time of under ten seconds and a maximum time of around thirty minutes, even on large instances with many thousands of reactions.

Source code implementing our cutting-plane algorithm for shortest hyperpaths in a new tool called **Mmunin** is available free for research use at <http://mmunin.cs.arizona.edu>.

---

S. Krieger—Research performed at the Department of Computer Science of the University of Arizona.



## 1 Introduction

Signaling and metabolic pathways are cornerstones of systems biology. They underly cellular communication, govern environmental response, and their perturbation has been implicated in the cause of disease [21]. Networks comprised of these pathways are traditionally represented as ordinary graphs [30,31], modeling each protein or molecule as a vertex, and each reaction by a collection of edges, directed from each reactant to each product. However, this does not faithfully model multiway reactions—ubiquitous in cellular processes—and shortest paths from these models are often not biologically meaningful [14,27].

Directed *hypergraphs* generalize ordinary graphs where an edge, now called a *hyperedge*, is directed from one set of vertices, called its *tail*, to another set of vertices, called its *head*. Hypergraphs have been used to model many cellular processes [7, 10, 11, 14, 23, 24, 26, 27, 33]. In particular, a biochemical reaction with multiple reactants—all of which must be present for the reaction to proceed—and multiple products—all of which are produced upon its completion—is correctly captured by a single hyperedge, directed from its set of reactants to its set of products. Despite hypergraphs affording more faithful models of reaction networks, the lack of practical algorithms has hindered their potential for properly representing and reasoning about molecular reactions.

Biologically, a typical metabolic or signaling pathway consists of a series of reactions synthesizing a set of target molecules—a key metabolite or transcription factor—from a set of source compounds—molecules available to the cell or activated membrane-bound receptors [2]. Computationally, finding the most efficient way to produce a set of target molecules from a set of available source compounds maps to the shortest hyperpath problem we consider here: Given a network of cellular reactions whose reactants and reactions are modeled by the vertices and weighted hyperedges of a directed hypergraph, together with a set of sources and a set of targets, find a hyperpath from the sources to the targets of minimum total weight. We briefly summarize prior work on related problems.

### Related Work

The two fundamental hypergraph models that have emerged for pathway inference are hyperpaths and factories (see [15] for a survey).

*Factories* are informally a set of reactions that produce the targets from the sources, while ensuring intermediate metabolites are not depleted. Unlike hyperpaths, a factory’s hyperedges are unordered, essentially running simultaneously. Current approaches find a factory producing the targets using either the fewest reactions (*min-edge*) or fewest sources (*min-source*). Cottret et al. [8] introduced the min-source factory problem and showed it is NP-hard, and Zarecki et al. [32] extended the problem to consider molecular weights of the sources. Methods from Acuña et al. [1] and Andrade et al. [3] enumerate all min-source factories either excluding or including stoichiometry. Krieger and Kececioglu [17] introduced the min-edge factory problem, showed it was NP-complete, incorpo-

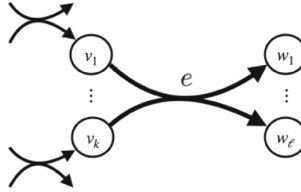
rated negative regulation into pathway inference for the first time, and solved the problem with a mixed-integer linear program that is fast in practice.

*Hyperpaths* were first studied in the field of algorithms (see the survey from Ausiello and Laura [4]). Italiano and Nanni [12] proved that finding a shortest source-sink hyperpath is NP-complete, even when hyperedges have a single head vertex. Gallo et al. [9] defined and explored special cases of hypergraphs and hyperpaths, including what they call a *B*-path (though see the correction of Nielsen and Pretolani [22]), which is essentially equivalent to our definition of hyperpath in Sect. 2. They showed the vertices reachable from a source vertex in a hypergraph can be found in time linear in the total size of the tail and head sets of all hyperedges, gave an efficient algorithm for a variant of shortest hyperpaths with a so-called additive cost function, and proved that finding a minimum cut in a hypergraph is NP-complete. Carbonell et al. [6] gave an efficient algorithm to find a source-sink hyperpath if one exists—irrespective of its length—and proved that finding any hyperpath that must contain a specified set of hyperedges is NP-complete. Ritz et al. [25, 26] were the first to solve the shortest *acyclic* hyperpath problem by formulating it as a mixed-integer linear program (MILP)—later extended by Schwob et al. [29] to include time-dependence among reactions—and showed that in the context of signaling networks, optimal acyclic hyperpaths can be found even for large cell-signaling hypergraphs. Their formulation does not extend to allow cycles, which are common in pathway databases (see experimental results in Sect. 4), and are often caused by feedback loops or by the components of protein complexes during assembly and disassembly. Krieger and Kececioğlu [16, 18] gave the first heuristic for the general shortest hyperpath problem allowing cycles, proved it finds optimal hyperpaths for the special case of singleton-tail hypergraphs, gave a tractable hyperpath enumeration algorithm, and verified that the heuristic is close to optimal on all instances from the standard pathway databases by leveraging the enumeration algorithm.

## Our Contributions

In contrast to prior work, we provide an exact algorithm for the general shortest hyperpath problem, allowing cycles. Note that cycles appear in pathway databases—often as feedback loops—so an algorithm that handles cycles is vital for the adoption of hypergraph models of cellular reaction networks. We formulate this problem as an integer linear program (ILP) and develop a cutting-plane algorithm that can quickly solve this ILP to optimality in practice. More specifically, we make the following contributions.

- We derive a new *graph-theoretic characterization* of hyperpaths in terms of source-sink cuts, that captures fully-general hyperpaths with cycles.
- We leverage this characterization to obtain the first *integer linear programming formulation* of the general shortest hyperpath problem. This cut-based ILP formulation has an exponential number of constraints, however, and cannot be solved directly.



**Fig. 1.** A hyperedge  $e$  with  $\text{tail}(e) = \{v_1, \dots, v_k\}$  and  $\text{head}(e) = \{w_1, \dots, w_\ell\}$ . To use  $e$  in a hyperpath  $P$ , every vertex  $v_i \in \text{tail}(e)$  must have a preceding hyperedge  $f$  in  $P$  with  $v_i \in \text{head}(f)$ .

- Nevertheless, we show we can solve this ILP on real biological instances with a practical *cutting-plane algorithm* that computes over a small subset of the constraints, while guaranteeing an optimal solution to the full ILP.
- Our cutting-plane algorithm is typically *fast in practice*, finding optimal hyperpaths with a median running time under 10s, and a maximum running time around 30 min, as measured through comprehensive experiments on thousands of instances from standard reaction databases.
- We demonstrate the strength of hyperpath models for pathway inference by showing the cutting-plane algorithm accurately recovers annotated pathways, with a median overlap score of over 95%, while recovered hyperedges outside the annotated pathway provide evidence for possible cross-talk.

A preliminary implementation of the cutting-plane algorithm in a new tool called **Mmunin** (short for “integer-linear-programming-based cutting-plane algorithm for shortest source-sink hyperpaths”) is available free for non-commercial use at <http://mmunin.cs.arizona.edu>.

## Plan of the Paper

The next section defines the computational problem of shortest hyperpaths. Section 3 gives a new graph-theoretic characterization of hyperpaths that leads to the first formulation of shortest hyperpaths as an *integer linear program* for fully-general hyperpaths with cycles, as well as a *cutting-plane algorithm* for solving it to optimality in practice. Section 4 compares our algorithm to alternate hyperpath methods through comprehensive experiments over the two standard pathway databases, and demonstrates on concrete biological examples that we can accurately recover known annotated pathways. Finally, Sect. 5 concludes.

## 2 Shortest Hyperpaths in Directed Hypergraphs

A hypergraph is a generalization of an ordinary graph, where an edge, instead of touching two vertices, now connects two subsets of vertices. Formally, a directed *hypergraph* is a pair  $(V, E)$ , where  $V$  is a set of vertices, and  $E$  is a set of directed *hyperedges*. Each hyperedge  $e \in E$  is an ordered pair  $(X, Y)$ , where both  $X, Y \subseteq V$  are non-empty vertex subsets. Edge  $e$  is directed from set  $X$  to

set  $Y$ . We call  $X$  the *tail* of  $e$ , and  $Y$  the *head* of  $e$ , and refer to them by the functions  $\text{tail}(e) = X$  and  $\text{head}(e) = Y$ . We also refer to the in- and out-edges of a vertex  $v$  by  $\text{in}(v) = \{e \in E : v \in \text{head}(e)\}$  and  $\text{out}(v) = \{e \in E : v \in \text{tail}(e)\}$ . Figure 1 shows a directed hyperedge.

In ordinary directed graphs, a path from a vertex  $s$  to a vertex  $t$  is a sequence of edges starting from  $s$  ending at  $t$ , where for consecutive edges  $e$  and  $f$  in the sequence, the preceding edge  $e$  must enter the vertex that the following edge  $f$  leaves. We say  $t$  is reachable from  $s$  when there is such a path from  $s$  to  $t$ .

In generalizing these notions to directed hypergraphs, the conditions both for when a hyperedge can follow another in a hyperpath, and when a vertex is reachable from another, become more involved. A hyperpath is again a sequence of hyperedges, but now for hyperedge  $f$  in a hyperpath, for every vertex  $v \in \text{tail}(f)$ , there must be some hyperedge  $e$  that precedes  $f$  in the hyperpath for which  $v \in \text{head}(e)$ . Reachability is captured by the following notion of superpath.

**Definition 1 (Superpath).** *In a directed hypergraph  $(V, E)$ , an  $s, t$ -superpath, for vertices  $s, t \in V$ , is an edge subset  $F \subseteq E$  such that the hyperedges of  $F$  can be ordered  $e_1, e_2, \dots, e_k$ , where*

- (i)  $\text{tail}(e_1) = \{s\}$ ,
- (ii) for each  $1 < i \leq k$ ,

$$\text{tail}(e_i) \subseteq \{s\} \cup \bigcup_{1 \leq j < i} \text{head}(e_j),$$

- (iii) and  $t \in \text{head}(e_k)$ .

For an  $s, t$ -superpath, we call  $s$  its source vertex and  $t$  its sink vertex, and we say  $t$  is reachable from  $s$ .  $\square$

This definition of superpath is equivalent to the notion of B-connectivity from the literature, but is more explicit, and more amenable to formulation as an integer linear program in Sect. 3.

We can now define hyperpaths in terms of superpaths. Recall that a set  $S$  is *minimal* with respect to some property  $X$  if  $S$  satisfies  $X$ , but no proper subset of  $S$  satisfies  $X$ .

**Definition 2 (Hyperpath).** *An  $s, t$ -hyperpath is a minimal  $s, t$ -superpath.  $\square$*

In other words, a hyperpath  $P$  is a superpath for which removing any edge  $e \in P$  leaves a subset  $P - \{e\}$  that is no longer a superpath.

We say a hyperpath  $P$  contains a *cycle* if, for every ordering  $e_1, \dots, e_k$  of its hyperedges satisfying properties (i)–(iii) in the definition of superpath,  $P$  contains some hyperedge  $f$  with a vertex in  $\text{head}(f)$  that also occurs in  $\text{tail}(e)$  for an earlier hyperedge  $e$  in the ordering. While in ordinary graphs a minimal  $s, t$ -path can never contain a cycle, in hypergraphs an  $s, t$ -hyperpath, which by definition is minimal, can in fact contain cycles.

Notice that when the hyperedges of a hypergraph all have real-valued weights that are strictly positive, then an  $s, t$ -superpath of minimum total weight must be minimal. (If it is not minimal, deleting an edge will give another superpath of strictly smaller weight, contradicting its optimality.) Hence for positive edge weights, a minimum weight superpath is a minimum weight hyperpath. This leads to the following definition of the shortest hyperpaths problem.

For an edge weight function  $\omega(e)$ , we extend  $\omega$  to edge subsets  $F \subseteq E$  by  $\omega(F) := \sum_{e \in F} \omega(e)$ .

**Definition 3 (Shortest Hyperpaths).** *The Shortest Hyperpaths problem is the following. Given a directed hypergraph  $(V, E)$ , a positive edge weight function  $\omega : E \rightarrow \mathcal{R}^+$ , source  $s \in V$  and sink  $t \in V$ , find*

$$\operatorname{argmin}_{F \subseteq E} \left\{ \omega(F) : F \text{ is an } s, t\text{-superpath} \right\}. \quad (1)$$

*This  $s, t$ -superpath of minimum weight is a shortest  $s, t$ -hyperpath.  $\square$*

For uniform weights  $\omega(e) = 1$ , this finds a series of the *fewest possible reactions* that produce  $t$  from  $s$ , where for each reaction in the series, all its input reactants are produced as output products of earlier reactions in the series.

We note that Shortest Hyperpaths with a single source and sink can also capture more general versions of the problem with *multiple sources* and *multiple sinks*, as follows. To find a hyperpath that starts from a *set* of sources  $S \subseteq V$ , simply add a new source vertex  $s$  to the hypergraph together with a single hyperedge  $(\{s\}, S)$  of zero weight, and equivalently find a hyperpath from the single source  $s$ . To find a hyperpath that reaches *all* vertices in a *set* of sinks  $T \subseteq V$ , add a new sink vertex  $t$ , a zero-weight hyperedge  $(T, \{t\})$ , and equivalently find a hyperpath to the single sink  $t$ . To find a hyperpath that reaches *some* vertex in a set of sinks  $T \subseteq V$ , add new sink vertex  $t$ , zero-weight hyperedges  $(\{v\}, \{t\})$  from all  $v \in T$ , and again equivalently find a hyperpath to the single sink  $t$ . Thus versions of shortest hyperpaths with multiple sources and sinks can be reduced to the problem above with a single source and sink.

Shortest Hyperpaths is NP-complete [25] (even for acyclic hypergraphs), so it is unlikely we can efficiently compute shortest hyperpaths in the worst-case. The next section presents a new formulation of Shortest Hyperpaths as an integer linear programming problem, which allows us to leverage techniques for solving integer linear programs to quickly find shortest hyperpaths in practice, even for large reaction networks.

### 3 Computing Hyperpaths by Integer Programming

We now formulate Shortest Hyperpaths as a discrete optimization problem known as an integer linear program, using a characterization of superpaths in terms of cuts. This integer linear program is the first formulation that handles fully general hyperpaths that may contain cycles.

### 3.1 Characterizing Superpaths via Cuts

We can give a clean characterization of superpaths—that captures fully general superpaths containing cycles—in terms of cuts.

An  $s, t$ -cut of a hypergraph is a bipartition  $(C, \overline{C})$  of its vertices  $V$ , for non-empty subsets  $C \subseteq V$  and  $\overline{C} := V - C$ , where source  $s \in C$  and sink  $t \in \overline{C}$ . We call  $C$  the *source side* and  $\overline{C}$  the *sink side* of the cut, and often refer to a cut by just specifying its source side  $C$ .

A hyperedge  $e$  crosses  $s, t$ -cut  $C$  iff  $\text{tail}(e) \subseteq C$  and  $\text{head}(e) \cap \overline{C} \neq \emptyset$ . In other words, for a hyperedge to cross a cut, all its tail vertices must be on the source side, while at least one head vertex must be on the sink side. We say an edge subset  $F \subseteq E$  crosses an  $s, t$ -cut iff at least one hyperedge  $e \in F$  crosses the cut.

The following characterization theorem for superpaths is the key that will enable us to find shortest hyperpaths via integer linear programming.

**Theorem 1 (Characterizing Superpaths).**  *$F$  is an  $s, t$ -superpath if and only if  $F$  crosses every  $s, t$ -cut.*

*Proof.* To prove the forward implication, take an ordering of the hyperedges of  $s, t$ -superpath  $F$  that satisfies the definition of superpath, and an arbitrary  $s, t$ -cut  $C$ . In the ordering of  $F$ , consider the first hyperedge  $e$  such that  $\text{head}(e) \cap \overline{C}$  is nonempty. (Such an edge  $e$  must exist, as  $F$  reaches  $t \in \overline{C}$ .) We claim that  $\text{tail}(e) \subseteq C$ , which can be shown by proving  $\bigcup_{f \in F: f \text{ precedes } e} \text{head}(f) \subseteq C$ , using induction over the ordering of  $F$ . Thus edge  $e \in F$  crosses cut  $C$ , so  $F$  crosses  $C$  as well.

For the reverse implication, we prove the contrapositive. Suppose  $F$  is not an  $s, t$ -superpath. Collect the set  $R$  of all vertices reachable from  $s$  in  $F$ . While  $s \in R$ , notice  $t \notin R$  (since otherwise  $F$  reaches  $t$  from  $s$ , contradicting that  $F$  is not an  $s, t$ -superpath). Thus  $(R, \overline{R})$  is an  $s, t$ -cut.  $F$  does not cross cut  $R$  (since  $e \in F$  crossing  $R$  would contradict that  $R$  holds all vertices reachable from  $s$  in  $F$ ).  $\square$

### 3.2 Representing Superpaths by Linear Inequalities

We now formulate Shortest Hyperpaths as an integer linear programming problem. In general, an integer linear program (ILP) is a mathematical optimization problem over integer-valued variables, that maximizes a linear function of these variables, subject to constraints that are linear inequalities in the variables. The key to the formulation is to represent the set of all  $s, t$ -superpaths in a hypergraph  $(V, E)$  by linear inequalities.

The *variables* of our ILP encode the hyperedges in a superpath  $F$ . For every hyperedge  $e \in E$ , there is a variable  $x_e$ , where  $x_e \in \{0, 1\}$ . An assignment of values to these variables encodes a superpath  $F$  by  $x_e = 1$  iff  $e \in F$ . We represent the collection of all variables in the ILP by a vector  $x = (x_e)_{e \in E}$ , where  $x \in \{0, 1\}^{|E|}$ .

The *constraints* of the ILP ensure an assignment of values to the variables actually encodes an  $s, t$ -superpath. The domain  $\mathcal{D}$  of the ILP is all assignments

of values to variables  $x$  that satisfy the constraints. For our ILP, the domain is

$$\mathcal{D} := \left\{ x \in \{0, 1\}^{|E|} : \forall s, t\text{-cuts } C \quad \sum_{e \in E : e \text{ crosses } C} x_e \geq 1 \right\}. \quad (2)$$

This has a constraint for every  $s, t$ -cut of the hypergraph, which is a linear inequality in the variables  $x_e$ . Notice that this inequality for a cut  $C$  is satisfied iff at least one hyperedge  $e$  crossing  $C$  has  $x_e = 1$ . Equivalently, the set  $F \subseteq E$  encoded by  $x$  must contain at least one hyperedge  $e \in F$  crossing  $C$ . Thus assignments  $x \in \mathcal{D}$  encode edge subsets  $F$  that cross every  $s, t$ -cut.

Consequently, by Theorem 1, the domain  $\mathcal{D}$  of the ILP in Eq. (2) is exactly the set of all  $s, t$ -superpaths in the hypergraph.

The *objective function* of the ILP is to minimize  $\sum_{e \in E} \omega(e) x_e$ , for fixed edge weights  $\omega$ , which is a linear function of the variables  $x$ . For  $x \in \mathcal{D}$ , the value of this objective function is the total weight of the hyperedges in superpath  $F$  encoded by  $x$ . We can write this objective function as a dot product  $\omega \circ x$ , where  $\omega$  is now a vector of edge weights.

Finally, our *integer linear program* is to compute  $\operatorname{argmin}_{x \in \mathcal{D}} \{\omega \circ x\}$ . Since domain  $\mathcal{D}$  is all  $s, t$ -superpaths, this is equivalent to the definition of the Shortest Hyperpaths problem, so a solution to this ILP is a shortest  $s, t$ -hyperpath.

### 3.3 Solving the Integer Program by a Cutting Plane Algorithm

For a hypergraph of  $n$  vertices and  $m$  hyperedges, the integer linear program given above has  $m$  variables and  $\Theta(2^n)$  constraints (corresponding to the number of  $s, t$ -cuts). Thus for a large hypergraph, we cannot even feasibly write down the corresponding ILP, due to its exponentially-many constraints. Nevertheless, we can actually compute optimal solutions to this full ILP in practice, even for large hypergraphs, using an approach known as a cutting-plane algorithm.

A *cutting-plane algorithm* computes over a subset of the constraints of the full integer linear program, and solves a series of less-constrained problems, stopping once it detects it has an optimal solution to the full ILP. The key to a cutting-plane algorithm is an efficient *separation algorithm*, which for a given solution  $x$  to the current ILP, reports whether  $x$  satisfies all constraints of the full ILP, and if  $x$  does not, returns a constraint violated by  $x$ . (This violated constraint is a hyperplane, called a cutting plane, that separates  $x$  from the domain of the full ILP.) For our above ILP for Shortest Hyperpaths, this proceeds as follows.

- (1) Let  $\mathcal{I}$  be an initial set of inequalities, containing a subset of the inequalities from the full ILP, and let  $\mathcal{S} := \mathcal{I}$  be the current set of inequalities.
- (2) Solve the ILP restricted to the inequalities in  $\mathcal{S}$ , and let  $x^*$  be the optimal solution to this current ILP.
- (3) Run the separation algorithm to efficiently find an  $s, t$ -cut  $C$  that is not crossed by the hyperedges  $e$  with  $x_e^* = 1$ , if such a cut exists.
- (4) If the separation algorithm found a cut  $C$  not crossed by  $x^*$ , add the new cut-inequality given by  $C$  to set  $\mathcal{S}$ , and go back to Step (2).

- (5) Otherwise, the hyperedges in  $x^*$  cross every  $s, t$ -cut. Halt and output the current solution  $x^*$ .

This starts with an initial set of inequalities  $\mathcal{I}$ , and adds inequalities to this set as it finds cuts that solutions to prior ILPs do not cross.

The above cutting-plane algorithm outputs an optimal solution to the full ILP, though it works with only a subset of its constraints. Note that when the algorithm outputs its final solution  $x^*$  at Step (5), which crosses every cut,  $x^*$  encodes an  $s, t$ -superpath (by Theorem 1). Furthermore, as  $x^*$  is an optimal solution for a less-constrained ILP that is a minimization problem, its objective function value is a lower bound on the weight of an optimal solution to the full ILP. Hence  $x^*$  is a minimum-weight  $s, t$ -superpath, or equivalently, a shortest  $s, t$ -hyperpath.

The next section specifies the initial inequalities  $\mathcal{I}$  used in practice, which are crucial to its success. Then Sect. 3.5 presents our efficient separation algorithm, which finds multiple violated inequalities that are all added to the current ILP. As demonstrated in Sect. 4, this cutting-plane algorithm can find optimal hyperpaths even for large instances from real cellular reaction networks.

### 3.4 Strengthening the Initial Integer Program

We can markedly reduce the number of iterations of the cutting-plane algorithm by seeding it with a strong set of initial inequalities  $\mathcal{I}$ . We start with  $\mathcal{I}$  containing both the structure-based and distance-based inequalities that we describe next.

**Structure-Based Inequalities.** We define three classes of inequalities, based on structural properties of hyperpaths.

The *tail-covering inequalities* ensure that for any hyperedge chosen by a solution to the ILP, its tail set is covered by the head sets of other chosen hyperedges (corresponding to condition (ii) in Definition 1 for a superpath). More formally, for every hyperedge  $e \in E$ , and every vertex  $v \in \text{tail}(e) - \{s\}$ , we have the inequality,  $\sum_{f \in \text{in}(v)} x_f \geq x_e$ .

The *head-hitting inequalities* ensure that for a hyperedge  $e$  chosen by a solution, its head intersects (or “hits”) the tail of another chosen hyperedge (following from the minimality condition in Definition 2 for a hyperpath, since otherwise  $e$  can be safely trimmed while maintaining reachability). More formally, for every hyperedge  $e \in E$  with  $t \notin \text{head}(e)$ , we have the inequality,  $(\sum_{f \neq e : \text{head}(e) \cap \text{tail}(f) \neq \emptyset} x_f) \geq x_e$ .

The *target-production inequality* ensures that target  $t$  is reached by hyperedges chosen by the solution (corresponding to condition (iii) in Definition 1 for an  $s, t$ -superpath). More formally,  $\sum_{f \in \text{in}(t)} x_f \geq 1$ .

Together, these structure-based inequalities drive the solution found by the cutting-plane algorithm toward having the connected structure of a hyperpath (whereas without them, the solutions over many iterations tend to be disconnected hyperedges that cross the current set of cuts). Adding these inequalities to the initial ILP dramatically reduces the number of iterations of the cutting-plane algorithm, greatly improving its running time.



**Distance-Based Inequalities.** We first show that for ordinary graphs, there is a small subset of constraints from our original ILP, given by what we call distance-based cuts, that we can efficiently find and that guarantee that the ILP solved on just these distance-based inequalities has objective function value equal to the shortest path length. We then generalize these inequalities to hypergraphs.

*Ordinary Graphs.* For an ordinary directed graph with source  $s$  and sink  $t$  reachable from  $s$ , let  $D(v)$  be the length of a shortest  $s, v$ -path for every vertex  $v$  reachable from  $s$ . Over these reachable vertices  $v$  other than  $s$  with distance  $D(v) \leq D(t)$ , let  $d_1 < \dots < d_k$  be the sorted set of their unique distances  $D(v)$ . Define a sequence of  $s, t$ -cuts  $C_1 \subset \dots \subset C_k$  associated with these unique distances, for  $1 \leq i \leq k$ , by

$$C_i := \{s\} \cup \left\{ v \in V : D(v) < d_i \right\}. \quad (3)$$

Finally, denote the family of these  $s, t$ -cuts by  $\mathcal{C} := \{C_1, \dots, C_k\}$ , which we call the *distance-based cuts*.

The following theorem implies that for ordinary graphs, if we solve our original ILP just over inequalities corresponding to these distance-based cuts—which for a graph with  $n$  vertices is an ILP with less than  $n$  inequalities—then the objective function value of the optimal solution to this small ILP will in fact be the length of a shortest  $s, t$ -path.

**Theorem 2 (Distance-Based Cuts Suffice for Ordinary Graphs).** *In an ordinary graph, let  $F$  be an edge set that crosses every cut in the family  $\mathcal{C}$  of distance-based  $s, t$ -cuts. Then  $F$  has total weight  $\omega(F)$  that is at least the length  $D(t)$  of a shortest  $s, t$ -path.*

*Proof.* We show by induction that the weight of an edge set crossing cuts  $C_1, \dots, C_i$  is at least  $d_i$ , for all  $1 \leq i \leq k$ . Since  $d_k = D(t)$ , this proves the theorem.

For the basis with  $i = 1$ , consider cut  $C_1 = \{s\}$ , and let  $e = (s, v)$  be a minimum-weight edge leaving  $s$ . Consider any set  $F$  crossing  $C_1$ , which must contain an edge  $f$  leaving  $s$ . Since we have that  $\omega(F) \geq \omega(f) \geq \omega(e) = D(v) = d_1$ , the basis holds.

For the inductive step with  $i > 1$ , let  $F$  be any edge set that crosses cuts  $C_1, \dots, C_i$ . Set  $F$  must contain an edge  $f = (x, y)$  that crosses cut  $C_i$ . Notice that  $D(x) < d_i$ , which implies  $D(x) = d_j$  for some  $j < i$ . Let  $F' \subseteq F - \{(x, y)\}$  be the subset of  $F$  that crosses cuts  $C_1, \dots, C_j$ . We have

$$\begin{aligned} \omega(F) &\geq \omega(F') + \omega(f) \\ &\geq D(x) + \omega(f) \end{aligned} \quad (4)$$

$$\geq D(y) \quad (5)$$

$$\geq d_i, \quad (6)$$

where inequality (4) follows from the inductive hypothesis on  $j < i$ , inequality (5) follows from the fact that  $D(x) + \omega(f) = D(x) + \omega(x, y) \geq D(y)$ , and inequality (6) follows from the definitions of edge  $(x, y)$  and cut  $C_i$ . Thus the induction holds.  $\square$

Notice that a shortest  $s, t$ -path crosses every cut in family  $\mathcal{C}$ . Hence a consequence of Theorem 2 is that, for the optimal solution to the ILP whose inequalities are just the cut constraints given by the distance-based cuts  $\mathcal{C}$ , its objective function value is the length of a shortest  $s, t$ -path.

The number of cuts in family  $\mathcal{C}$  is less than the number of vertices. Moreover, Dijkstra's single-source shortest-path algorithm computes distance  $D(v)$  for every vertex  $v$  reachable from  $s$  with distance at most  $D(t)$ . Thus for an ordinary graph with  $n$  vertices and  $m$  edges, we can find the *distance-based inequalities* given by cuts  $\mathcal{C}$ , which constitute less than  $n$  inequalities, in the same time as running Dijkstra's algorithm, namely  $O(m + n \log n)$  time.

*Generalizing to Hypergraphs.* To generalize the distance-based cuts  $\mathcal{C}$  to hypergraphs, we need both a measure of distance to a vertex in a hypergraph, and a way to efficiently compute this measure. While there does not appear to be any natural distance measure on vertices for shortest hyperpaths that corresponds to  $D(v)$  in ordinary graphs, we can define a generalized vertex distance as follows.

For a hyperedge  $e$ , let an  $s, e$ -superpath be a superpath from source  $s$  that reaches all vertices in  $\text{tail}(e)$ , and define an  $s, e$ -hyperpath to be a minimal  $s, e$ -superpath. For a hyperedge  $e$ , let its *tail-distance*  $D(\text{tail}(e))$  be the length of a shortest  $s, e$ -hyperpath. Finally, for a vertex  $v$  in a hypergraph, we can define its *vertex-distance* from source  $s$  to be,  $D(v) := \min_{e \in \text{in}(v)} \{D(\text{tail}(e)) + \omega(e)\}$ .

Note that computing tail-distances  $D(\text{tail}(e))$  for hyperedges is at least as hard as Shortest Hyperpaths, so computing the above vertex-distances  $D(v)$  is unfortunately NP-complete as well.

To make this practical, we run the efficient hyperpath heuristic of Krieger and Kececioglu [16, 18], which computes estimated tail-distances  $\tilde{D}(\text{tail}(e))$  for all hyperedges  $e$  that are reachable from source  $s$ . We then apply these tail-distance estimates in the above definition of vertex distance, to obtain efficiently-computable estimated vertex-distances  $\tilde{D}(v)$ .

Using these vertex-distance estimates  $\tilde{D}(v)$ , and their unique estimated vertex-distances  $\tilde{d}_1 < \dots < \tilde{d}_k$ , we can directly generalize our prior distance-based cuts  $C_1, \dots, C_k$ , defined by (3), to hypergraphs. This yields the *distance-based inequalities* that our cutting-plane algorithm starts from in its initial set  $\mathcal{I}$ .

### 3.5 A Separation Algorithm Leveraging Distance-Based Cuts

The cutting-plane algorithm starts with inequalities  $\mathcal{I}$ , where  $\mathcal{I}$  contains the structure-based inequalities, as well as the distance-based inequalities from the family of cuts  $\mathcal{C}$ . Since the solution  $x^*$  to the current ILP crosses all cut-inequalities in  $\mathcal{S} \supseteq \mathcal{I}$  with its *active hyperedges*  $e$  where  $x_e^* = 1$ , solution  $x^*$  already crosses every cut in  $\mathcal{C}$ . To find new cuts not crossed by  $x^*$ , the separation algorithm considers every  $s, t$ -cut  $C \in \mathcal{C}$ , and enlarges its source-side  $C \supseteq \{s\}$ , called *source-augmentation*, or enlarges its sink-side  $\bar{C} = V - C \supseteq \{t\}$ , called *sink-augmentation*, to obtain a new cut  $\hat{C}$  not crossed by  $x^*$ .

Source-augmentation of  $C \in \mathcal{C}$ , to obtain  $\hat{C} \supset C$  not crossed by  $x^*$ , proceeds as follows. Recall that hyperedge  $e$  crosses  $C$  if  $\text{tail}(e) \subseteq C$  but  $\text{head}(e) \not\subseteq C$ .

For a given active hyperedge  $e$  that crosses  $C$ , we enlarge  $C$  to form a new cut  $\widehat{C} = C \cup \text{head}(e)$ , which is the minimal enlargement of  $C$  that  $e$  no longer crosses. We then repeat this process on  $\widehat{C}$  for every active hyperedge that crosses it, until we obtain a final cut  $\widehat{C}$  that no active hyperedge crosses. (Here  $\widehat{C}$  could grow until it includes sink  $t$ , which makes it no longer an  $s, t$ -cut, in which case there is no source-augmentation of  $C$  that  $x^*$  does not cross.) This process actually finds the cut  $\widehat{C} \supset C$  of minimum size  $|\widehat{C}|$  that  $x^*$  does not cross.

Since the cut  $C = \{s\}$  is one of the distance-based cuts in  $\mathcal{C}$ , and source-augmentation of this trivial cut yields  $\widehat{C} \supset \{s\}$  consisting of all vertices reachable from  $s$  along active hyperedges, by the proof of Theorem 1 this separation algorithm is guaranteed to find a cut not crossed by  $x^*$  whenever one exists.

Sink-augmentation of  $C \in \mathcal{C}$ , to obtain  $\widehat{C} \subset C$  not crossed by  $x^*$ , proceeds as follows. For a given active hyperedge  $e$  that crosses  $C$ , we enlarge  $\overline{C}$  by moving one vertex  $v \in \text{tail}(e) - \{s\}$  from  $C$  to its sink-side  $\overline{C}$ , yielding new cut  $\widehat{C} = C - \{v\}$ . This is a minimal enlargement of  $\overline{C}$  that  $e$  no longer crosses. Of course, this may cause new active hyperedges to now cross  $\widehat{C}$  that did not before. To reduce the number of new edges crossing  $\widehat{C}$ , we exploit the freedom in picking  $v$  by greedily choosing the  $v \in \text{tail}(e) - \{s\}$  that causes the fewest active hyperedges to newly cross  $\widehat{C}$ . (Once an active hyperedge  $e$  crossing  $\widehat{C}$  has  $\text{tail}(e) = \{s\}$ , this fails to find a sink-augmentation of  $C$  not crossed by  $x^*$ .) We repeat this process on  $\widehat{C}$  for every active hyperedge crossing it, until we obtain a final cut  $\widehat{C} \subset C$  that  $x^*$  does not cross.

This separation algorithm can find up to  $2k$  inequalities that are violated by  $x^*$ , where  $k = |\mathcal{C}|$  is the number of distance-based cuts, all of which are added to the current set  $\mathcal{S}$  by the cutting-plane algorithm. For a hypergraph of size  $\ell = \sum_{e \in E} (|\text{tail}(e)| + |\text{head}(e)|)$ , this separation algorithm can be implemented to run in  $O(k^2 \ell)$  time.

## 4 Experimental Results

We present experimental results with our implementation of the cutting-plane algorithm, named `Mmunin` [20], that show it can find optimal hyperpaths in large real-world cellular reaction networks quickly, often in less than 10s. We first give details of our experimental setup, describing our datasets and implementation. We then show, through comprehensive experiments over all source-sink instances from the two standard reaction databases in the literature, how `Mmunin` surpasses both the state-of-the-art heuristic for general shortest hyperpaths [16, 18], and the state-of-the-art exact algorithm for shortest acyclic hyperpaths [25]. Finally, we discuss an illustrative biological example, and analyze how well `Mmunin` recovers known biological pathways.

### 4.1 Experimental Setup

We first briefly describe our datasets and how we transform them into hypergraphs, and then give details on our implementation, including two modifications that further improve running time.

**Table 1.** Dataset summaries

	NCI-PID		Reactome	
Pathways	213		2,516	
Vertices	9,009		20,458	
Hyperedges	8,456		11,802	
Sources	3,200		8,296	
Targets	2,636		5,066	
Reachable targets	2,220		2,432	
	mean	max	mean	max
Tail size	1.9	10	2.4	26
Head size	1.1	5	1.6	28
In-degree	1.0	323	0.9	1,056
Out-degree	1.7	326	1.4	1,167
Doubly-reachable set	756	1,836	929	1,725

**Datasets and Preparation.** We prepared instances from two benchmark datasets, called NCI-PID and Reactome, following the hypergraph construction protocol of Ritz et al. [25] and Krieger and Kececioğlu [18]. The NCI-PID dataset aggregates all pathways from NCI-PID [28], while the Reactome dataset aggregates all pathways from Reactome [13]. To build a hypergraph from each dataset, we map each protein or small molecule to a vertex, and each reaction to a hyperedge, with reactants and positive regulators in the tail, and products in the head. All hyperedges are given unit weight, even though the cutting-plane algorithm handles general weights, as NCI-PID is missing reaction rates for many reactions. Table 1 gives summaries of these hypergraphs. The NCI-PID hypergraph has 9,009 vertices and 8,456 hyperedges, while the Reactome hypergraph has 20,458 vertices and 11,802 hyperedges. For all instances, we create a super-source  $s$  and add a single hyperedge  $e$ , where  $\text{tail}(e) = \{s\}$  and  $\text{head}(e)$  contains all vertices with no in-edge, which we call *sources*. For each individual instance, we create a sink  $t$  and connect it by an ordinary graph edge  $(v, t)$  to one vertex  $v$  with no out-edges, which we call a *target*. Considering all possible targets  $v$ , generates 2,636 instances from NCI-PID, and 5,066 instances from Reactome.

**Implementation.** The cutting-plane algorithm and its separation algorithm are all implemented in Python 3.8, comprising around 2,000 lines of code. All procedures are implemented as described earlier, with a few exceptions. First, we use a procedure from `Hhugin` to compute the doubly-reachable subgraph  $H$ , which contains only those hyperedges from the input hypergraph  $G$  that can possibly be in any  $s, t$ -hyperpath. Next, the initial distance-based inequalities require approximate tail-distances from `Hhugin`, but to improve running time, the cutting-plane algorithm begins with only the distance-based inequality given by cut  $C = \{s\}$ . We begin execution of `Hhugin` and the cutting plane algorithm in

**Table 2.** Suboptimality of alternate hyperpath methods

	NCI-PID		Reactome	
	median	max	median	max
Reachable instances		2,220		2,432
AcycMILP suboptimal		38		22
Hhugin suboptimal		23		0
Mmunin suboptimal		(none)		(none)
	median	max	median	max
AcycMILP path-length difference-from-optimal	( $\infty$ )	( $\infty$ )	( $\infty$ )	( $\infty$ )
Hhugin path-length difference-from-optimal	1	6	0	0

**Table 3.** Performance of Mmunin

Instances	NCI-PID		Reactome	
	2,220		2,432	
	median	max	median	max
Number of iterations	3	1,598	3	874
Time per iteration (sec)	2	12	3	12
Total time (sec)	7	1,788	9	776

parallel, and at each iteration, the cutting-plane algorithm checks if Hhugin has terminated, in which case it computes the full set of distance-based inequalities using the approximate tail-distances returned by Hhugin and adds them to the current constraint set. Lastly, at each iteration the cutting-plane algorithm compares its objective value to the length of Hhugin’s heuristic hyperpath  $P$ , and if they are equal, returns  $P$  (since we then know  $P$  is optimal, as the cutting-plane objective value is a lower bound on the shortest hyperpath length).

We also made one modification to Hhugin. Previously, when hyperedges were removed from the heap, their in-edge lists were frozen, so that new hyperedges were never added. We changed this behavior so that these in-edge lists continue to grow as new hyperedges are extracted from the heap.

Source code for the cutting-plane algorithm, including all datasets, is available at <http://mmunin.cs.arizona.edu> [20].

## 4.2 Comparing Alternate Hyperpath Methods

Mmunin outperforms state-of-the-art hypergraph methods for pathway inference: the hyperpath heuristic Hhugin [19], and the MILP for shortest acyclic hyperpaths [25], which we call AcycMILP. We compare these methods over all instances from Reactome and NCI-PID. Table 2 gives statistics for these instances.

For all these instances, Mmunin computes an optimal shortest hyperpath in less than 30 min, while allowing cycles for the first time. Mmunin surpasses AcycMILP on 22 Reactome instances and 38 NCI-PID instances where all  $s, t$ -hyperpaths are cyclic, hence AcycMILP fails to return any hyperpath. Mmunin outperforms Hhugin on the 23 NCI-PID instances where Hhugin is suboptimal.

`Mmunin` not only returns an optimal hyperpath for these instances, but also finishes its computation before `Hhugin`, showing that `Mmunin` (even without the distance-based constraints given by `Hhugin`) is faster for these instances. In fact, we found that `Mmunin` is faster than `Hhugin` on over 20% of all instances.

### 4.3 Speed of Computing Optimal Hyperpaths

`Mmunin` is typically fast in practice, with a median running time under 10s. Table 3 gives statistics on the running time, number of iterations, and time per iteration over the instances from each dataset. The maximum running time over all these instances is just under 30 min, demonstrating it is now feasible to find optimal shortest hyperpaths even for large instances with over 10,000 hyperedges. We note that inherent randomness in the CPLEX solver may cause variable running times, even for the same instance.

The number of iterations needed for the cutting-plane algorithm to return an optimal solution tends to be low, around 2 or 3 iterations, but can be as high as 1,598 iterations. Even though the cutting-plane algorithm may require many iterations, the instance with the highest average time per iteration takes only 12s per iteration. Note that at each iteration, the cutting-plane algorithm solves an ILP containing thousands of variables and inequalities within this time.

### 4.4 A Concrete Biological Example

As an illustration of the hyperpaths found by `Mmunin`, we show one concrete biological example from NCI-PID. This instance was chosen from the 23 instances where `Mmunin` outperforms the hyperpath heuristic `Hhugin`, because the size of the hyperpaths makes them reasonable to draw. Note that `AcycMILP` is also optimal on this instance, since the optimal hyperpath is acyclic. Figure 2 shows the hyperpaths returned by `Mmunin` and `Hhugin` for this instance, which represents the deactivation of “nuclear factor of activated T cells” (NFATC2) by “cAMP response element modulator” (CREM) in NCI-PID. Hyperedges drawn in a red dash are unique to `Mmunin`’s hyperpath, hyperedges in a green dotted line are unique to `Hhugin`’s hyperpath, and hyperedges in a solid black line are common to both hyperpaths. Eight of the nine hyperedges that are shared by both pathways have been omitted for simplicity, and have been replaced with ellipses. The hyperedges from MAPK8 to JUN and from MAPK3 to JUN denote transcription of JUN, where all other hyperedges denote biochemical reactions. Vertices with gray fill denote the activated form of a given protein. Note that stoichiometry of the reactions are not considered in our hyperpath formulation, so two copies of NFATC2 are needed in the final reaction.

The hyperpaths contain hyperedges from four different NCI-PID pathways: “Calcium signaling in the CD4<sup>+</sup> TCR pathway”, “RAS signaling in the CD4<sup>+</sup> TCR pathway”, “JNK signaling in the CD4<sup>+</sup> TCR pathway”, and “Nongenotropic androgen signaling”. The hyperpaths show CREM repressing the activated form of NFATC2 before forming a ternary complex with NFATC2

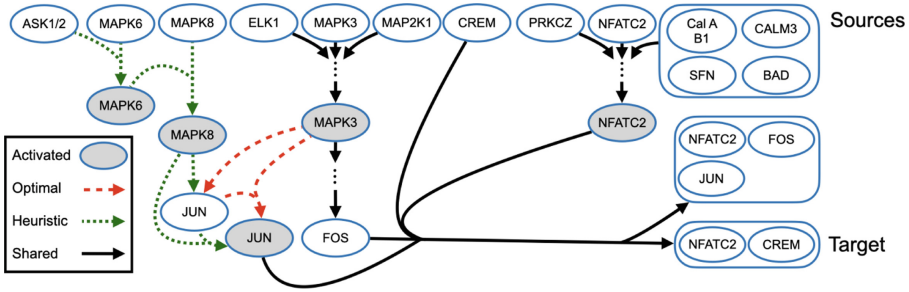


Fig. 2. Comparing hyperpaths whose target is the NFATC2/CREM complex.

and its DNA binding sites, resulting in the attenuation of transcription of T helper-1-specific cytokine genes in human medullary thymocytes [5].

Hhugin’s hyperpath contains 13 hyperedges and Mmunin’s hyperpath contains 11 hyperedges, which is optimal. Notably, hyperedges unique to Mmunin use vertices that are shared between both hyperpaths, and is therefore much simpler, making it a more likely pathway.

#### 4.5 Analysis of Recovering Known Pathways

Mmunin accurately recovers annotated pathways from Reactome. We define new problem instances for a small number of annotated pathways  $P^*$  from Reactome: ten pathways from the 22 instances with only cyclic hyperpaths (so AcycMILP fails to return any hyperpath), where the target appears in only one annotated Reactome pathway. (These ten benchmark pathways are: “Regulation of Complement Cascade”, “Sphingolipid Metabolism”, “Triglyceride Catabolism”, “Hydrocarboxylic Acid-Binding Receptors”, “Transport of Small Molecules”, “Proton/Oligopeptide Cotransporters”, “Interleukin-37 Signaling”, “Uncoating of the Influenza Virion”, “Glycogen Synthesis”, and “Tolerance by Mtb to Nitric Oxide Produced by Macrophages”.) For each instance, hypergraph  $G$  includes all vertices and hyperedges from Reactome, the sources consist of all vertices in  $G$  with no in-edges and all vertices in  $P^*$  with no in-edges from  $P^*$ , and the targets are all vertices in  $P^*$  with no out-edges from  $P^*$ . For these ten instances, the number of hyperedges in  $P^*$  ranges from 2 to 98, with a median of [14,17]. Note that these instances now contain multiple targets. For five of these instances, the sink is not reachable from the source, so to restore reachability, we added to the source set vertices that are unreachable due to an unreachable cycle. (A simple example of this is when vertex  $a$  is in the tail of all in-edges to vertex  $b$  and vice versa.) This resulted in a doubly-reachable set containing 3,600 hyperedges for some instances, which is twice the size of the doubly-reachable set for any single-target instance (shown in Table 1). Due to this increase, the running time of Mmunin on these instances was significantly longer, taking at most 25 h to compute an optimal hyperpath.

We compared `Mmunin`'s hyperpath  $P$  with the known pathway  $P^*$  from `Reactome` for each instance. For five of the instances,  $P = P^*$ , meaning `Mmunin` perfectly recovered the annotated pathway. For the other five instances,  $P$  contained fewer hyperedges than  $P^*$ , either due to redundant branches in  $P^*$  (so  $P \subset P^*$ ), or hyperedges outside  $P^*$  more efficiently reaching vertices within  $P^*$ , which is evidence of potential crosstalk between biological pathways. We measured the similarity of  $P$  and  $P^*$  for each instance by the so-called Sorensen coefficient  $2|P \cap P^*| / (|P| + |P^*|)$ . Over these ten instances, this similarity measure ranged from 0.62 to 1, with a median of  $[0.95, 1]$ . Overall, this experiment shows that `Mmunin` is able to accurately recover known pathways, or possibly discover more efficient ones.

## 5 Conclusion

We have presented a new formulation of the general shortest hyperpath problem as an integer linear program, and a practical cutting-plane algorithm that for the first time can find shortest hyperpaths with cycles. Comprehensive experiments on large real-world cellular reaction networks show we can quickly compute optimal hyperpaths, and accurately recover annotated biological pathways.

**Acknowledgments.** We thank Anna Ritz and T.M. Murali for helpful discussions and for providing the `BioPax` parser, and the anonymous referees for their useful comments. This research was supported by the US National Science Foundation, through grants CCF-1617192 and IIS-2041613 to JK.

## References

1. Acuña, V., Milreu, P.V., Cottret, L., Marchetti-Spaccamela, A., Stougie, L., Sagot, M.F.: Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. *Bioinformatics* **28**(19), 2474–2483 (2012)
2. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: *Molecular Biology of the Cell*. Garland Science, New York (2007)
3. Andrade, R., Wannagat, M., Klein, C.C., Acuña, V., Marchetti-Spaccamela, A., Milreu, P.V., Stougie, L., Sagot, M.F.: Enumeration of minimal stoichiometric precursor sets in metabolic networks. *Alg. Mol. Bio.* **11**(1), 25 (2016)
4. Ausiello, G., Laura, L.: Directed hypergraphs: introduction and fundamental algorithms—a survey. *Theoret. Comput. Sci.* **658**, 293–306 (2017)
5. Bodor, J., Habener, J.F.: Role of transcriptional repressor ICER in cyclic amp-mediated attenuation of cytokine gene expression in human thymocytes. *J. Biol. Chem.* **273**(16), 9544–9551 (1998)
6. Carbonell, P., Fichera, D., Pandit, S.B., Faulon, J.L.: Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms. *BMC Syst. Biol.* **6**(1), 10 (2012)
7. Christensen, T.S., Oliveira, A.P., Nielsen, J.: Reconstruction and logical modeling of glucose repression signaling pathways in *Saccharomyces cerevisiae*. *BMC Syst. Biol.* **3**(1), 7 (2009)



8. Cottret, L., et al.: Enumerating precursor sets of target metabolites in a metabolic network. In: Proceedings of the 8th Workshop on Algorithms in Bioinformatics, pp. 233–244 (2008)
9. Gallo, G., Longo, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. *Discret. Appl. Math.* **42**(2–3), 177–201 (1993)
10. Heath, L.S., Sioson, A.A.: Semantics of multimodal network models. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **6**(2), 271–280 (2009)
11. Hu, Z., Mellor, J., Wu, J., Kanehisa, M., Stuart, J.M., DeLisi, C.: Towards zoomable multidimensional maps of the cell. *Nat. Biotech.* **25**(5), 547–554 (2007)
12. Italiano, G.F., Nanni, U.: Online maintenance of minimal directed hypergraphs. Department of Computer Science, Columbia University, Tech. Rep. (1989)
13. Joshi-Tope, G., et al.: Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res.* **33**, D428–432 (2005)
14. Klamt, S., Haus, U.U., Theis, F.: Hypergraphs and cellular networks. *PLoS Comput. Biol.* **5**(5), e1000385 (2009)
15. Krieger, S.: Algorithmic Inference of Cellular Reaction Pathways and Protein Secondary Structure. Ph.D. dissertation, Department of Computer Science, The University of Arizona (July 2022)
16. Krieger, S., Kececioglu, J.: Fast approximate shortest hyperpaths for inferring pathways in cell signaling hypergraphs. In: Proceedings of the 21st ISCB Workshop on Algorithms in Bioinformatics (WABI). Leibniz International Proceedings in Informatics, vol. 201, pp. 1–20 (2021)
17. Krieger, S., Kececioglu, J.: Computing optimal factories in metabolic networks with negative regulation. *Bioinformatics*, In: Proceedings of the 30th ISCB Conference on Intelligent Systems for Molecular Biology (ISMB) **38**(Suppl\_1), i369–i377 (2022)
18. Krieger, S., Kececioglu, J.: Heuristic shortest hyperpaths in cell signaling hypergraphs. *Algorithms Mol. Biol.* **17**(1), 12 (2022)
19. Krieger, S., Kececioglu, J.: **Hhugin**: hypergraph heuristic for general shortest source-sink hyperpaths, version 1.0 (2022). <http://hhugin.cs.arizona.edu>
20. Krieger, S., Kececioglu, J.: **Mmunin**: integer-linear-programming-based cutting-plane algorithm for shortest source-sink hyperpaths, version 1.0 (2022). <http://mmunin.cs.arizona.edu>
21. Li, Y., McGrail, D.J., Latysheva, N., Yi, S., Babu, M.M., Sahni, N.: Pathway perturbations in signaling networks: linking genotype to phenotype. *Semin. Cell Dev. Biol.* **99**, 3–11 (2020)
22. Nielsen, L.R., Pretolani, D.: A remark on the definition of a  $B$ -hyperpath. Department of Operations Research, University of Aarhus, Tech. Rep. (2001)
23. Ramadan, E., Perincheri, S., Tuck, D.: A hyper-graph approach for analyzing transcriptional networks in breast cancer. In: Proceedings of the 1st ACM Conference on Bioinformatics and Computational Biology, pp. 556–562 (2010)
24. Ramadan, E., Tarafdard, A., Pothen, A.: A hypergraph model for the yeast protein complex network. In: Proceedings of the 18th Parallel and Distributed Processing Symposium, pp. 189–196 (2004)
25. Ritz, A., Avent, B., Murali, T.: Pathway analysis with signaling hypergraphs. *IEEE/ACM Transactions on Comp. Bio. and Bioinf.* **14**(5), 1042–1055 (2017)
26. Ritz, A., Murali, T.: Pathway analysis with signaling hypergraphs. In: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB), pp. 249–258 (2014)
27. Ritz, A., Tegge, A.N., Kim, H., Poirel, C.L., Murali, T.: Signaling hypergraphs. *Trends Biotechnol.* **32**(7), 356–362 (2014)

28. Schaefer, C.F., et al.: PID: the pathway interaction database. *Nucl. Acids Res.* **37**, 674–679 (2009)
29. Schwob, M.R., Zhan, J., Dempsey, A.: Modeling cell communication with time-dependent signaling hypergraphs. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **18**, 1151–1163 (2019)
30. Sharan, R., Ideker, T.: Modeling cellular machinery through biological network comparison. *Nat. Biotechnol.* **24**(4), 427–433 (2006)
31. Vidal, M., Cusick, M.E., Barabási, A.L.: Interactome networks and human disease. *Cell* **144**(6), 986–998 (2011)
32. Zarecki, R., Oberhardt, M.A., Reshef, L., Gophna, U., Ruppin, E.: A novel nutritional predictor links microbial fastidiousness with lowered ubiquity, growth rate, and cooperativeness. *PLoS Comput. Biol.* **10**(7), 1–12 (2014)
33. Zhou, W., Nakhleh, L.: Properties of metabolic graphs: biological organization or representation artifacts? *BMC Bioinform.* **12**(1), 132 (2011)



# T-Cell Receptor Optimization with Reinforcement Learning and Mutation Polices for Precision Immunotherapy

Ziqi Chen<sup>1</sup>, Martin Renqiang Min<sup>2(✉)</sup>, Hongyu Guo<sup>3</sup>, Chao Cheng<sup>4</sup>,  
Trevor Clancy<sup>5</sup>, and Xia Ning<sup>1,6,7(✉)</sup>

<sup>1</sup> Computer Science and Engineering, The Ohio State University,  
Columbus, OH 43210, USA  
[ning.104@osu.edu](mailto:ning.104@osu.edu)

<sup>2</sup> Machine Learning Department, NEC Labs, Princeton, NJ 08540, USA  
[renqiang@nec-labs.com](mailto:renqiang@nec-labs.com)

<sup>3</sup> Digital Technologies Research Centre, National Research Council Canada,  
Ontario, Canada

<sup>4</sup> Department of Medicine, Baylor College of Medicine, Houston, TX 77030, USA

<sup>5</sup> NEC Oncolmmunity AS, Oslo Cancer Cluster, Innovation Park,  
Ullernchausséen 64, 0379 Oslo, Norway

<sup>6</sup> Biomedical Informatics, The Ohio State University, Columbus,  
OH 43210, USA

<sup>7</sup> Translational Data Analytics Institute, The Ohio State University,  
Columbus, OH 43210, USA

**Abstract.** T cells monitor the health status of cells by identifying foreign peptides displayed on their surface. T-cell receptors (TCRs), which are protein complexes found on the surface of T cells, are able to bind to these peptides. This process is known as TCR recognition and constitutes a key step for immune response. Optimizing TCR sequences for TCR recognition represents a fundamental step towards the development of personalized treatments to trigger immune responses killing cancerous or virus-infected cells. In this paper, we formulated the search for these optimized TCRs as a reinforcement learning (RL) problem, and presented a framework TCRPPO with a mutation policy using proximal policy optimization. TCRPPO mutates TCRs into effective ones that can recognize given peptides. TCRPPO leverages a reward function that combines the likelihoods of mutated sequences being valid TCRs measured by a new scoring function based on deep autoencoders, with the probabilities of mutated sequences recognizing peptides from a peptide-TCR interaction predictor. We compared TCRPPO with multiple baseline methods and demonstrated that TCRPPO significantly outperforms all the baseline methods to generate positive binding and valid TCRs. These results demonstrate the potential of TCRPPO for both precision immunotherapy and peptide-recognizing TCR motif discovery.

**Keywords:** T-cell receptor · Immunotherapy · Reinforcement learning · Biological sequence design

## 1 Introduction

Immunotherapy is a fundamental treatment for human diseases, which uses a person’s immune system to fight diseases [9,30,31]. In the immune system, immune response is triggered by cytotoxic T cells which are activated by the engagement of the T cell receptors (TCRs) with immunogenic peptides presented by Major Histocompatibility Complex (MHC) proteins on the surface of infected or cancerous cells. The recognition of these foreign peptides is determined by the interactions between the peptides and TCRs on the surface of T cells. This process is known as TCR recognition and constitutes a key step for immune response [8,10]. Adoptive T cell immunotherapy (ACT), which has been a promising cancer treatment, genetically modifies the autologous T cells taken from patients in laboratory experiments, after which the modified T cells are infused into patients’ bodies to fight cancer. As one type of ACT therapies, TCR T cell (TCR-T) therapy directly modifies the TCRs of T cells to increase the binding affinities, which makes it possible to recognize and kill tumor cells effectively [22]. TCR is a heterodimeric protein with an  $\alpha$  chain and a  $\beta$  chain. Each chain has three loops as complementary determining regions (CDR): CDR1, CDR2 and CDR3. CDR1 and CDR2 are primarily responsible for interactions with MHC, and CDR3 interacts with peptides [21]. The CDR3 of the  $\beta$  chain has a higher degree of variations and is therefore arguably mainly responsible for the recognition of foreign peptides [17]. In this paper, we focused on the optimization of the CDR3 sequence of  $\beta$  chain in TCRs to enhance their binding affinities against peptide antigens, and we conducted the optimization through novel reinforcement learning. The success of our approach will have the potential to guide TCR-T therapy design. For the sake of simplicity, when we refer to TCRs in the rest of the paper, we mean the CDR3 of  $\beta$  chain in TCRs.

Despite the significant promise of TCR-T therapy, optimizing TCRs for therapeutic purposes remains a time-consuming process, which typically requires exhaustive screening for high-affinity TCRs, either *in vitro* or *in silico*. To accelerate this process, computational methods have been developed recently to predict peptide-TCR interactions [28], leveraging the experimental peptide-TCR binding data [25,29] and TCR sequences [5]. However, these peptide-TCR binding prediction tools cannot immediately direct the rational design of new high-affinity TCRs. Existing computational methods for biological sequence design include search-based methods [3], generative methods [14,16], optimization-based methods [12] and reinforcement learning (RL)-based methods [2,26]. However, all these methods generate sequences without considering additional conditions such as peptides, and thus cannot optimize TCRs tailored to recognizing different peptides. In addition, these methods do not consider the validity of generated sequences, which is important for TCR optimization as valid TCRs should follow specific characteristics [15].

In this paper, we presented a new reinforcement-learning (RL) framework based on proximal policy optimization (PPO) [24], referred to as TCRPPO<sup>1</sup> to computationally optimize TCRs through a mutation policy. In particular,

<sup>1</sup> The code is available at <https://github.com/ninglab/TCRPPO>.

TCRPP0 learns a joint policy to optimize TCRs customized for any given peptides. In TCRPP0, we designed a new reward function that measures both the likelihoods of the mutated sequences being valid TCRs, and the probabilities of the TCRs recognizing peptides. To measure TCR validity, we developed a TCR auto-encoder, referred to as TCR-AE, and utilized reconstruction errors from TCR-AE and also its latent space distributions, quantified by a Gaussian Mixture Model, to calculate novel validity scores. To measure peptide recognition, we leveraged a state-of-the-art peptide-TCR binding predictor ERGO [28] to predict peptide-TCR binding. Please note that TCRPP0 is a flexible framework, as ERGO can be replaced by any other binding predictors [4,32]. In addition, we designed a novel buffering mechanism, referred to as Buf-Opt, to revise TCRs that are difficult to optimize. We conducted extensive experiments using 7 million TCRs from TCRdb [5], 10 peptides from McPAS [29] and 15 peptides from VDJDDB [25]. Our experimental results demonstrated that TCRPP0 can substantially outperform the best baselines with best improvement of 45.04% and 52.89% in terms of generating qualified TCRs with high validity scores and high recognition probabilities, over McPAS and VDJDDB peptides, respectively. Figure 1 presents the overall architecture of TCRPP0.

## 2 Related Work

Existing methods developed for biological sequence design include search-based methods, deep generative methods, optimization-based methods and RL-based methods. Among search-based methods, the classical evolutionary search [3] uses an evolution strategy to randomly mutate the sequences and select desired ones in an iterative way. Among generative methods, Killoran *et al.* [16] optimized the latent embeddings of DNA sequences learned from a variational autoencoder towards better properties. Gupta *et al.* [14] used generative adversarial networks (GANs) to generate DNA sequences and selected the generated ones with desired properties to further optimize GANs. Among optimization-based methods, Gonzalez *et al.* [12] used a Gaussian process model to emulate the production rates of a certain protein across different gene designs in living cells, and then optimized the gene designs to improve the production rates using Bayesian optimization. Both the above generative methods and optimization methods aim at optimizing the biological sequences without any additional conditions. As a consequence, these methods are not applicable to our TCR optimization problem. In our problem, the optimization of TCRs must be tailored to given peptides, because TCRs binding to different peptides have different characteristics.

Despite the success of RL on many applications, there remains limited work of applying RL to biological sequence design. Angermueller *et al.* [2] developed a model-based RL method for biological sequence design using PPO to improve the sample efficiency, where the policy is trained over a simulator model learned to approximate the reward function. Skwark *et al.* [26] then leveraged a RL method based on PPO to discover a potential Covid-19 cure. Their method aims at identifying the variants of human angiotensin-converting enzyme (ACE2) protein

sequence that have higher binding affinities against the SARS-CoV-2 spike protein than the original ACE2 protein. Our TCRPPO also applies PPO with a mutation policy to optimize TCR sequences. However, TCRPPO is fundamentally different from previous methods in three aspects. First, TCRPPO learns a joint policy to optimize the TCRs customized for any given peptides. In addition, TCRPPO employs a comprehensive reward function to simultaneously optimize the validity and the recognition probability of the TCRs against the peptides. TCRPPO also leverages a buffering mechanism to generalize the optimization capability of TCRPPO to TCRs that are hard to optimize, or peptides with fewer positive binding TCRs.

### 3 Methods

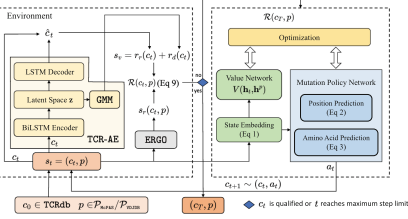


Fig. 1. Model architecture of TCRPPO

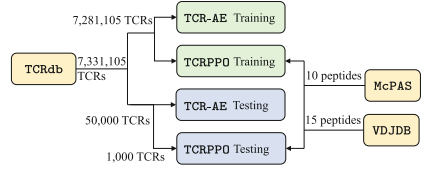


Fig. 2. Data flow for TCRPPO and TCR-AE training and testing

#### 3.1 Problem Definition

In this paper, the recognition ability of a TCR sequence against the given peptides is measured by a recognition probability, denoted as  $s_r$ . The likelihood of a sequence being a valid TCR is measured by a score, denoted as  $s_v$ . A *qualified* TCR is defined as a sequence with  $s_r > \sigma_r$  and  $s_v > \sigma_c$ , where  $\sigma_r$  and  $\sigma_c$  are pre-defined thresholds ( $\sigma_r=0.9$  and  $\sigma_c=1.2577$ , as discussed in Appendix A.2 and A.4.4, respectively). The goal of TCRPPO is to mutate the existing TCR sequences that have low recognition probability against the given peptide, into qualified ones. A peptide  $p$  or a TCR sequence  $c$  is represented as a sequence of its amino acids  $\langle o_1, o_2, \dots, o_i, \dots, o_l \rangle$ , where  $o_i$  is one of the 20 types of natural amino acids at the position  $i$  in the sequence, and  $l$  is the sequence length. We formulated the TCR mutation process as a Markov Decision Process (MDP)  $M = \{\mathcal{S}, \mathcal{A}, P, \mathcal{R}\}$  containing the following components:

- $\mathcal{S}$ : the state space, in which each state  $s \in \mathcal{S}$  is a tuple of a potential TCR sequence  $c$  and a peptide  $p$ , that is,  $s = (c, p)$ . Subscript  $t$  ( $t = 0, \dots, T$ ) is used to index step of  $s$ , that is,  $s_t = (c_t, p)$ . Please note that  $c_t$  may not be a valid TCR. A state  $s_t$  is a terminal state, denoted as  $s_T$ , if it contains a qualified  $c_t$ , or  $t$  reaches the maximum step limit  $T$ . Please also note that  $p$  will be sampled at  $s_0$  and will not change over time  $t$ ,

- $\mathcal{A}$ : the action space, in which each action  $\mathbf{a} \in \mathcal{A}$  is a tuple of a mutation site  $i$  and a mutant amino acid  $o$ , that is,  $\mathbf{a} = (i, o)$ . Thus, the action will mutate the amino acid at position  $i$  of a sequence  $c = \langle o_1, o_2, \dots, o_i, \dots, o_l \rangle$  into another amino acid  $o$ . Note that  $o$  has to be different from  $o_i$  in  $c$ .
- $\mathcal{P}$ : the state transition probabilities, in which  $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$  specifies the probability of next state  $s_{t+1}$  at time  $t + 1$  from state  $s_t$  at time  $t$  with the action  $\mathbf{a}_t$ . In our problem, the transition to  $s_{t+1}$  is deterministic, that is  $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t) = 1$ .
- $\mathcal{R}$ : the reward function at a state. In TCRPPO, all the intermediate rewards at states  $s_t$  ( $t = 0, \dots, T - 1$ ) are 0; only the final reward at  $s_T$  is used to guide the optimization.

### 3.2 Mutation Policy Network

TCRPPO mutates one amino acid in a sequence  $c$  at a step to modify  $c$  into a qualified TCR. Specifically, at the initial step  $t = 0$ , a peptide  $p$  is sampled as the target, and a valid TCR  $c_0$  is sampled to initialize  $s_0 = (c_0, p)$ ; at a state  $s_t = (c_t, p)$  ( $t > 0$ ), the mutation policy network of TCRPPO predicts an action  $\mathbf{a}_t$  that mutates one amino acid of  $c_t$  to modify it into  $c_{t+1}$  that is more likely to lead to a final, qualified TCR bound to  $p$ . TCRPPO encodes the TCRs and peptides in a distributed embedding space. It then learns a mapping between the embedding space and the mutation policy, as discussed below.

**Encoding of Amino Acids.** Following the idea in Chen *et al.* [6], we represented each amino acid  $o$  by concatenating three vectors: 1)  $\mathbf{o}^b$ , the corresponding row of  $o$  in the BLOSUM matrix, 2)  $\mathbf{o}^o$ , the one-hot encoding of  $o$ , and 3)  $\mathbf{o}^d$ , the learnable embedding, that is,  $o$  is encoded as  $\mathbf{o} = \mathbf{o}^b \oplus \mathbf{o}^o \oplus \mathbf{o}^d$ , where  $\oplus$  represents the concatenation operation. We used such a mixture of encoding methods to enrich the representations of amino acids within  $c$  and  $p$ .

**Embedding of States.** We embedded  $s_t = (c_t, p)$  via embedding its associated sequences  $c_t$  and  $p$ . For each amino acid  $o_{i,t}$  in  $c_t$ , we embedded  $o_{i,t}$  and its context information in  $c_t$  into a hidden vector  $\mathbf{h}_{i,t}$  using a one-layer bidirectional LSTM [13] as below,

$$\begin{aligned} \vec{\mathbf{h}}_{i,t}, \vec{\mathbf{c}}_{i,t} &= \text{LSTM}(\mathbf{o}_{i,t}, \vec{\mathbf{h}}_{i-1,t}, \vec{\mathbf{c}}_{i-1,t}; \vec{W}); \\ \overleftarrow{\mathbf{h}}_{i,t}, \overleftarrow{\mathbf{c}}_{i,t} &= \text{LSTM}(\mathbf{o}_{i,t}, \overleftarrow{\mathbf{h}}_{i+1,t}, \overleftarrow{\mathbf{c}}_{i+1,t}; \overleftarrow{W}); \\ \mathbf{h}_{i,t} &= \vec{\mathbf{h}}_{i,t} \oplus \overleftarrow{\mathbf{h}}_{i,t} \end{aligned} \quad (1)$$

where  $\vec{\mathbf{h}}_{i,t}$  and  $\overleftarrow{\mathbf{h}}_{i,t}$  are the hidden state vectors of the  $i$ -th amino acid in  $c_t$ ;  $\vec{\mathbf{c}}_{i,t}$  and  $\overleftarrow{\mathbf{c}}_{i,t}$  are the memory cell states of  $i$ -th amino acid;  $\vec{W}$  and  $\overleftarrow{W}$  are the learnable parameters of the two LSTM directions, respectively; and  $\vec{\mathbf{h}}_{0,t}$ ,  $\overleftarrow{\mathbf{h}}_{l_c,t}$ ,  $\vec{\mathbf{c}}_{0,t}$  and  $\overleftarrow{\mathbf{c}}_{l_c,t}$  ( $l_c$  is the length of  $c_t$ ) are initialized with random vectors. With the embeddings of all the amino acids, we defined the embedding of  $c_t$  as the concatenation of hidden vectors at the two ends, that is,  $\mathbf{h}_t = \vec{\mathbf{h}}_{l_c,t} \oplus \overleftarrow{\mathbf{h}}_{0,t}$ . We embedded a peptide sequence into a hidden vector  $\mathbf{h}^p$  using another bidirectional LSTM in the same way.

**Action Prediction.** To predict the action  $\mathbf{a}_t = (i, o)$  at time  $t$ , TCRPPO needs to make two predictions: 1) the position  $i$  of current  $c_t$  where  $\mathbf{a}_t$  needs to occur; 2) the new amino acid  $o$  that  $\mathbf{a}_t$  needs to place with at position  $i$ . To measure “how likely” the position  $i$  in  $c_t$  is the action site, TCRPPO uses the following network:

$$f(i) = \mathbf{w}^\top(\text{ReLU}(W_1\mathbf{h}_{i,t} + W_2\mathbf{h}^p)) / (\sum_{j=1}^{l_c} \mathbf{w}^\top(\text{ReLU}(W_1\mathbf{h}_{j,t} + W_2\mathbf{h}^p))), \quad (2)$$

where  $\mathbf{h}_{i,t}$  is the latent vector of  $o_{i,t}$  in  $c_t$  (Eq. 1);  $\mathbf{h}^p$  is the latent vector of  $p$ ;  $\mathbf{w}/W_j$  ( $j=1,2$ ) are the learnable vector/matrices. Thus, TCRPPO measures the probability of position  $i$  being the action site by looking at its context encoded in  $\mathbf{h}_{i,t}$  and the peptide  $p$ . The predicted position  $i$  is sampled from the probability distribution from Eq. 2 to ensure necessary exploration.

Given the predicted position  $i$ , TCRPPO needs to predict the new amino acid that should replace  $o_i$  in  $c_t$ . TCRPPO calculates the probability of each amino acid type being the new replacement as follows:

$$g(o) = \text{softmax}(U_1 \times \text{ReLU}(U_2\mathbf{h}_{i,t} + U_3\mathbf{h}^p)), \quad (3)$$

where  $U_j$  ( $j=1,2,3$ ) are the learnable matrices;  $\text{softmax}(\cdot)$  converts a 20-dimensional vector into probabilities over the 20 amino acid types. The replacement amino acid type is then determined by sampling from the distribution, excluding the original type of  $o_{i,t}$ .

### 3.3 Potential TCR Validity Measurement

Leveraging the literature [1, 35], we designed a novel scoring function to quantitatively measure the likelihood of a given sequence  $c$  being a valid TCR (i.e., to calculate  $s_v$ ), which will be part of the reward of TCRPPO. Specifically, we trained a novel auto-encoder model, denoted as TCR-AE, from only valid TCrs. We used the reconstruction accuracy of a sequence in TCR-AE to measure its TCR validity. The intuition is that since TCR-AE is trained from only valid TCrs, its encoding-decoding process will obey the “rules” of true TCR sequences, and thus, a non-TCR sequence could not be well reproduced from TCR-AE. However, it is still possible that a non-TCR sequence can receive a high reconstruction accuracy from TCR-AE, if TCR-AE learns some generic patterns shared by TCrs and non-TCrs and fails to detect irregularities, or TCR-AE has high model complexity [19, 35]. To mitigate this, we additionally evaluated the latent space within TCR-AE using a Gaussian Mixture Model (GMM), hypothesizing that non-TCrs would deviate from the dense regions of TCrs in the latent space.

TCR-AE. Figure 1 presents the auto-encoder TCR-AE. TCR-AE uses a bidirectional LSTM to encode an input sequence  $c$  into  $\mathbf{h}'$  by concatenating the last hidden vectors from the two LSTM directions (similarly as in Eq. 1). Please note that this bidirectional LSTM is independent of the mutation policy network in TCRPPO.  $\mathbf{h}'$  is then mapped into a latent embedding  $\mathbf{z}'$  as follows,

$$\mathbf{z}' = W^z\mathbf{h}', \quad (4)$$



which will be decoded back to a sequence  $\hat{c}$  via a decoder. The decoder has a single-directional LSTM that decodes  $\mathbf{z}'$  by generating one amino acid at a time as follows,

$$\mathbf{h}'_i, \mathbf{c}'_i = \text{LSTM}(\hat{\mathbf{o}}_{i-1}, \mathbf{h}'_{i-1}, \mathbf{c}'_{i-1}; W'); \quad \hat{o}_i = \text{softmax}(U' \times \text{ReLU}(U'_1 \mathbf{h}'_i + U'_2 \mathbf{z}')), \quad (5)$$

where  $\hat{\mathbf{o}}_{i-1}$  is the encoding of the amino acid  $\hat{o}_{i-1}$  that is decoded from step  $i - 1$ ;  $W'$  is the parameter. The LSTM starts with a zero vector  $\mathbf{o}_0 = \mathbf{0}$  and  $\mathbf{h}_0 = W^h \mathbf{z}'$ . The decoder infers the next amino acid by looking at the previously decoded amino acids encoded in  $\mathbf{h}'_i$  and the entire prospective sequence encoded in  $\mathbf{z}'$ .

Please note that TCR-AE is trained from TCRs, independently of TCRPPO and in an end-to-end fashion. Teacher forcing [34] is applied during training to feed the ground truth amino acids as inputs to predict the next amino acid, and thus cross entropy loss is applied on each amino acid to optimize TCR-AE. As a stand-alone module, TCR-AE is used to calculate the score  $s_v$ . The input sequence  $c$  to TCR-AE is encoded using only BLOSUM matrix as we found empirically that BLOSUM encoding can lead to a good reconstruction performance and a fast convergence compared to other combinations of encoding methods.

**Reconstruction-Based Score.** With a well-trained TCR-AE, we calculated the reconstruction-based TCR validity score of a sequence  $c$  as follows,

$$r_r(c) = 1 - \text{lev}(c, \text{TCR-AE}(c))/l_c \quad (6)$$

where  $\text{TCR-AE}(c)$  represents the reconstructed sequence of  $c$  from TCR-AE;  $\text{lev}(\cdot)$  is the Levenshtein distance, an edit-distance-based metric, between  $c$  and  $\text{TCR-AE}(c)$ ;  $l_c$  is the length of  $c$ . Higher  $r_r(c)$  indicates higher probability of  $c$  being a valid TCR. Please note that when TCR-AE is used in testing, the length of the reconstructed sequence might not be the same as the input  $c$ , because TCR-AE could fail to accurately predict the end of the sequence, leading to either too short or too long reconstructed sequences. Therefore, we normalized the Levenshtein distance using the length of input sequence  $l_c$  similarly to Snover *et al.* [27]. Please note that  $r_r(c)$  could be negative when the distance is greater than the sequence length. The negative values will not affect the use of the scores (i.e., negative  $r_r(c)$  indicates very different  $\text{TCR-AE}(c)$  and  $c$ ).

**Density Estimation-Based Score.** To better distinguish valid TCRs from invalid ones, TCRPPO also conducts a density estimation over the latent space of  $\mathbf{z}'$  (Eq. 4) using GMM. For a given sequence  $c$ , TCRPPO calculates the likelihood score of  $c$  falling within the Gaussian mixture region of training TCRs as follows,

$$r_d(c) = \exp\left(1 + \frac{\log P(\mathbf{z}')}{\tau}\right) \quad (7)$$

where  $\log P(\mathbf{z}')$  is the log-likelihood of the latent embedding  $\mathbf{z}'$ ;  $\tau$  is a constant used to rescale the log-likelihood value ( $\tau = 10$ ). We carefully selected the parameter  $\tau$  such that 90% of TCRs can have  $r_d(c)$  above 0.5. As we do not have invalid TCRs, we cannot use classification-based scaling methods such as Platt scaling [20] to calibrate the log likelihood values to probabilities.

**TCR Validity Scoring.** Combining the reconstruction-based scoring and density estimation-based scoring, we developed a new scoring method to measure TCR validity as follows:

$$s_v(c) = r_r(c) + r_d(c). \quad (8)$$

This method is used to evaluate if a sequence is likely to be a valid TCR and is used in the reward function.

### 3.4 TCRPPO Learning

**Final Reward.** We defined the final reward for TCRPPO based on  $s_r$  and  $s_v$  scores as follows,

$$\mathcal{R}(c_T, p) = s_r(c_T, p) + \alpha \min(0, s_v(c_T) - \sigma_c) \quad (9)$$

where  $s_r(c_T, p)$  is the predicted recognition probability by ERGO,  $\sigma_c$  is a threshold that  $c_T$  is very likely to be a valid TCR; and  $\alpha$  is the hyperparameter used to control the tradeoff between  $s_r$  and  $s_v$  ( $\alpha = 0.5$ ).

**Policy Learning.** We adopted the proximal policy optimization (PPO) [24] to optimize the policy network as discussed in Sect. 3.2. The objective function of PPO is defined as follows:

$$\begin{aligned} \max_{\Theta} L^{\text{CLIP}}(\Theta) &= \hat{\mathbb{E}}_t[\min(r_t(\Theta)\hat{A}_t, \text{clip}(r_t(\Theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)], \\ \text{where } r_t(\Theta) &= \frac{\pi_{\Theta}(a_t|s_t)}{\pi_{\Theta_{\text{old}}}(a_t|s_t)}, \end{aligned} \quad (10)$$

where  $\Theta$  is the set of learnable parameters of the policy network and  $r_t(\Theta)$  is the probability ratio between the action under current policy  $\pi_{\Theta}$  and the action under previous policy  $\pi_{\Theta_{\text{old}}}$ . Here,  $r_t(\Theta)$  is clipped to avoid moving  $r_t$  outside of the interval  $[1 - \epsilon, 1 + \epsilon]$ .  $\hat{A}_t$  is the advantage at timestep  $t$  computed with the generalized advantage estimator [23], measuring how much better a selected action is than others on average:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \quad (11)$$

where  $\gamma \in (0, 1)$  is the discount factor determining the importance of future rewards;  $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$  is the temporal difference error in which  $V(s_t)$  is a value function;  $\lambda \in (0, 1)$  is a parameter used to balance the bias and variance of  $V(s_t)$ . Here,  $V(\cdot)$  uses a multi-layer perceptron (MLP) to predict the future return of current state  $s_t$  from the peptide embedding  $\mathbf{h}^p$  and the TCR embedding  $\mathbf{h}_t$ . The objective function of  $V(\cdot)$  is as follows:

$$\min_{\Theta} L^V(\Theta) = \hat{\mathbb{E}}_t[(V(\mathbf{h}_t, \mathbf{h}^p) - \hat{R}_t)^2], \quad (12)$$

where  $\hat{R}_t = \sum_{i=t+1}^T \gamma^{i-t} r_i$  is the rewards-to-go. Because we only used the final rewards, that is  $r_i = 0$  if  $i \neq T$ , we calculated  $\hat{R}_t$  with  $\hat{R}_t = \gamma^{T-t} r_T$ . We also added the entropy regularization loss  $H(\Theta)$ , a popular strategy used for policy

gradient methods [18,24], to encourage the exploration of the policy. The final objective function of TCRPPO is defined as below,

$$\min_{\Theta} L(\Theta) = -L^{\text{CLIP}}(\Theta) + \alpha_1 L^V(\Theta) - \alpha_2 H(\Theta), \quad (13)$$

where  $\alpha_1$  and  $\alpha_2$  are two hyperparameters controlling the tradeoff among the PPO objective, the value function and the entropy regularization term.

**Reward-Informed Buffering and Re-optimization.** TCRPPO implements a novel buffering and re-optimizing mechanism, denoted as **Buf-Opt**, to deal with TCRs that are difficult to optimize, and to generalize its optimization capacity to more, diverse TCRs. To optimize TCRs, various number of mutations will be applied to get the binding TCRs. For TCRs requiring more mutations, it could be more difficult for TCRPPO to optimize; and thus re-optimizing these TCRs enables TCRPPO to explore more actions for the optimization of difficult TCRs, instead of being overwhelmed by relatively simple cases. This mechanism includes a buffer, which memorizes the TCRs that cannot be optimized to qualify. These hard sequences and the corresponding peptides will be sampled from the buffer again following the probability distribution below, to be further optimized by TCRPPO,

$$S(c, p) = \xi^{(1-\mathcal{R}(c_T, p))} / \Sigma. \quad (14)$$

In Eq. 14,  $S$  measures how difficult to optimize  $c$  against  $p$  based on its final reward  $\mathcal{R}(c_T, p)$  in the previous optimization,  $\xi$  is hyper-parameter ( $\xi = 5$  in our experiments), and  $\Sigma$  converts  $S(c, p)$  as a probability. It is expected that by doing the sampling and re-optimization, TCRPPO is better trained to learn from hard sequences, and also the hard sequences have the opportunity to be better optimized by TCRPPO. In case a hard sequence still cannot be optimized to qualify, it will have 50% chance of being allocated back to the buffer. In case the buffer is full (size 2,000 in our experiments), the sequences earliest allocated in the buffer will be removed. We referred to the TCRPPO with **Buf-Opt** as TCRPPO+b.

## 4 Experimental Settings

### 4.1 Datasets

We selected peptides and TCR sequences for the training and testing of TCRPPO and TCR-AE. Figure 2 summaries the peptides and TCRs used in our experiments.

**Peptides.** To test TCRPPO, we first identified a set of peptides that TCRPPO needs to optimize TCR sequences for. We aimed at selecting the peptides which are very likely to have reliable peptide-TCR binding predictions, such that their binding predictions can serve to test TCRPPO’s optimized TCRs against the respective peptides. We identified such peptides from two databases: McPAS [29] and VDJDDB [25], which have experimentally validated TCR-peptide binding pairs. We applied the autoencoder-based ERGO models [28] on McPAS and VDJDDB (McPAS and VDJDDB have pre-specified training and testing sets for each peptide), and

selected the peptides which have AUC values above 0.9 on their respective testing sets. This resulted in 10 peptides selected from McPAS, denoted as  $\mathcal{P}_{\text{McPAS}}$ , and 15 peptides selected from VDJDDB, denoted as  $\mathcal{P}_{\text{VDJDDB}}$ , with lengths ranging from 8 to 21, as presented in Appendix Table A.1. Additional discussion on peptides is available in Appendix A.1.

**TCR Sequences.** We then selected TCR sequences that TCRPPO needs to optimize against each of the peptides selected as above. We selected such sequences from the TCRdb database [5], which contains 277 million human TCR sequences, each with a TCR- $\beta$  sequence. Here, we used TCRdb, not McPAS or VDJDDB, because TCRdb’s sequences are valid TCRs and have no information on their binding affinities with the selected peptides. Therefore, these valid TCRs can be used to train TCR-AE to calculate  $s_v$ . Meanwhile, since TCRdb is much larger than McPAS (4,528 TCRs) and VDJDDB (50,049 TCRs), it is very likely that the  $s_v$  calculated from the TCR-AE, which is trained over TCRdb data, will be independent of the  $s_r$  calculated from ERGO, which is trained on McPAS or VDJDDB, avoiding possible correlation between  $s_v$  and  $s_r$  as in the reward (Eq.9).

We selected all the TCRs with lengths below 27 (ERGO can only predict sequences of length 27 or shorter) from TCRdb, resulting in 7,331,105 unique TCR- $\beta$  sequences. Figure A.1 presents the distribution of lengths of TCRs in TCRdb. As shown in Figure A.1, the most common length of TCRs is 15. Additional discussion on the length of TCRs is available in Appendix A.1. Among these selected sequences, we sampled 50K sequences, denoted as the validation set  $\mathcal{S}_v$ , to test and validate  $s_v$ ; within these 50K sequences, we again sampled 1K sequences, denoted as the testing set  $\mathcal{S}_{\text{test}}$ , to test TCRPPO performance once it is well trained. The remaining selected sequences (i.e., not in the validation set), denoted as the training set  $\mathcal{S}_{\text{train}}$ , are used to train TCRPPO; they are also used to train TCR-AE.

## 4.2 Experimental Setup

For all the selected peptides from a same database (i.e., 10 peptides from McPAS, 15 peptides from VDJDDB), we trained one TCRPPO agent, which optimizes the training sequences (i.e., 7,281,105 TCRs in Fig. 2) to be qualified against one of the selected peptides. The ERGO model trained on the corresponding database (the same ERGO model also used to select the peptides from the database as in Sect. 4.1) will be used to test recognition probabilities  $s_r$  for the TCRPPO agent. Please note that as in Springer *et al.* [28], one ERGO model is trained for all the peptides in each database (i.e., one ERGO predicts TCR-peptide binding for multiple peptides). Thus, the ERGO model is suitable to test  $s_r$  for multiple peptides in our setting. Also note that we trained one TCRPPO agent corresponding

to each database, because peptides and TCRs in these two databases are very different, demonstrated by the inferior performance of an ERGO trained over the two databases together, and discussed in Springer *et al.* [28].

TCRPP0 mutates each sequence up to 8 steps (i.e.,  $T = 8$ ). In TCRPP0 training, an initial TCR sequence (i.e.,  $c_0$  in  $s_0$ ) is randomly sampled from  $\mathcal{S}_{trn}$ , and will be mutated in the following states; a peptide  $p$  is randomly sampled at  $s_0$ , and remains the same in the following states (i.e.,  $s_t = (c_t, p)$ ). Once a TCRPP0 is well trained from  $\mathcal{S}_{trn}$ , it will be tested on  $\mathcal{S}_{tst}$ . We set the dimensions of the hidden layers (e.g., hidden layers of action prediction networks) as 256, and the dimensions of latent embeddings (e.g.,  $\mathbf{h}^p$ ,  $\mathbf{h}_t$ ) as 128 (i.e., half of the hidden dimensions). Other hyper-parameters and the details of hyper-parameter selection of the TCR mutation environment, the policy network and the RL agent are available in Appendix A.2.

### 4.3 Baseline Methods

We compared the TCRPP0 method and TCRPP0+b with multiple baseline methods of two primary categories: 1) generative methods that generate a new TCR in its entirety, and 2) mutation-based methods that optimize TCRs via mutating amino acids of existing TCRs. For generative methods, we used two baseline methods including Monte Carlo tree search (MCTS) [7] and a variational autoencoder with backpropagation (BP-VAE) [11]. For mutation-based methods, we used three baseline methods to mutate each TCR sequence up to 8 steps and stop the mutation once a qualified TCR is generated, including random mutation (RM), greedy mutation (Greedy) and genetic mutation (Genetic) [33]. In addition, we used a random selection method (RS) as another baseline to randomly sample a TCR from TCRdb, which helps quantify the space of valid TCRs. More details about baseline methods are available in Appendix A.3.

### 4.4 Evaluation Metrics

We evaluated all the methods using six metrics including: (1) qualification rate  $q\%$ , which measures the percentage of qualified TCRs (Sect. 3.1) among all the output TCRs; (2) edit distance between  $c_0$  and  $c_T$  (**edist**), which measures sequence difference between  $c_0$  and qualified  $c_T$ ; (3) average TCR validity score  $\bar{s}_v$  over valid TCRs or over qualified TCRs; (4) average recognition probability  $\bar{s}_r$  over valid TCRs or over qualified TCRs; (5) validity rate  $v\%$ , which measures the percentage of valid TCRs (Sect. 3.1) among all the output TCRs; (6) average number of calls to  $\mathcal{R}$  calculation ( $\#\mathcal{R}$ ) (Eq. 9) over all the generated TCRs, which estimates the efficiency of methods. We calculated the metrics over two different sets of output TCRs: (1) the set of valid TCRs  $\mathcal{C}_v$ :  $\mathcal{C}_v = \{c | s_v(c) > \sigma_c\}$ ; (2) the set of qualified TCRs  $\mathcal{C}_q$ :  $\mathcal{C}_q = \{c | s_r(c) > \sigma_c, c \in \mathcal{C}_v\}$ .

Table 1. Overall performance comparison

Dataset	Method	q%	edist	$\overline{s_v}(C_q)$	$\overline{s_v}(C_v)$	$\overline{s_r}(C_q)$	$\overline{s_r}(C_v)$	v%	# $\mathcal{R}$
$\mathcal{P}_{\text{McPAS}}$	RS	0.05 ± 0.12	–	1.46 ± 0.16	<b>1.73 ± 0.17</b>	0.95 ± 0.01	0.05 ± 0.00	95.22 ± 0.45	<b>1</b>
	MCTS	0.01 ± 0.03	–	1.38 ± 0.00	1.34 ± 0.03	0.93 ± 0.00	0.09 ± 0.09	0.59 ± 0.28	200
	BP-VAE	0.04 ± 0.09	–	1.31 ± 0.02	1.34 ± 0.02	0.95 ± 0.02	0.09 ± 0.04	6.86 ± 2.52	7
	RM (n=5)	1.27 ± 2.09	<b>2.88 ± 1.50</b>	1.49 ± 0.07	1.54 ± 0.02	0.93 ± 0.02	0.18 ± 0.09	99.36 ± 0.21	39
	RM (n=10)	2.15 ± 3.86	3.04 ± 1.48	1.50 ± 0.11	1.52 ± 0.02	0.93 ± 0.02	0.26 ± 0.11	99.84 ± 0.12	77
	Greedy	20.58 ± 17.85	5.10 ± 1.09	1.46 ± 0.03	1.44 ± 0.02	0.93 ± 0.01	0.62 ± 0.12	99.98 ± 0.04	74
	Genetic	25.18 ± 21.28	5.16 ± 1.05	1.46 ± 0.03	1.46 ± 0.02	0.93 ± 0.01	0.69 ± 0.12	<b>100.00 ± 0.00</b>	166
	TCRPP0	25.89 ± 29.60	5.15 ± 2.02	<b>1.55 ± 0.19</b>	1.51 ± 0.17	<b>0.97 ± 0.03</b>	0.76 ± 0.27	65.23 ± 12.47	7
	TCRPP0+b	<b>36.52 ± 30.25</b>	5.37 ± 1.81	<b>1.55 ± 0.17</b>	1.53 ± 0.17	0.96 ± 0.03	<b>0.83 ± 0.21</b>	75.83 ± 9.46	7
	RS	0.08 ± 0.10	–	1.58 ± 0.20	<b>1.73 ± 0.28</b>	0.94 ± 0.02	0.03 ± 0.03	95.02 ± 0.46	<b>1</b>
$\mathcal{P}_{\text{VDJDB}}$	MCTS	0.00 ± 0.00	–	0.00 ± 0.00	1.35 ± 0.03	0.00 ± 0.00	0.05 ± 0.06	0.46 ± 0.09	200
	BP-VAE	0.08 ± 0.12	–	1.33 ± 0.03	1.35 ± 0.01	0.95 ± 0.03	0.05 ± 0.03	18.46 ± 5.13	9
	RM (n=5)	2.15 ± 1.82	<b>2.84 ± 1.47</b>	1.48 ± 0.09	1.55 ± 0.01	0.94 ± 0.02	0.16 ± 0.06	99.33 ± 0.21	39
	RM (n=10)	4.41 ± 3.72	3.03 ± 1.52	1.48 ± 0.07	1.52 ± 0.02	0.94 ± 0.01	0.24 ± 0.08	99.86 ± 0.08	77
	Greedy	31.81 ± 17.09	5.01 ± 1.12	1.47 ± 0.03	1.44 ± 0.02	0.93 ± 0.01	0.64 ± 0.09	99.96 ± 0.06	71
	Genetic	38.57 ± 20.57	5.07 ± 1.07	1.48 ± 0.03	1.47 ± 0.02	0.94 ± 0.01	0.72 ± 0.09	<b>100.00 ± 0.00</b>	156
	TCRPP0	45.82 ± 21.59	5.34 ± 1.58	1.51 ± 0.19	1.51 ± 0.19	0.96 ± 0.02	0.87 ± 0.18	69.99 ± 11.74	7
	TCRPP0+b	<b>58.97 ± 29.11</b>	5.20 ± 1.69	<b>1.65 ± 0.20</b>	1.63 ± 0.19	<b>0.97 ± 0.02</b>	<b>0.89 ± 0.17</b>	82.41 ± 5.54	6

Columns represent: q%: the percentage of qualified TCR sequences; edist: the edit distance between the original TCR sequences and the mutated qualified TCR sequences;  $\overline{s_v}(C_q)/\overline{s_v}(C_v)$ : the average  $s_v$  scores over the TCRs in set  $C_q/C_v$ ;  $\overline{s_r}(C_q)/\overline{s_r}(C_v)$ : the average  $s_r$  probabilities over the TCRs in set  $C_q/C_v$ ; v%: the percentage of valid TCR sequences. # $\mathcal{R}$ : the average number of calls to calculate  $\mathcal{R}$  functions. Values  $x \pm y$  represent mean  $x$  and standard deviation  $y$ . “–”: not applicable. The “n” value in RM indicates that RM is done  $n$  times over each sequence.

## 5 Experimental Results

### 5.1 Comparison on TCR Optimization Methods

**Overall Comparison.** Table 1 presents the overall comparison among all the methods over  $\mathcal{P}_{\text{MCPAS}}$  and  $\mathcal{P}_{\text{VDJDB}}$ . In  $\mathcal{P}_{\text{MCPAS}}$ , TCRPPO methods achieve overall the best performance: in terms of  $q\%$ , TCRPPO achieves  $25.89 \pm 29.60\%$ , which slightly outperforms the best  $q\%$  from the baseline method **Genetic** ( $25.18 \pm 21.28\%$ ). TCRPPO+b achieves the best  $q\%$  at  $36.52 \pm 30.25\%$  among all the methods, which is 45.04% better than the best from the baseline methods ( $25.18 \pm 21.28\%$  from **Genetic**). TCRPPO+b achieves so with a few  $\mathcal{R}$  calls ( $\#\mathcal{R} = 7$ ). In  $\mathcal{P}_{\text{VDJDB}}$ , TCRPPO methods also achieve overall the best performance: in terms of  $q\%$ , TCRPPO and TCRPPO+b outperform the best baseline **Genetic** by 18.80% and 52.89%.

Among the qualified TCRs ( $C_q$ ) for  $\mathcal{P}_{\text{MCPAS}}$ , TCRPPO and TCRPPO+b methods achieve the highest  $\bar{s}_v$  values on average ( $1.55 \pm 0.19$  for TCRPPO;  $1.55 \pm 0.17$  for TCRPPO+b), with above 6% improvement from those of **Genetic**, which achieves the best  $q\%$  among all the baseline methods. Note that qualified TCRs must have  $s_v$  above  $\sigma_c$ , which is set as 1.2577 as discussed in Appendix A.4.4. The significant high  $s_v$  values from TCRPPO methods demonstrate that TCRPPO is able to generate qualified TCRs that are highly likely to be valid TCRs. In terms of  $s_r$  among qualified TCRs, TCRPPO methods have  $\bar{s}_r(C_q)$  value  $0.97 \pm 0.03$ , above the  $\sigma_r$ . Note that  $\sigma_r = 0.9$ , a very high threshold for  $s_r$  to determine TCR-peptide binding, is actually a very tough constraint. The fact that TCRPPO methods can survive this constraint with substantially high  $q\%$  and highly likely valid TCRs as results demonstrates the strong capability of TCRPPO methods. In addition, TCRPPO methods need a few number of calls to calculate  $\mathcal{R}$  (i.e., 7 for TCRPPO, compared to 166 for **Genetic**), indicating that TCRPPO is very efficient in identifying qualified TCRs. In  $\mathcal{P}_{\text{VDJDB}}$ , we observed very similar trends as those in  $\mathcal{P}_{\text{MCPAS}}$ .

Among all the baseline methods, **RS** randomly selects a valid TCR from **TCRdb**, given that some TCRs in **TCRdb** may already qualify. Thus, it is a naive baseline for all the other methods. It has  $v\%$  around 95%, corresponding to how our  $s_v$  threshold  $\sigma_c$  is identified (by 95 percentile true positive rate) as discussed in Appendix A.4.4. On average, among all valid TCRs, about 0.05% ( $q\%$  in **RS**) TCRs are qualified TCRs. **RM**, **Greedy**, **Genetic** and TCRPPO methods substantially outperform this baseline method.

**Comparison Among Mutation-Based Methods.** **RM**, **Greedy**, **Genetic**, TCRPPO and TCRPPO+b are mutation-based methods: they start from a valid TCR from **TCRdb**, and optimize the TCR by mutating its amino acids. Among all the mutation-based methods, TCRPPO and TCRPPO+b outperform others as discussed above. Below we only focused on  $\mathcal{P}_{\text{MCPAS}}$ , as similar trends exist on  $\mathcal{P}_{\text{VDJDB}}$ .

In  $\mathcal{P}_{\text{MCPAS}}$ , **RM** underperforms all the other mutation-based methods: it has  $q\%$  below 3% on average, but has very high  $v\%$  (close to 100%). **RM** uses  $\mathcal{R}$  to select randomly mutated sequences, which decomposes to its  $s_v$  and its  $s_r$  components. **RM** starts from valid TCRs with high  $s_v$  already, but low  $s_r$  in general. During random mutation,  $s_r$  cannot be easily improved as no knowledge

or informed guidance is used to direct the mutation towards better  $s_r$ . Therefore, the  $s_v$  component in  $\mathcal{R}$  will dominate, leading to that the final selected, best mutated sequence tends to have high  $s_v$  to satisfy high  $\mathcal{R}$ . Such best mutated sequence tends to occur after only a few random mutations, since as shown in Appendix A.4.4, random mutations can quickly decrease  $s_v$  values. Thus, the qualified sequences produced from RM tend to be more similar to the initial TCRs ( $\text{edist}$  is small, around 3; high  $v\%$  as  $99.36 \pm 0.21\%$ ), their  $s_v$  values are high (close to 1.5; highest among all mutation-based methods) but  $q\%$  is low due to hardly improved  $s_r$  values.

In  $\mathcal{P}_{\text{McPAS}}$ , **Greedy** has a better  $q\%$  ( $20.58 \pm 17.85\%$ ) than RM and also a high  $v\%$  ( $99.98 \pm 0.04\%$ ). **Greedy** leverages a greedy strategy to select the random mutation that gives the best  $\mathcal{R}$  at the current step. Thus, it leverages some guidance on  $s_r$  improvement based on  $\mathcal{R}$ , and can improve  $s_r$  values compared to RM. As in Table 1, it has better  $s_r(\mathcal{C}_v)$  value ( $0.62 \pm 0.12$ ) for valid TCRs. Meanwhile, **Greedy** explores a large sequence space, allowing its to identify more valid TCRs, leading to high  $v\%$  ( $99.98 \pm 0.04$ ), high  $q\%$  ( $20.58 \pm 17.85$ ; also due to better  $s_r$  improvement) but more diverse results ( $\text{edist} = 5.10 \pm 1.09$ ) than RM.

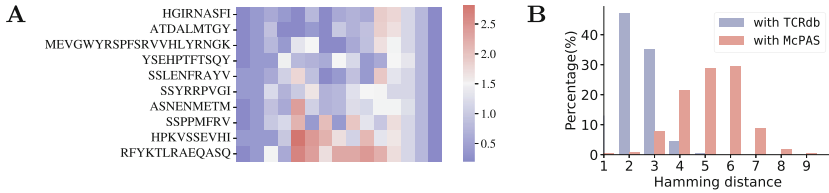
**Genetic** is the second best mutation-based method on  $\mathcal{P}_{\text{McPAS}}$ , after TCRPPO methods. **Genetic** explores a sequence space even larger than that in **Greedy**, and uses  $\mathcal{R}$  to guide next mutations also in a greedy way. Thus, it enjoys the opportunity to reach more, potentially qualified TCRs, demonstrated by high  $\text{edist}$  ( $5.16 \pm 1.05$ ) indicating diverse sequences, and thus achieves a better  $q\%$  ( $25.18 \pm 21.28\%$ ) than **Greedy**, even better than that of TCRPPO, and a high  $v\%$  ( $100.00 \pm 0.00\%$ ), at a significant cost of many more calls to calculate  $\mathcal{R}$ . Even though, it still significantly underperforms TCRPPO+b in terms of  $q\%$  and qualities of qualified TCRs as discussed earlier.

**Comparison Between Mutation-Based Methods and Generation-Based Methods.** Overall, mutation-based methods substantially outperform generation-based methods. For example, in terms of  $q\%$ , mutation-based methods (excluding random mutation RM) has an average  $q\%$  22.88% in  $\mathcal{P}_{\text{McPAS}}$ , compared to 0.03% of the generation-based methods (MCTS and BP-VAE). In addition to the superior performance, mutation-based methods have strong biological relevance that make them very suitable and practical for TCR-T based precision immunotherapy, in which they can be readily employed to optimize existing TCRs found in patient’s TCR repertoire. However, any promising TCRs generated from generation-based methods have to be either synthesized, which could be both very costly and technically challenging, or mutated from existing TCRs that are similar to the generated TCRs in their amino acids. Detailed discussions on generation-based methods are available in Appendix A.4.1.

## 5.2 Evaluation on Optimized TCR Sequences

Figure 3A presents the entropy of amino acid distributions at each sequence position among the length-15 TCRs for each  $\mathcal{P}_{\text{McPAS}}$  peptide. Here the TCRs are optimized by TCRPPO with respect to the McPAS peptides. Figure 3A clearly





**Fig. 3.** Optimized TCR patterns for McPAS peptides (A); TCR distances (B).

shows some common patterns among all the optimized TCRs. For example, the first three positions and last four positions tend to have high conservation. TCRs for some peptides (e.g., “SSPPMFRV”, “ASNENMETM”, and “RFYKTLRAEQASQ”) have high variations at internal regions. Similar patterns are also observed among the binding TCRs for other peptides in *VDJDB*<sup>2</sup>.

Figure 3B presents the difference between qualified TCRs optimized by *TCRPP0* and existing TCRs. It presents the distribution of Hamming distances between qualified TCRs (length 15) for peptide “RFYKTLRAEQASQ” and their most similar (in terms of Hamming distance) TCRs that are known to bind to this peptide in *McPAS*; and the distribution of Hamming distances between qualified TCRs for this peptide and their most similar TCRs in *TCRdb*. This figure demonstrates that the qualified TCRs by *TCRPP0* are actually different from known binding TCRs, but there are TCRs similar to them existing in *TCRdb*. This can be meaningful for precision immunotherapy, as *TCRPP0* can produce diverse TCR candidates that are different from known TCRs, leading to a novel sequence space; meanwhile, these TCR candidates actually have similar human TCRs available for further medical evaluation and investigation purposes.

Additional analyses are available in Appendix A.4, such as the overview of amino acid distributions of TCRs, the patterns for binding TCRs and the comparison on TCR detection. Specifically, we found that *TCRPP0* can successfully learn the patterns of TCRs (Appendix A.4.2); we also found that *TCRPP0* can identify the specific binding patterns which are more conserved than the real binding patterns (Appendix A.4.3). We also found that our  $s_v$  scoring method successfully distinguishes TCRs from non-TCRs in Appendix A.4.4.

## 6 Conclusions and Outlook

In this paper, we presented a reinforcement learning framework to optimize TCRs for more effective TCR recognition, which has the potential to guide TCR engineering therapy. Our experimental results in comparison with generation-based methods and mutation-based methods on optimizing TCRs demonstrate that *TCRPP0* outperforms the baseline methods. Our analysis on the TCRs generated by *TCRPP0* demonstrates that *TCRPP0* can successfully learn the conservation patterns of TCRs. Our experiments on the comparison between the generated TCRs and existing TCRs demonstrate that *TCRPP0* can generate TCRs similar

<sup>2</sup> <https://vdjdb.cdr3.net/motif>.

to existing human TCRs, which can be used for further medical evaluation and investigation. Our results in TCR detection comparison show that the  $s_v$  score in our framework can very effectively detect non-TCR sequences. Our analysis on the distribution of  $s_v$  scores over mutations demonstrates that TCRPPO mutates sequences along the trajectories not far away from valid TCRs.

Our proposed TCRPPO is a modular and flexible framework. Thus, the TCR-AE and ERGO scoring functions in the reward design can be replaced with other predictors trained on large-scale data when available. Also, TCRPPO can be further improved from the following perspectives. First, the recognition probabilities of TCRs considered in our paper are based on a peptide-TCR binding predictor (i.e., ERGO) rather than experimental validation. Therefore, testing the generated qualified TCR candidates in a wet-lab will be needed ultimately to validate the interactions between TCRs and peptides. Moreover, when the predicted recognition probabilities are not sufficiently accurate, TCRPPO learned from unreliable rewards could be inaccurate, resulting in generating TCRs that could not recognize the given peptide. Thus, it could be an interesting and challenging future work to incorporate the reliabilities of predictions in the reward function of TCRPPO, so that the effect of unreliable recognition probabilities can be alleviated. Finally, TCRPPO only considers the CDR3 of  $\beta$  chain in TCRs, while other regions of TCRs, though not contributing most to interactions between TCRs and peptides, are not considered. In this sense, incorporating other regions of TCRs (e.g., CDR3 of alpha chains) could be an interesting future work.

## References




1. Abati, D., Porrello, A., Calderara, S., Cucchiara, R.: Latent space autoregression for novelty detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
2. Angermüller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., Colwell, L.: Model-based reinforcement learning for biological sequence design. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020 (2020)
3. Arnold, F.H.: Design by directed evolution. *Acc. Chem. Res.* **31**(3), 125–131 (1998)
4. Cai, M., Bang, S., Zhang, P., Lee, H.: ATM-TCR: TCR-epitope binding affinity prediction using a multi-head self-attention model. *Front. Immunol.* **13** (2022)
5. Chen, S.Y., Yue, T., Lei, Q., Guo, A.Y.: TCRdb: a comprehensive database for t-cell receptor sequences with powerful search function. *Nucleic Acids Res.* **49**(D1), D468–D474 (2020)
6. Chen, Z., Min, M.R., Ning, X.: Ranking-based convolutional neural network models for peptide-MHC class i binding prediction. *Front. Mol. Biosci.* **8** (2021)
7. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: van den Herik, H.J., Ciancarini, P., Donkers, H.H.L.M.J. (eds.) CG 2006. LNCS, vol. 4630, pp. 72–83. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-75538-8\\_7](https://doi.org/10.1007/978-3-540-75538-8_7)
8. Craiu, A., Akopian, T., Goldberg, A., Rock, K.L.: Two distinct proteolytic processes in the generation of a major histocompatibility complex class i-presented peptide. *Proc. Natl. Acad. Sci.* **94**(20), 10850–10855 (1997)

9. Esfahani, K., Roudaia, L., Buhlaiga, N., Rincon, S.D., Papneja, N., Miller, W.: A review of cancer immunotherapy: From the past, to the present, to the future. *Curr. Oncol.* **27**(12), 87–97 (2020)
10. Glanville, J., et al.: Identifying specificity groups in the t cell receptor repertoire. *Nature* **547**(7661), 94–98 (2017)
11. Gómez-Bombarelli, R., et al.: Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Sci.* **4**(2), 268–276 (2018)
12. González, J., Longworth, J., James, D.C., Lawrence, N.D.: Bayesian optimization for synthetic gene design (2015)
13. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **18**(5–6), 602–610 (2005)
14. Gupta, A., Zou, J.: Feedback GAN for DNA optimizes protein functions. *Nat. Mach. Intell.* **1**(2), 105–111 (2019)
15. Hou, X., et al.: Analysis of the repertoire features of TCR beta chain CDR3 in human by high-throughput sequencing. *Cell. Physiol. Biochem.* **39**(2), 651–667 (2016)
16. Killoran, N., Lee, L.J., Delong, A., Duvenaud, D., Frey, B.J.: Generating and designing DNA with deep generative models. *CoRR* abs/1712.06148 (2017)
17. La Gruta, N.L., Gras, S., Daley, S.R., Thomas, P.G., Rossjohn, J.: Understanding the drivers of MHC restriction of t cell receptors. *Nat. Rev. Immunol.* **18**(7), 467–478 (2018)
18. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, pp. 1928–1937. PMLR, New York, New York, USA (20–22 June 2016)
19. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: a review. *ACM Comput. Surv.* **54**(2) (2021)
20. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.* **10**(3), 61–74 (1999)
21. Rossjohn, J., Gras, S., Miles, J.J., Turner, S.J., Godfrey, D.I., McCluskey, J.: T cell antigen receptor recognition of antigen-presenting molecules. *Annu. Rev. Immunol.* **33**, 169–200 (2015)
22. Sadelain, M., Rivière, I., Riddell, S.: Therapeutic t cell engineering. *Nature* **545**(7655), 423–431 (2017)
23. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2016)
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. *CoRR* abs/1707.06347 (2017)
25. Shugay, M., Bagaev, D.V., Zvyagin, I.V., Vroomans, R.M., Crawford, J.C., Dolton, G., et al.: VDJdb: a curated database of t-cell receptor sequences with known antigen specificity. *Nucleic Acids Res.* **46**(D1), D419–D427 (2017)
26. Skwark, M.J., et al.: Designing a prospective COVID-19 therapeutic with reinforcement learning. *CoRR* abs/2012.01736 (2020)
27. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pp. 223–231. Cambridge, Massachusetts, USA (8–12 August 2006)
28. Springer, I., Besser, H., Tickotsky-Moskovitz, N., Dvorkin, S., Louzoun, Y.: Prediction of specific TCR-peptide binding from large dictionaries of TCR-peptide pairs. *Front. Immunol.* **11** (2020)

29. Tickotsky, N., Sagiv, T., Prilusky, J., Shifrut, E., Friedman, N.: McPAS-TCR: a manually curated catalogue of pathology-associated t cell receptor sequences. *Bioinformatics* **33**(18), 2924–2929 (2017)
30. Verdegaal, E.M.E., et al.: Neoantigen landscape dynamics during human melanoma–t cell interactions. *Nature* **536**(7614), 91–95 (2016)
31. Waldman, A.D., Fritz, J.M., Lenardo, M.J.: A guide to cancer immunotherapy: from t cell basic science to clinical practice. *Nat. Rev. Immunol.* **20**(11), 651–668 (2020)
32. Weber, A., Born, J., Martínez, M.R.: TITAN: T-cell receptor specificity prediction with bimodal attention networks. *Bioinformatics* **37**(Supplement\_1), i237–i244 (2021)
33. Whitley, D.: A genetic algorithm tutorial. *Stat. Comput.* **4**(2) (1994)
34. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.* **1**(2), 270–280 (1989)
35. Zong, B., et al.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: *International Conference on Learning Representations* (2018)

# **Short Papers**

# TREE-QMC: Improving Quartet Graph Construction for Scalable and Accurate Species Tree Estimation from Gene Trees

Yunheng Han<sup>1</sup>  and Erin K. Molloy<sup>1,2</sup>  

<sup>1</sup> Department of Computer Science, University of Maryland, College Park, MD 20742, USA

<sup>2</sup> University of Maryland Institute for Advanced Computer Studies, College Park, MD 20742, USA  
ekmolloy@umd.edu

*Background:* Summary methods are one of the dominant approaches to species tree estimation from genome-scale data sets. They are utilized as part of a pipeline in which the first step is to estimate trees from individual regions of the genome (called gene trees) and the second step is to “summarize” them into a species tree. This approach can fail to produce accurate species trees when the input gene trees are highly discordant due to gene tree estimation error as well as biological processes, like incomplete lineage sorting (ILS) [5].

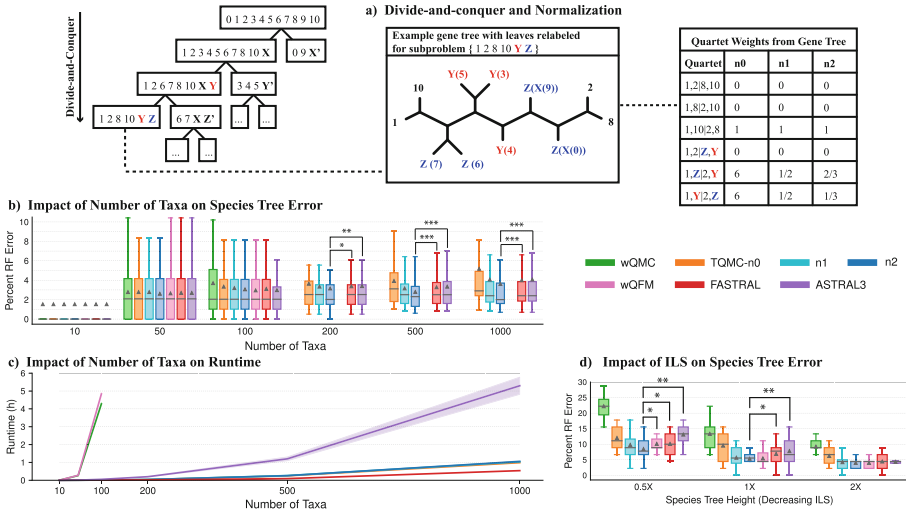
*Methods:* We present TREE-QMC, a new summary method that is fast and accurate under such challenging scenarios. TREE-QMC builds upon the algorithmic framework of wQMC [1], which takes a set of weighted quartets (four-leaf trees) as input and then builds a species tree in a divide-and-conquer fashion. At each step in the divide phase, a branch (split) in the species tree is identified by constructing a graph and then seeking its max cut. We improve upon this approach in two ways. First, we address scalability by providing an algorithm to construct the graph directly from the input gene trees instead of the  $\Theta(n^4)$  weighted quartets. This gives TREE-QMC a time complexity of  $O(n^3k)$  with some mild assumptions on subproblem sizes, where  $n$  is the number of taxa and  $k$  is the number of gene trees. Second, we address accuracy by normalizing the quartet weights to account for “artificial taxa,” which are introduced during the divide phase so that solutions on subproblems can be combined during the conquer phase (Fig. 1a). We introduce both uniform (n1) and non-uniform (n2) normalization schemes, with the latter up-weighting quartets with leaves labeled by species more closely related to the subproblem (n0 denotes no normalization).

*Results:* We explore the utility of TREE-QMC for multi-locus species tree estimation, benchmarking it against the current leading methods. Our simulation study shows TREE-QMC-n2 is at least as accurate and often more accurate than the dominant method ASTRAL-III [6] (and its improvement FASTRAL [2]), while being highly competitive in terms of runtime (Fig. 1b–d). Moreover,

---

Supported by the State of Maryland.

TREE-QMC-n2 is at least as accurate as wQFM [4], while scaling to much larger data sets. We also re-analyze an avian data set [3], finding that the estimated species trees differ only on short branches suggestive of ILS and that the tree produced by TREE-QMC is closest to the published reference tree.



**Fig. 1.** Subfigure (a) shows how taxa are split at each step in the divide phase, including the introduction of artificial taxa. To compute the quartet weights, the leaves of each gene tree are relabeled by artificial taxa for a given subproblem. Subfigures (b) and (d) show percent species tree error for different model conditions, all with 1000 estimated gene trees (note: \*, \*\*, and \*\*\* indicate a significant difference between methods with  $p < 0.05$ , 0.005, and 0.0005, respectively). Subfigure (c) shows runtime (hours).

Preprint: <https://doi.org/10.1101/2022.06.25.497608>.

## References

- Avni, E., Cohen, R., Snir, S.: Weighted quartets phylogenetics. *Syst. Biol.* **64**(2), 233–242 (2014)
- Dibaenia, P., Tabe-Bordbar, S., Warnow, T.: FASTRAL: improving scalability of phylogenomic analysis. *Bioinformatics* **37**(16), 2317–2324 (2021)
- Jarvis, E.D., Mirarab, S., et al.: Whole-genome analyses resolve early branches in the tree of life of modern birds. *Science* **346**(6215), 1320–1331 (2014)
- Mahbub, M., Wahab, Z., Reaz, R., Rahman, M.S., Bayzid, M.S.: wQFM: highly accurate genome-scale species tree estimation from weighted quartets. *Bioinformatics* **37**(21), 3734–3743 (2021)
- Molloy, E.K., Warnow, T.: To include or not to include: the impact of gene filtering on species tree estimation methods. *Syst. Biol.* **67**(2), 285–303 (2018)
- Zhang, C., Rabiee, M., Sayyari, E., Mirarab, S.: ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinform.* **19**(6), 153 (2018). <https://doi.org/10.1186/s12859-018-2129-y>

# mapquik: Efficient Low-Divergence Mapping of Long Reads in Minimizer Space

Barış Ekim<sup>1,2</sup>, Kristoffer Sahlin<sup>3</sup>, Paul Medvedev<sup>4,5,6</sup>, Bonnie Berger<sup>1,2(✉)</sup>,  
and Rayan Chikhi<sup>7(✉)</sup>

<sup>1</sup> Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts  
Institute of Technology (MIT), Cambridge, MA, USA

`bab@mit.edu`

<sup>2</sup> Department of Mathematics, Massachusetts Institute of Technology (MIT),  
Cambridge, MA, USA

<sup>3</sup> Department of Mathematics, Science for Life Laboratory, Stockholm University,  
Stockholm, Sweden

<sup>4</sup> Department of Computer Science and Engineering, Pennsylvania State University,  
University Park, PA, USA

<sup>5</sup> Department of Biochemistry and Molecular Biology, Penn State University,  
University Park, PA, USA

<sup>6</sup> Huck Institutes of the Life Sciences, Penn State University, University Park, PA,  
USA

<sup>7</sup> Department of Computational Biology, Institut Pasteur, Paris, France  
`rchikhi@pasteur.fr`

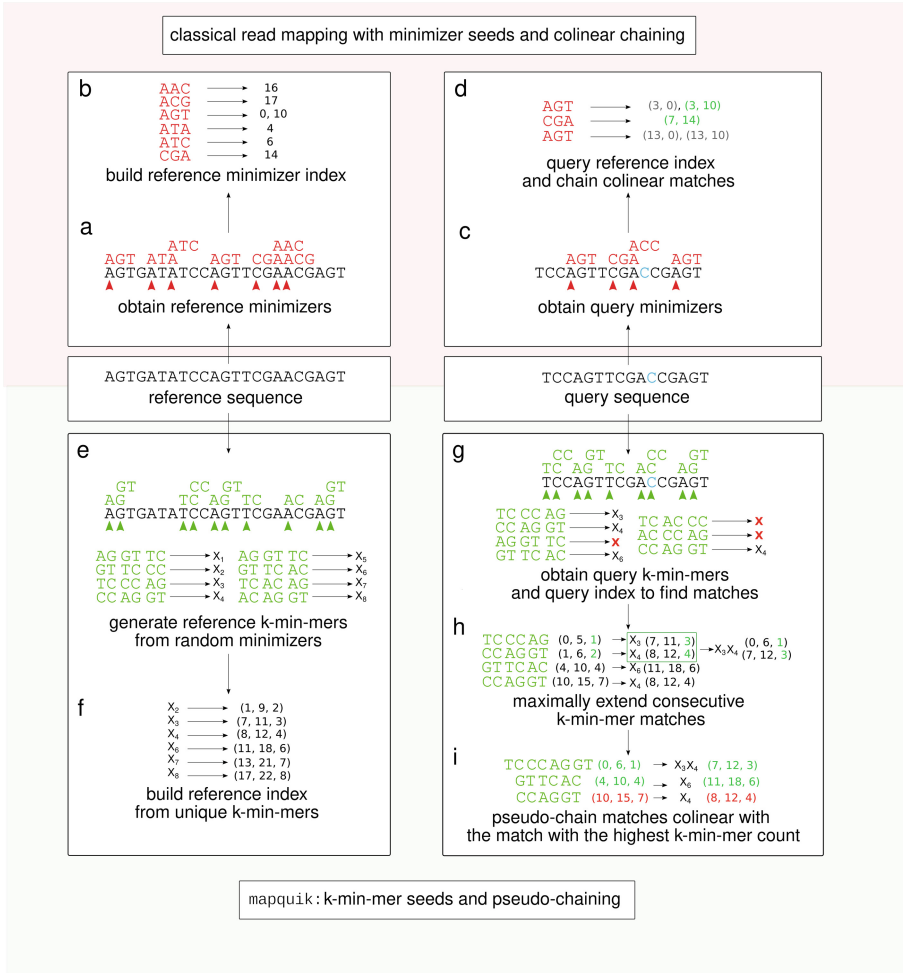
**Keywords:** Genomics · Sequence analysis · Long reads · Read  
mapping ·  $k$ -mers · Minimizer-space ·  $k$ -min-mers

DNA sequencing data continues to progress towards longer reads with increasingly lower sequencing error rates. We focus on the critical problem of mapping, or aligning, low-divergence sequences from long reads (e.g. PacBio HiFi) to a reference genome, which poses challenges in terms of accuracy and computational resources when using cutting-edge read mapping approaches that are designed for all types of alignments. A natural idea would be to optimize efficiency with longer seeds to reduce the probability of extraneous matches; however, contiguous exact seeds quickly reach a sensitivity limit.

We introduce **mapquik**, a novel strategy that creates accurate longer seeds by anchoring alignments through matches of  $k$  consecutively-sampled minimizers ( $k$ -min-mers) and only indexing  $k$ -min-mers that occur once in the reference genome, thereby unlocking ultra-fast mapping while retaining high sensitivity. Figure 1 gives an overview of the algorithmic pipeline of **mapquik**, compared with those of the state-of-the-art methods that use minimizers as seeds.

We demonstrate that **mapquik** significantly accelerates the seeding and chaining steps—fundamental bottlenecks to read mapping—for both the human and maize genomes with >96% sensitivity and near-perfect specificity. On the human genome, for both simulated and real HiFi reads, **mapquik** achieves a 30× speed-up over the state-of-the-art tool **minimap2**, and on the maize genome, a 350× speed-up over **minimap2**, making **mapquik** the fastest mapper to date.





**Fig. 1. Overview of the long-read mapping pipeline using mapquik and comparison with state-of-the-art methods using minimizers as seeds.** State-of-the-art read mappers such as `minimap2` and `Winnowmap2` (top) build an index for a reference sequence by computing window minimizers ( $k = 3, w = 5$ ) (a), and storing the positions of the minimizers in the index (b). In order to map a query sequence using the reference index (top right, nucleotide C in blue denotes a sequencing error), mappers compute the minimizers on the query sequence (c), and find matches between the minimizers of the query and those in the reference index. Once minimizer matches are found, `minimap2` and `Winnowmap2` perform a colinear chaining step to output a high-scoring set of matches, using dynamic programming (d). In contrast, `mapquik` (bottom) indexes reference sequences by generating  $k$ -min-mers,  $k$  consecutive, randomly-selected minimizers of length  $\ell$  ( $k = 3, \ell = 2$ ) (e), and storing only the  $k$ -min-mers that appear exactly once in the reference (f). `mapquik` stores the start and end position of each  $k$ -min-mer, along with the order the  $k$ -min-mers appear in. In order to map a query sequence using the  $k$ -min-mer index, `mapquik` first obtains matches between the query and the reference index by querying the index with each query  $k$ -min-mer (g).  $k$ -min-mer matches are extended if the next immediate pair of  $k$ -min-mers also match (h). Instead of a colinear chaining step, `mapquik` performs a linear-time pseudo-chaining step, which locates matches that are colinear with the match with the highest number of  $k$ -min-mers (i).

These accelerations are enabled not only by minimizer-space seeding but also a novel heuristic  $\mathcal{O}(n)$  pseudo-chaining algorithm, which improves over the long-standing  $\mathcal{O}(n \log n)$  bound. Minimizer-space computation builds the foundation for achieving real-time analysis of long-read sequencing data.

**Full manuscript and software available at:**

<https://www.biorxiv.org/content/10.1101/2022.12.23.521809v2>

<https://github.com/ekimb/mapquik>

**Acknowledgement.** This research is partially based upon work supported by the National Science Foundation under Grants DBI-2138585 and OAC-1931531 (to P.M.). Research reported in this publication was also supported by the National Institutes of Health under Grants NIH R01HG010959 (for B.E. to B.B.), NIH 1R35GM141861 (to B.B.), and NIH R01GM146462 (to P.M.). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

# Deriving Confidence Intervals for Mutation Rates Across a Wide Range of Evolutionary Distances Using FracMinHash

Mahmudur Rahman Hera<sup>1</sup> , N. Tessa Pierce-Ward<sup>2</sup> ,  
and David Koslicki<sup>1,3,4</sup>  

<sup>1</sup> Department of CSE, The Pennsylvania State University, State College, USA  
mbr5797@psu.edu

<sup>2</sup> Department of PHR, University of California, Davis, USA

<sup>3</sup> Department of Biology, The Pennsylvania State University, State College, USA

<sup>4</sup> Huck Institutes of the Life Sciences, The Pennsylvania State University, State College, USA

**Summary:** Sketching-based approaches in recent years have been successfully applied to genomic and metagenomic analyses. For example, Mash [5], is a MinHash [1]-based approach that was used to characterize the similarity between all pairs of RefSeq genomes in less than 30 CPU hours. Importantly, the accuracy and efficiency of sketching approaches can be characterized theoretically. For example, widely used MinHash has been shown to be well-suited to quantify the similarity of sets of roughly the same sizes but falters when sets of very different sizes are compared [4]. To ameliorate this, an approach called “FracMinHash” was recently introduced [2, 3] that allows the sketch size to scale with the size of the underlying data, similar to ModHash dynamic scaling [1]. While there is ample computational evidence for the superiority of FracMinHash when compared to the classic MinHash, particularly when comparing sets of different sizes, no theoretical characterization about the accuracy and efficiency of the FracMinHash approach has yet been given.

In this work, we perform such a theoretical analysis. Our work determines various statistics of FracMinHash and shows its asymptotic normality. We then assume a simple mutation model, in which every nucleotide of a genome sequence of length  $L$  is mutated with a constant mutation rate  $p$ . Finally, we show how to derive a confidence interval for  $p$  given an observed containment index  $C_{frac}$  using the FracMinHash sketching approach, and thus, leading to point estimates and confidence intervals for the Average Nucleotide Identity (ANI).

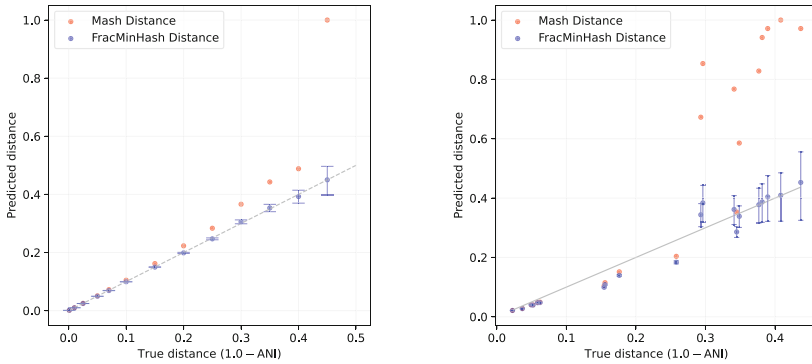
**Experiments and Results:** Our experiments showed that the confidence interval for mutation rate  $p$  is statistically sound. To do this, we took a genome sequence of varying lengths  $L$  and simulated the simple mutation process with various mutation rates. We then observed the containment index  $C_{frac}$  and recorded the percentage of experiments in which the true mutation rate fell within the confidence interval. Table 1 shows the results and demonstrates that the confidence intervals are indeed statistically significant.

---

A full version of this paper with the same title is available as a preprint: <https://doi.org/10.1101/2022.01.11.475870>.

**Table 1.** The percentage of experiments that resulted in the true mutation rate falling within the 95% confidence interval when using various mutation rates across multiple  $k$ -mer sizes and  $L$  values. A scale factor (elaborated in the full version) of  $s = 0.1$  was used. The results show an average of over 10,000 simulations for each setting. N/A entries indicate that the parameters are not particularly interesting and the experiments were not performed.

	$L = 10\text{ K}$			$L = 100\text{ K}$			$L = 1\text{ M}$		
$p = 0.001$	0.1	0.2	0.001	0.1	0.2	0.001	0.1	0.2	
$k = 21$	95.7	94.9	95.0	95.2	95.0	95.3	95.0	94.8	95.1
$k = 51$	95.2	94.6	N/A	95.2	95.5	N/A	95.0	94.8	N/A
$k = 100$	95.1	N/A	N/A	95.2	N/A	N/A	95.1	94.7	N/A



(a) Estimates of evolutionary distances between original and mutated *Staphylococcus* genome

(b) Estimates of evolutionary distances between pairs of real bacterial genomes

**Fig. 1.** Mash distances and FracMinHash estimates of evolutionary distance when (a) introducing point mutations to a *Staphylococcus* genome at a known rate, and (b) between pairs of real bacterial genomes. Error bars indicate the confidence intervals surrounding the FracMinHash estimate calculated using Theorem 8.

We also made a comparison of the confidence interval with the mutation distance computed by Mash [5]. We used a *Staphylococcus* genome, introduced artificial mutations using the simple mutation process, and recorded the Mash distance as well as the FracMinHash distance. We then repeated the same experiment for a set of real genomes that do not follow the simple mutation model at all. The results are shown in Fig. 1, which shows that the confidence interval that we derived is more accurate and precise.

**Acknowledgements.** MR and DK were supported by NSF award No. DMS-1664803 and the NIH grant 1R01GM146462. NP was supported by NSF grants 1711984 and

2018911. The authors express their sincere thanks to Luiz Irber and Paul Medvedev for their invaluable inputs to this manuscript.

## References

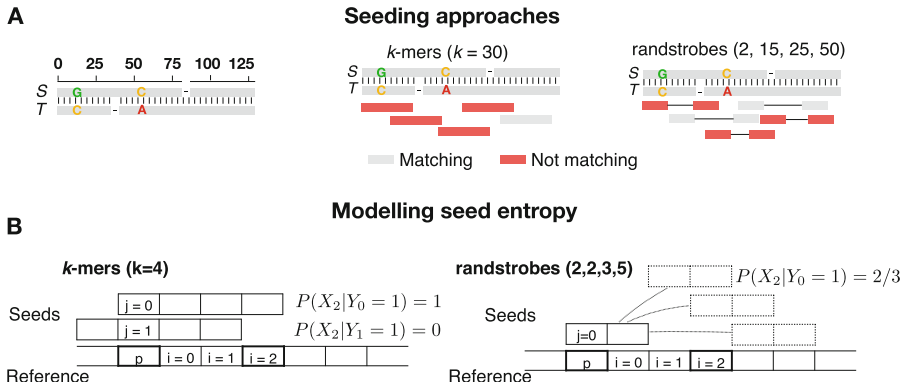
1. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), pp. 21–29. IEEE (1997)
2. Irber, L.C., et al.: Lightweight compositional analysis of metagenomes with fracminhash and minimum metagenome covers. bioRxiv (2022). <https://doi.org/10.1101/2022.01.11.475838>, <https://www.biorxiv.org/content/early/2022/01/12/2022.01.11.475838>
3. Irber Jr, L.C.: Decentralizing indices for genomic data. Ph.D. thesis, University of California, Davis (2020)
4. Koslicki, D., Zabeti, H.: Improving minhash via the containment index with applications to metagenomic analysis. *Appl. Math. Comput.* **354**, 206–215 (2019)
5. Ondov, B.D., et al.: Mash: fast genome and metagenome distance estimation using minhash. *Genome Biol.* **17**(1), 1–14 (2016)

# Entropy Predicts Sensitivity of Pseudo-random Seeds

Benjamin Dominik Maier  and Kristoffer Sahlin  <sup>(✉)</sup>

Department of Mathematics, Science for Life Laboratory, Stockholm University,  
106 91 Stockholm, Sweden  
ksahlin@math.su.se

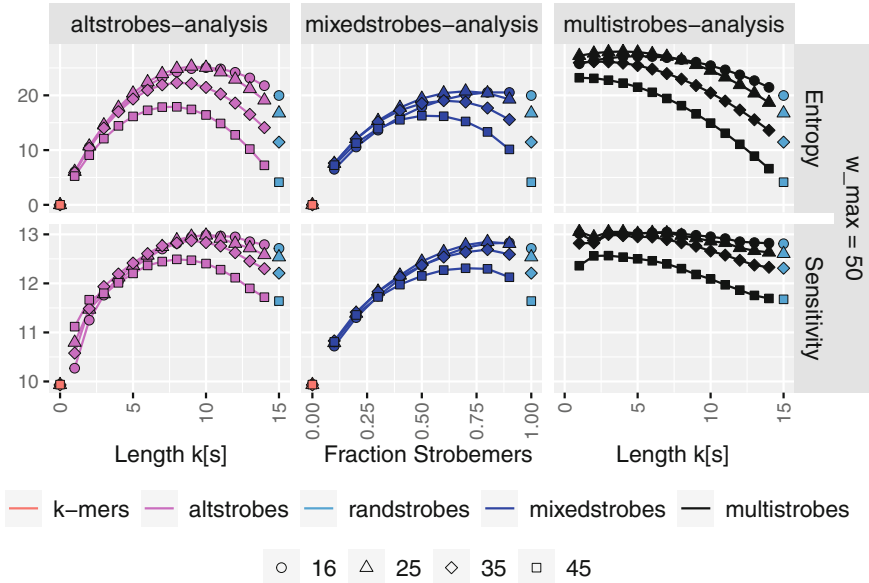
**Motivation.** In sequence similarity search applications such as read mapping, it is desired that seeds match between a read and reference in regions with mutations or read errors (seed sensitivity) but do not produce redundant matches due to repeats, which can lower specificity in, *e.g.*, read mapping.  $K$ -mers are likely the most well-known and used seed construct in bioinformatics, and many studies on, *e.g.*, spaced  $k$ -mers aim to improve sensitivity over  $k$ -mers. Spaced  $k$ -mers are highly sensitive when mutation rate is largely dominated by substitutions, but deteriorates quickly when indels are present. Recently, we developed a pseudo-random seeding construct, strobemers [1] (Fig. 1A), which were empirically demonstrated to have high sensitivity also at high indel rates, but the study lacked a deeper understanding of why.



**Fig. 1.** Panel **A**:  $k$ -mers and randstrobes are sampled over two similar sequences  $S$  and  $T$  different by four mutations. Due to the pseudo-random selection of strobe position in randstrobes, it is likely that at least one of the seeds match in a mutation dense regions. Panel **B**: the core idea behind modelling the entropy of a seed. We compute the probability that a position  $i$  downstream of a position  $p$  is covered by a seed, given that the seed is covering position  $p$ . Randstrobes have a pseudo-random component and therefore higher entropy than  $k$ -mers.

**Methods.** In this study, we demonstrate that the entropy (randomness) of a seed is a good predictor for seed sensitivity. We propose a model to estimate the entropy of a seed (Fig. 1B), and find that seeds with high entropy according to our model in most cases have high match sensitivity. We also present three

new strobemer seed constructs, mixedstrobemes, altstrobemes, and multistrobemes. We use both simulated and biological data to demonstrate that our new seed constructs improve sequence matching sensitivity to other strobemers (Fig. 2). We also implement strobemers into minimap2 and observe slightly faster alignment time and higher accuracy than using  $k$ -mers at various error rates.



**Fig. 2.** For various parametrizations (x-axis), the entropies (y-axis; upper panels) of three novel seeding constructs that we propose, altstrobemes, mixedstrobemes and multistrobemes. The entropy of  $k$ -mers (orange) and randstrobemes (turquoise) are also shown. Each line corresponds to a window parameter in the seeding constructs. Lower three panels shows the corresponding summed seed sensitivity (y-axis), where the sensitivity is measured as producing at least one match between two sequences different by  $m$  mutations in a given window. Here,  $m$  is summed so that the sequences go from 99.5% down to 70% sequence identity. (Color figure online)

**Results.** Our discovered seed randomness-sensitivity relationship gives an explanation as to why some seeds perform better than others, and the relationship provides a framework for designing even more sensitive seeds. In addition, we show that the three new seed constructs are practically useful. Finally, in cases where our entropy model does not predict the observed sensitivity well, we explain why and how to improve the model in future work.

**Code:** [https://github.com/benjamindominikmaier/mixedstrobemes\\_altstrobemes](https://github.com/benjamindominikmaier/mixedstrobemes_altstrobemes).  
**bioRxiv:** <https://doi.org/10.1101/2022.10.13.512198>.

## Reference

1. Sahlin, K.: Effective sequence similarity detection with strobemers. *Genome Res.* **31**(11), 2080–2094 (2021). 34667119[pmid]

# Seed-Chain-Extend Alignment is Accurate and Runs in Close to $O(m \log n)$ Time for Similar Sequences: A Rigorous Average-Case Analysis

Jim Shaw<sup>1(✉)</sup> and Yun William Yu<sup>1,2</sup>

<sup>1</sup> Department of Mathematics, University of Toronto, Toronto, Canada  
{jshaw, ywyu}@math.toronto.edu

<sup>2</sup> Computer and Mathematical Sciences, University of Toronto at Scarborough,  
Scarborough, Canada

**Motivation.** Modern sequence alignment algorithms need to handle massive amounts of data. Given a reference sequence of length  $n$  and a query sequence of length  $m$ , optimal alignment algorithms such as Smith-Waterman run in time  $O(mn)$ ; this is too slow when dealing with large amounts of sequencing data. As a result, these algorithms are eschewed in favor of heuristic methods, which are much faster. A popular heuristic algorithm is the *seed-chain-extend* method, which is employed in popular genome-to-genome alignment and long-read mapping tools such as minimap2 [3]. Unfortunately, these heuristic methods lack theoretical guarantees on the resulting alignment.

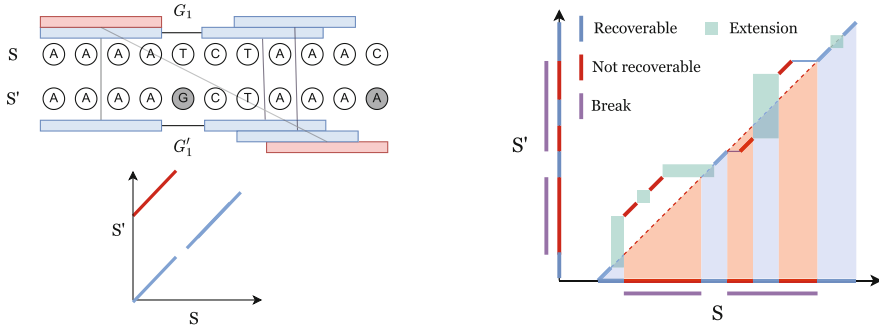
**Theoretical Results.** In this work, we give, to our knowledge, the first theoretical guarantees on the goodness of seed-chain-extend. We do this by using a probabilistic model. Let  $S$  be a uniformly random string of length  $n$ , and  $S'$  be a mutated substring of  $S$  with additional independent point substitutions. We apply seed-chain-extend to  $S, S'$  and average the results to obtain our bounds in expectation. In our model of seed-chain-extend, we: (1) use exact fixed-length k-mer matches of length  $k$  (2) allow for overlapping k-mer anchors in the chain (3) use a linear gap chaining cost [1] (4) perform quadratic time extension between gaps, *not* banded alignment. To measure the goodness of alignment, we propose a metric called *recoverability*, which measures how many homologous bases can be recovered by our resulting alignment. We now state our main result:

**Simplified Theorem 1 (Informal main result; no sketching).** Suppose we are given a uniformly random DNA string of length  $n$  and a mutated substring of length  $m$  where each base is substituted with probability  $\theta$ . If  $\theta < 0.206$  and the longer string is already seeded, then we can choose  $k = \Theta(\log n)$  such that the expected runtime of k-mer seed-chain-extend is  $O(mn^{f(\theta)} \log(n)) = O(mn^{2.43-\theta} \log(n))$ , and in expectation  $\geq 1 - O(\frac{1}{\sqrt{m}})$  of the homologous bases can be recovered from this alignment.

---

J. Shaw was supported by an NSERC CGS-D scholarship. This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2022-03074.





(a) k-mer matches (anchors) under mutations and their corresponding alignment matrix. Blue anchors are “homologous anchors” while red are “spurious anchors”.

(b) Recoverable bases correspond to the bases on the homologous diagonal where a k-mer lies or is accessible by the green DP matrix between gaps. Breaks cover non-recoverable sections.

**Fig. 1.** Recoverability and seed-chain-extend diagrams. k-mer matches correspond to matches in the alignment matrix; chaining is the task of finding an optimal sequence of k-mer matches. Extension corresponds to performing dynamic programming (DP) on the sub-matrix between k-mer anchors in a chain. (Color figure online)

We also prove a version of the above theorem with *sketching*, i.e. subsampling the set of k-mer seeds to reduce chaining time. It turns out that with open syncmer seeds [2], the above theorem still holds, except we can reduce the runtime of chaining by a factor of  $\log n$ , which is useful in practice.

**Experimental Results.** We simulated sequences with point substitutions and aligned them to each other using a basic implementation of a seed-chain-extend aligner. The experimental runtimes were strongly sub-quadratic, as predicted by our theory, and well predicted by our  $O(mn^{f(\theta)} \log n)$  bound. Importantly, our theory, which predicts that sketching does not increase extension runtime asymptotically yet reduces chaining runtime asymptotically, is validated by our results. By sketching with  $\Theta(1/\log n)$  density, we can reduce chaining time by a factor of  $\log n$ , which is  $\sim 20$  on our simulated genomes, yet the extension time plateaus at a  $\sim 3$  times slowdown (Fig. 1).

**Availability.** Our simulations and implementation of seed-chain-extend is available at [https://github.com/bluenote-1577/basic\\_seed\\_chainer/](https://github.com/bluenote-1577/basic_seed_chainer/).



**Preprint.** Our preprint is available at <https://doi.org/10.1101/2022.10.14.512303>.

## References

1. Abouelhoda, M.I., Ohlebusch, E.: Chaining algorithms for multiple genome comparison. *J. Discrete Algorithms* **3**(2–4), 321–341 (2005). <https://doi.org/10.1016/j.jda.2004.08.011>

2. Edgar, R.: Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ* **9**, e10805 (2021). <https://doi.org/10.7717/peerj.10805>
3. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**(18), 3094–3100 (2018). <https://doi.org/10.1093/bioinformatics/bty191>

# Extremely-Fast Construction and Querying of Compacted and Colored de Bruijn Graphs with GGCAT

Andrea Cracco<sup>1</sup><sup>(✉)</sup> and Alexandru I. Tomescu<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Verona, Verona, Italy  
`andrea.cracco@univr.it`

<sup>2</sup> Department of Computer Science, University of Helsinki, Helsinki, Finland  
`alexandru.tomescu@helsinki.fi`

**Abstract.** Compacted de Bruijn graphs are one of the most fundamental data structures in computational genomics. Colored compacted graphs Bruijn graphs are a variant built on a *collection* of sequences, and associate to each  $k$ -mer the sequences in which it appears. We present GGCAT, a tool for constructing both types of graphs. Compared to Cuttlefish 2 (Genome Biology, 2022), the state-of-the-art for constructing compacted de Bruijn graphs, GGCAT has a speedup of up to  $3.4\times$  for  $k = 63$  and up to  $20.8\times$  for  $k = 255$ . Compared to Bifrost (Genome Biology, 2020), the state-of-the-art for constructing the colored variant, GGCAT achieves a speedup of up to  $33.3\times$  for  $k = 27$ . GGCAT is up to  $480\times$  faster than BiFrost for batch sequence queries on colored graphs. GGCAT is based on a new approach merging the  $k$ -mer counting step with the unitig construction step, and on many practical optimizations. GGCAT is implemented in Rust and is freely available at <https://github.com/algbio/ggcat>.

**Keywords:** de Bruijn graph · unitig · Sequencing data ·  $k$ -mers

De Bruijn graphs are one of the most fundamental data structures in computational genomics, appearing in countless applications. To obtain an (edge-centric) de Bruijn graph of order  $k$  for a multiset of strings (usually sequencing reads, or assembled genomes), for every  $k$ -mer in the strings, one adds an edge from the node corresponding to its prefix of length  $k - 1$ , to the node corresponding to its suffix of length  $k - 1$ . De Bruijn graphs usually have associated also an *abundance* threshold  $a$ , so that edges (and thus nodes) are added only for  $k$ -mers appearing at least  $a$  times in the input strings. A *colored* de Bruijn graph [4] is built from a *collection* of datasets, for example different sequencing datasets or different (full) genome sequences. For every  $k$ -mer, colored de Bruijn graphs also store the identifiers (*colors*) of the datasets in which the  $k$ -mer appears. Given a de Bruijn graph, a key problem is to compute the set of all its *maximal unitigs* (a *unitig* is a path whose internal nodes have in-degree and out-degree one); this problem is also called *graph compaction*.

We propose a new tool for constructing compacted, and optionally colored, de Bruijn graphs, GGCAT.<sup>1</sup> As opposed to state-of-the-art tools BCALM2 [1] and Cuttlefish 2 [5], the first idea of GGCAT is to merge the  $k$ -mer counting step with unitig construction, by adding a little more “context” information that allows us to compute valid global unitigs inside each bucket that the input is split into. This avoids the storage of every single  $k$ -mer, since only unitigs built inside the buckets are written to disk. Moreover, as opposed to other tools, these unitigs are lz4-compressed before writing to disk, which allows for a substantial reduction in disk usage for highly repetitive datasets. Second, we avoid a union-find data structure (used by BCALM2) with a new joining step across buckets, that guarantees exact results with very low *expected* running time. Third, we devise a parallelization pipeline that divides the algorithm into smaller execution units (e.g., reading from disk,  $k$ -mer counting,  $k$ -mer extension), thus preventing core stalling due to waiting for data.

For colored graphs we extend our algorithm above with an approach inspired by the state-of-the-art tool BiFrost [3], but with several optimizations that allow improved build and query times, with a comparable colormap size. The main difference w.r.t. BiFrost is to map each color set to a *color set index*, instead of using an individual (compressed) color bitmap for each possible  $k$ -mer. In addition, to store each color set, we compute the difference between consecutive colors and compress them using a run-length encoding. Combined with our improvements over Cuttlefish 2, this leads to a major speed up over BiFrost for colored graphs.

**Acknowledgements.** We are grateful to Massimo Cairo and Romeo Rizzi for initial discussions on this problem.

This work was partially funded by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 851093, SAFEBIO), and partially by the Academy of Finland (grants No. 322595, 352821, 346968).

## References

1. Chikhi, R., Limasset, A., Medvedev, P.: Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* **32**(12), i201–i208 (2016)
2. Cracco, A., Tomescu, A.I.: Extremely-fast construction and querying of compacted and colored de Bruijn graphs with GGCAT. *bioRxiv* (2022). <https://doi.org/10.1101/2022.10.24.513174>
3. Holley, G., Melsted, P.: Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biol.* **21**(1), 1–20 (2020)
4. Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., McVean, G.: De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat. Genet.* **44**(2), 226–232 (2012)
5. Khan, J., Kokot, M., Deorowicz, S., Patro, R.: Scalable, ultra-fast, and low-memory construction of compacted de Bruijn graphs with Cuttlefish 2. *Genome Biol.* **23**(1), 1–32 (2022)

<sup>1</sup> A full version preprint of this extended abstract is available at [2].

# PASTE2: Partial Alignment of Multi-slice Spatially Resolved Transcriptomics Data

Xinhao Liu<sup>1</sup>, Ron Zeira<sup>2</sup>, and Benjamin J. Raphael<sup>1</sup>(✉)

<sup>1</sup> Department of Computer Science, Princeton University, 35 Olden St., Princeton, NJ 08540, USA

[braphael@princeton.edu](mailto:braphael@princeton.edu)

<sup>2</sup> Verily Life Sciences, Tel Aviv, Israel

Spatially resolved transcriptomics (SRT) technologies measure mRNA expression at thousands of locations in a tissue slice. However, nearly all SRT technologies measure expression in two dimensional slices extracted from a three-dimensional tissue, thus losing information that is shared across multiple slices from the same tissue. Integrating SRT data across multiple slices can help recover this information and enables innovative downstream tasks such as 3D differential expression analysis, 3D cell-cell communication, and 3D clustering. Many existing methods for integrating SRT slices either do not use spatial information or assume that the morphology of the tissue is largely preserved across slices, an assumption that is often violated due to biological or technical reasons. A recent method, PASTE [3], aligns adjacent SRT slices according to both transcriptomic and spatial similarity, but assumes that adjacent slices overlap over the full 2D assayed region, with similar field of view and similar number and proportion of cell types.

We introduce PASTE2, a method to perform *partial* alignment and 3D reconstruction of multi-slice SRT datasets, allowing only partial overlap between slices and/or slice-specific cell types. PASTE2 is based on a novel *partial Fused Gromov-Wasserstein (partial-FGW) optimal transport* formulation, which we optimize using a conditional gradient algorithm. PASTE2 includes a model selection procedure to estimate the fraction of overlap between slices, and optionally uses information from histological images that accompany some SRT experiments. PASTE2 also provides a generalized Procrustes analysis method for 3D spatial reconstruction of the tissue from partially aligned 2D slices.

We demonstrate PASTE2's advantages on both simulated and real SRT datasets. We show on simulated data that PASTE2 achieves accurate alignment when slices do not fully overlap. On SRT dataset from the human dorsolateral prefrontal cortex [1], we show that PASTE2 computes more accurate alignments than competing methods, and the use of histological images can further improve the alignment. We further use PASTE2 to reconstruct a 3D map of gene expression of a *Drosophila* embryo using a 16 slice Stereo-seq SRT dataset [2]. PASTE2 enables detailed studies of 3D spatial gene expression across a wide range of biological applications.

The PASTE2 software is available at <https://github.com/raphael-group/paste2>.

## References

1. Maynard, K.R., et al.: Transcriptome-scale spatial gene expression in the human dorsolateral prefrontal cortex. *Nat. Neurosci.* **24**(3), 425–436 (2021)
2. Wang, M., et al.: High-resolution 3D spatiotemporal transcriptomic maps of developing drosophila embryos and larvae. *Dev. Cell* **57**(10), 1271–1283 (2022)
3. Zeira, R., Land, M., Strzalkowski, A., Raphael, B.J.: Alignment and integration of spatial transcriptomics data. *Nat. Methods* **19**(5), 567–575 (2022)

# FastRecomb: Fast Inference of Genetic Recombination Rates in Biobank Scale Data

Ardalan Naseri<sup>1</sup>, William Yue<sup>1</sup>, Shaojie Zhang<sup>2</sup>, and Degui Zhi<sup>1</sup>(✉)

<sup>1</sup> School of Biomedical Informatics, University of Texas Health Science Center at Houston, Houston 77030, USA

{ardalan.naseri, degui.zhi}@uth.tmc.edu

<sup>2</sup> Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA

A genetic map for a population or a species contains the locations of genetic markers or variant sites in relation to one another based on the probability of recombination, rather than a physical location along each chromosome. An accurate genetic map, which is an estimation of the recombination rates along a chromosome, serves as the foundation for genetic studies like gene mapping, population genetics, and genealogical studies. Given that recombination rates differ between populations, the estimation of population-specific genetic maps is crucial for advancing genetic research, particularly in diverse populations. The traditional approach to infer the recombination rates is to use genotype data from a large number of parent-offspring pairs to capture an adequate number of meiotic crossover events [1, 2]. An alternative approach is to use recombination event signals that are dispersed among individuals in population samples [3, 4]. IBDrecomb [4] leverages the recent development of fast Identity-by-Descent (IBD) segment calling methods. IBD segments are identical DNA fragments that are inherited from a common ancestor. Under the assumption that the IBD segment boundaries were caused by recombination events, IBDrecomb counts the IBD segment boundaries and generates a map iteratively using the normalized counts of the IBD segments. However, IBDrecomb is not adequately efficient as it requires outputting all pairwise IBD segments, which is not conducive for biobank-scale cohorts.

We present FastRecomb, a novel method that efficiently identifies potential recombination breakpoints in very large cohorts using positional Burrows-Wheeler transform (PBWT) [5]. For short IBD segments, we assume that PBWT can be used as a lightweight alternative to conventional IBD calling methods if the error rate is low. Our approach detects blocks of haplotype matches, such as IBD segments, and takes into account the number of mismatches at each site rather than counting all pairwise IBD segments. The number of mismatches (or minor alleles) within each matching block corresponds to the number of diverging haplotypes. In fact, FastRecomb counts the number of diverging haplotypes at each variant site to infer the recombination rates. Blocks of haplotype matches can be detected in linear time using PBWT [6]. Additionally, a pre-processing step [7] is applied in an effort to smooth the haplotype panel by using blocks of haplotype matches to reduce the impact of genotyping errors within a panel. It is anticipated that the more individuals available in the panel, the closer the

computed rates mirror the actual values. With the growing availability of genetic data in large-scale cohorts, we anticipate our method will enable an accurate and population-specific estimation of recombination rates in diverse populations as it has the capability of being applied to samples containing millions of individuals without requiring the use of extensive computational resources.

The accuracy of recombination rate estimation of FastRecomb improves with the increasing number of haplotypes while the running time increases linearly with the sample size. FastRecomb is also robust against genotyping errors as its performance was not affected by increasing the error rates from 0 to 0.2%. FastRecomb outperformed LDhat and IBDrecomb in mid-regions (excluding 5 Mbps from both sides) while one million samples for FastRecomb were used. Other tools are not tractable for such large sample sizes. We applied FastRecomb on four different subsets of UK Biobank data. Approximately 1 CPU hour and 55 MB memory were used to estimate the recombination rates for chromosome 20 using 458,677 individuals.

The current implementation of FastRecomb does not treat the end regions differently. Hence, the estimated rates for the end-regions are not as accurate as those of the mid-region. In summary, FastRecomb unleashes the potential of large-scale haplotype panels by providing an efficient approach for estimating population-specific genetic maps. A preprint of the full paper is available at <https://www.biorxiv.org/content/10.1101/2023.01.09.523304v1>. Source code is available at <https://github.com/ZhiGroup/FastRecomb>.

## References

1. Kong, A., et al.: Fine-scale recombination rate differences between sexes, populations and individuals. *Nature* **467**, 1099–1103 (2010)
2. Halldorsson, B., et al.: Characterizing mutagenic effects of recombination through a sequence-level genetic map. *Science* **363** (2019)
3. McVean, G., Myers, S., Hunt, S., Deloukas, P., Bentley, D., Donnelly, P.: The fine-scale structure of recombination rate variation in the human genome. *Science* **304**, 581–584 (2004)
4. Zhou, Y., Browning, B., Browning, S.: Population-specific recombination maps from segments of identity by descent. *Am. J. Hum. Genet.* **107**, 137–148 (2020)
5. Durbin, R.: Efficient haplotype matching and storage using the positional Burrows-Wheeler transform (PBWT). *Bioinformatics* **30**, 1266–1272 (2014)
6. Naseri, A., Yue, W., Zhang, S., Zhi, D.: Efficient haplotype block matching in Bi-directional PBWT. In: 21st International Workshop on Algorithms in Bioinformatics (WABI 2021), vol. 201, pp. 19:1–19:13 (2021)
7. Yue, W., Naseri, A., Wang, V., Shakya, P., Zhang, S., Zhi, D.: P-smoother: efficient PBWT smoothing of large haplotype panels. *Bioinform. Adv.* **2**, vbac045 (2022)



# Efficient Taxa Identification Using a Pangenome Index

Omar Ahmed<sup>(✉)</sup>, Massimiliano Rossi, Christina Boucher, and Ben Langmead

Baltimore, USA

**Abstract.** Tools that classify sequencing reads against a database of reference sequences require efficient index data structures. The  $r$ -index is a compressed full-text index that answers substring presence/absence, count and locate queries in space proportional to the amount of distinct sequence in the database:  $\mathcal{O}(r)$  space where  $r$  is the number of Burrows-Wheeler runs. To date, the  $r$ -index has lacked the ability to quickly classify matches according to which reference sequences (or sequence groupings, i.e. taxa) a match overlaps. We present new algorithms and methods for solving this problem. Specifically, given a collection  $\mathcal{D}$  of  $d$  documents  $\mathcal{D} = \{T_1, T_2, \dots, T_d\}$  over an alphabet of size  $\sigma$ , we extend the  $r$ -index with  $\mathcal{O}(rd)$  additional words to support document listing queries for a pattern  $S[1..m]$  that occurs in  $ndoc$  documents in  $\mathcal{D}$  in  $\mathcal{O}(m \log \log_w \sigma + ndoc)$  time and  $\mathcal{O}(rd)$  space, where  $w$  is the machine word size. Applied in a bacterial mock community experiment, our method is up to 3 times faster than a comparable method that uses the standard  $r$ -index locate queries. We show that our method classifies both simulated and real nanopore reads at the strain level with higher accuracy compared to other approaches. Finally, we present strategies for compacting this structure in applications where read lengths or match lengths can be bounded. Our source code can be found at <https://github.com/oma219/docprofiles>, and our experimental code can be found at <https://github.com/oma219/docprof-experiments>.

## 1 Introduction

Metagenomic read classification allows researchers to study organisms present in an environmental sample. Tools like Kraken 2 [13] and Centrifuge [6] accomplish this using an index of the reference sequences. Kraken 2 [13] builds a compact hash table that maps minimizer sequences onto the taxonomic lowest-common ancestor of the genomes it occurs in. Centrifuge [6] uses an FM-index [4] to find substring matches which are combined to make classification decisions. But as databases of reference sequences continue to grow, these tools encounter difficulties with scaling and accuracy. Nasko et al. [9] showed that the specificity of  $k$ -mer based approaches like Kraken 2 can suffer as the reference database (i.e., RefSeq) grows, since the addition of new sequences causes more  $k$ -mers (or minimizers) to co-occur in distant parts of the taxonomy. The FM-index at the core of Centrifuge does not naturally scale to pangenomes; rather, it requires

an initial work-intensive step that compresses the genomes in a way that elides some of the underlying genetic variation.

The  $r$ -index [5] is a successor to the FM-index that indexes repetitive texts using  $\mathcal{O}(r)$ -space, where  $r$  is the number of runs in the text’s Burrows-Wheeler Transform (BWT). Since  $r$  grows only with the amount of *distinct* sequence in the collection, the  $r$ -index scales naturally to large pangenomes and reference databases like the ones used for taxonomic classification. Since it is a full-text index, the  $r$ -index can find matches of any length, unconstrained by a particular choice of  $k$ -mer length.

While the  $r$ -index has already been applied to pangenomic pattern-matching [7, 11] and binary classification [2], it has so far lacked the ability to solve multi-class classification problems in an accurate and efficient manner. A straightforward approach would be to use standard backward search in the  $r$ -index, then use locate queries to locate the offsets in the concatenated text where the pattern occurs. These offsets can then be cross-referenced with another structure to determine which documents they occur in. This requires an amount of work proportional to the number of occurrences  $occ$ , which is expensive, particularly for repetitive matches against a pangenome.

We hypothesized that extending the  $r$ -index to multi-class classification could be accomplished by augmenting it with efficient facilities for *document listing*, i.e. the ability to report all the reference sequences (documents) where a particular pattern occurs. A document—which we will sometimes call a “class”—could consist of a single genome or a collection of related genomes.

An early study by Muthukrishnan [8] described a specialized index for document listing consisting of a generalized suffix tree and a document array. It provided  $\mathcal{O}(m + ndoc)$  queries, where  $m$  is the length of the pattern and  $ndoc$  is the number of distinct documents it occurs in. But this came at the cost of  $\mathcal{O}(n \log n)$  bits of space, where  $n$  is the total length of the texts, which is impractical for large pangenome databases. Sadakane [12] improved on this by introducing a new succinct document array representation and building on succinct representations of suffix trees and arrays. He showed how to reduce the index size to  $|CSA| + \mathcal{O}(n)$  bits, where  $|CSA|$  is the size of the compressed suffix array using statistical compression with an increased time complexity of  $\mathcal{O}(m + ndoc \cdot \log n)$ , a high cost for repetitive text collections [3]. Later efforts further reduced the required space using grammar-compression [3] and relative Lempel-Ziv compression [10].

We present a new method that solves the document listing problem in  $\mathcal{O}(m \log \log_w \sigma + ndoc)$ -time and  $\mathcal{O}(rd)$ -space using the  $r$ -index. Importantly, we also show how to use the prefix-free parsing process to build the profile simultaneously with the BWT. This document-array structure can be sampled and stored at the run boundaries of the BWT, yielding a space complexity of  $\mathcal{O}(rd)$ . At query time, after performing backward search for a pattern, we can report the document listing by simply examining the current document array profile which is an array of  $d$  integers—as opposed to performing a query for each occurrence of a pattern. We also discuss practical optimizations that can

be used to reduce the space usage of this data structure even further in the context of metagenomic read classification.

We compare the query time and index size for our approach to an alternative that uses the standard  $r$ -index locate query to report document listings. We show that as a database of bacterial genomes grows larger, our approach is much faster than the locate-query approach, up to 3.2 times faster. Furthermore, we show when attempting to classify different strains of *Escherichia coli* and *Salmonella enterica*, that using our document array profiles yields higher recall and precision compared to SPUMONI 2's sampled document array [1]. Finally, we believe that our theoretical guarantees will prove useful for the community by allowing read classification to be compared in a grounded manner that complements practical evaluation.

## References

1. Ahmed, O., Rossi, M., Gagie, T., Boucher, C., Langmead, B.: SPUMONI 2: Improved pangenome classification using a compressed index of minimizer digests. *bioRxiv* (2022)
2. Ahmed, O., Rossi, M., Kovaka, S., Schatz, M.C., Gagie, T., Boucher, C.: Pangenomic matching statistics for targeted nanopore sequencing. *iScience* **24**(6), 102696 (2021)
3. Cobas, D., Navarro, G.: Fast, small, and simple document listing on repetitive text collections. In: Brisaboa, N.R., Puglisi, S.J. (eds.) SPIRE 2019. LNCS, vol. 11811, pp. 482–498. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-32686-9\\_34](https://doi.org/10.1007/978-3-030-32686-9_34)
4. Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS), pp. 390–398. IEEE (2000)
5. Gagie, T., Navarro, G., Prezza, N.: Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *J. ACM (JACM)* **67**(1), 1–54 (2020)
6. Kim, D., Song, L., Breitwieser, F.P., Salzberg, S.L.: Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res.* **26**(12), 1721–1729 (2016)
7. Kuhnle, A., Mun, T., Boucher, C., Gagie, T., Langmead, B., Manzini, G.: Efficient construction of a complete index for pan-genomics read alignment. *J. Comput. Biol.* **27**(4), 500–513 (2020)
8. Muthukrishnan, S.: Efficient algorithms for document retrieval problems. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 657–666 (2002)
9. Nasko, D.J., Koren, S., Phillippy, A.M., Treangen, T.J.: RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. *Genome Biol.* **19**(1), 1–10 (2018)
10. Puglisi, S.J., Zhukova, B.: Document retrieval hacks. In: Proceedings of the International Symposium on Experimental Algorithms (SEA), pp. 12:1–12:12 (2021)
11. Rossi, M., Oliva, M., Langmead, B., Gagie, T., Boucher, C.: Moni: a pangenomic index for finding maximal exact matches. *J. Comput. Biol.* **29**(2), 169–187 (2022)
12. Sadakane, K.: Succinct data structures for flexible text retrieval systems. *J. Discrete Algorithms* **5**(1), 12–22 (2007)
13. Wood, D.E., Lu, J., Langmead, B.: Improved metagenomic analysis with Kraken 2. *Genome Biol.* **20**(1), 1–13 (2019)

# Vector-Clustering Multiple Sequence Alignment: Aligning into the Twilight Zone of Protein Sequence Similarity with Protein Language Models

Claire McWhite<sup>1</sup>  and Mona Singh<sup>1,2</sup> 

<sup>1</sup> Lewis-Sigler Institute for Integrative Genomics,  
Princeton University, Princeton, USA

`cmcwhite@princeton.edu`, `mona@cs.princeton.edu`

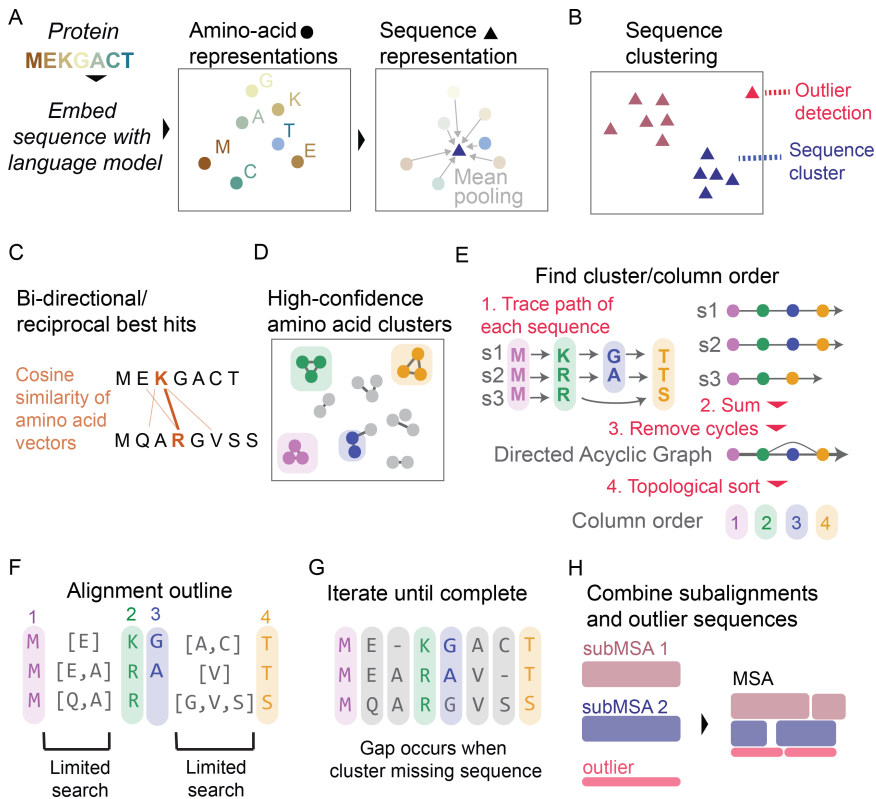
<sup>2</sup> Department of Computer Science, Princeton University, Princeton, USA

**Summary.** Multiple sequence alignment (MSA) is a critical step in the study of protein sequence and function. Typically, MSA algorithms progressively align pairs of sequences and combine these alignments with the aid of a guide tree. These alignment algorithms use scoring systems based on substitution matrices to measure amino-acid similarities. While successful, standard methods struggle on sets of proteins with low sequence identity—the so-called twilight zone of protein alignment. For these difficult cases, another source of information is needed. Protein language models are a powerful new approach that leverage massive sequence datasets to produce high-dimensional contextual embeddings for each amino acid in a sequence. These embeddings have been shown to reflect physicochemical and higher-order structural and functional attributes of amino acids within proteins. Here, we present a novel approach to align multiple sequences, based on clustering and ordering amino acid contextual embeddings. Our method for aligning semantically consistent groups of proteins circumvents the need for many standard components of MSA algorithms, avoiding initial guide tree construction, intermediate pairwise alignments, gap penalties, and substitution matrices. The added information from contextual embeddings leads to higher accuracy alignments for structurally similar proteins with low amino-acid similarity. We anticipate that protein language models will become a fundamental component of the next generation of algorithms for generating MSAs.

**Software Availability:** <https://github.com/clairemwhite/vcmsa>.

**Methods.** A brief overview of our method, vcMSA, is given in Fig. 1.

**Results.** We apply vcMSA to align a benchmark set of 147 protein families, and evaluate the quality of each alignment by determining the percent of columns in the output alignment that fully match columns in the reference gold standard alignment. We compare vcMSA to nine MSA algorithms and find that vcMSA tends to produce higher quality alignments than previous methods. Notably, vcMSA has good performance when considering sequence families with highly divergent sequences, precisely the scenario that current methods have the most difficulty on.



**Fig. 1. Overview of vcMSA algorithm.** (A) Proteins are embedded using a protein language model to produce vector representations of each amino acid, and the mean of these amino acid embeddings is taken to produce a sequence-level representation. (B) We cluster sequence representations, and detect outlier sequences. (C) For each sequence cluster, we determine bi-directional/reciprocal best hits of cosine-similarity between pairs of amino acids in different sequences. (D) From a network built from reciprocal best hits, we determine confident clusters of amino acids, corresponding to columns in the MSA. (E) To determine column order, we trace the path of each sequence through clusters and combine all paths into one network, taking edge weights from the number of sequences which traverse between the pairs of clusters. We trim any clusters which cause cycles, and use a topological sort of the resulting directed acyclic graph to find column order. (F) Clusters/columns limit scope of search for unplaced amino acids. (G) We iterate limited searches until all amino acids are placed. Gaps in the alignment occur when a cluster does not contain an amino acid from a sequence. (H) We combine alignments from each sequence cluster and outliers in the final MSA.

**Full Paper:** A complete manuscript describing this work can be accessed at: <https://www.biorxiv.org/content/XX.XXXX/2022.01.31.XXXXXv1>.

# Single-Cell Methylation Sequencing Data Reveal Succinct Metastatic Migration Histories and Tumor Progression Models

Yuelin Liu<sup>1,4,6</sup>, Xuan Cindy Li<sup>1,3</sup>, Farid Rashidi Mehrabadi<sup>1,7</sup>,  
Alejandro A. Schäffer<sup>1</sup>, Drew Pratt<sup>2</sup>, David R. Crawford<sup>1,5</sup>, Salem Malikić<sup>1</sup>,  
Erin K. Molloy<sup>4</sup>, Vishaka Gopalan<sup>1</sup>, Stephen M. Mount<sup>5</sup>, Eytan Ruppin<sup>1</sup>,  
Kenneth Aldape<sup>2</sup>, and S. Cenk Sahinalp<sup>1</sup>(✉)

<sup>1</sup> Cancer Data Science Laboratory, Center for Cancer Research,  
National Cancer Institute, NIH, Bethesda, MD 20892, USA  
[cenk.sahinalp@nih.gov](mailto:cenk.sahinalp@nih.gov)

<sup>2</sup> Laboratory of Pathology, Center for Cancer Research, National Cancer Institute,  
NIH, Bethesda, MD 20892, USA

<sup>3</sup> Program in Computational Biology, Bioinformatics, and Genomics,  
University of Maryland, College Park, MD 20742, USA

<sup>4</sup> Department of Computer Science, University of Maryland,  
College Park, MD 20742, USA

<sup>5</sup> Department of Cell Biology and Molecular Genetics, University of Maryland,  
College Park, MD 20742, USA

<sup>6</sup> Center for Bioinformatics and Computational Biology, University of Maryland,  
College Park, MD 20742, USA

<sup>7</sup> Department of Computer Science, Indiana University,  
Bloomington, IN 47408, USA

The recent rise of single-cell sequencing technology empowers more accurate tumor lineage inference by allowing the examination of intratumor heterogeneity at a cellular resolution. However, since single-cell sequencing data is derived from an incredibly limited amount of genetic material, the signals obtained are more scarce and unstable than those from bulk sequencing DNA methylation, the addition of a methyl group to cytosine, which results in the formation of 5-methylcytosine (5mC) especially in the context of CpG sites, is an epigenetic marker that has been extensively studied for its role in regulating gene expression and maintaining cellular memory. Prior research suggests the changes in methylation status at CpG sites in cancer cells may provide a greater amount of observable evidence for tumor evolution than single-nucleotide variations (SNVs) or copy number aberrations (CNAs) [2, 5]. That said, constructing tumor lineage with single-cell methylation data has two major challenges. First, single-cell methylation data exhibit high level of sparsity. Second, not all CpG sites have their methylation statuses stably retained during in tumor evolution [4].

In this work, we introduce **Sgootr** (Single-cell Genomic methylatiOn tumOr Tree Reconstruction, Fig. 1), the first distance-based computational method to jointly select informative CpG sites and reconstruct tumor lineages from single-cell methylation data. **Sgootr** consists of five key components: (i) biclustering of

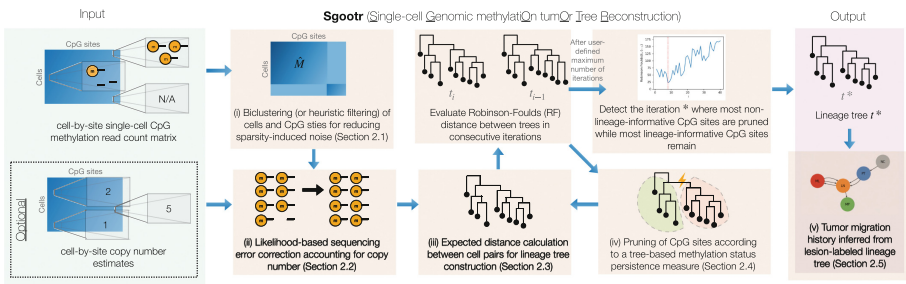
---

Y. Liu, X. C. Li—These authors contributed equally.

cells and sites for reducing sparsity-induced noise; (ii) likelihood-based sequencing error correction accounting for copy number variation; (iii) expected distance calculation between cell pairs for tree construction; (iv) pruning of CpG sites according to a tree-based methylation status persistence measure; and (v) inference of migration history from the lesion-labeled tree. Components (iii) and (iv) are iteratively applied.

Application of **Sgootr** on simulated data shows that it can more accurately capture the evolution history of a tumor compared to a baseline approach. Application of **Sgootr** to a multiregionally-sampled metastatic CRC scBS-seq data set [1] and a multiregionally-sampled GBM MscRRBS patient sample [3] reveals tumor progression models and metastatic migration histories simpler than previously reported. A comparison of **Sgootr** against alternative lineage-reconstruction and site-selection approaches on the same datasets shows that it infers a simpler migration history in a shorter amount of time. Interestingly lineage-informative CpG sites identified by **Sgootr** appear to be primarily in inter-CpG island (CGI) regions, as opposed to CGI's which have been the main regions of interest in genomic methylation-related analyses.

**Sgootr** is implemented as a **Snakemake** workflow, available at <https://github.com/algo-cancer/Sgootr>. The full manuscript is available on bioRxiv at <https://www.biorxiv.org/content/10.1101/2021.03.22.436475v2>.



**Fig. 1. Overview of Sgootr.** Sgootr leverages single-cell methylation sequencing data from tumor samples, incorporating copy number information when available, to jointly infer a single-cell tumor lineage tree and identify CpG sites that may harbor lineage-informative methylation changes.

## References

1. Bian, S., et al.: Single-cell multiomics sequencing and analyses of human colorectal cancer. *Science* **362**(6418), 1060–1063 (2018). <https://doi.org/10.1126/science.aao3791>
2. Biezuner, T., et al.: A generic, cost-effective, and scalable cell lineage analysis platform. *Genome Res.* **26**(11), 1588–1599 (2016). <https://doi.org/10.1101/gr.202903.115>

3. Chaligne, R., et al.: Epigenetic encoding, heritability and plasticity of glioma transcriptional cell states. *Nat. Genet.* **53**(10), 1469–1479 (2021). <https://doi.org/10.1038/s41588-021-00927-7>
4. Meir, Z., Mukamel, Z., Chomsky, E., Lifshitz, A., Tanay, A.: Single-cell analysis of clonal maintenance of transcriptional and epigenetic states in cancer cells. *Nat. Genet.* **52**(7), 709–718 (2020). <https://doi.org/10.1038/s41588-020-0645-y>
5. Ushijima, T., Watanabe, N., Okochi, E., Kaneda, A., Sugimura, T., Miyamoto, K.: Fidelity of the methylation pattern and its variation in the genome. *Genome Res.* **13**(5), 868–874 (2003)



# Information-Theoretic Classification Accuracy: A Criterion That Guides Data-Driven Combination of Ambiguous Outcome Labels in Multi-class Classification

Chihao Zhang<sup>1</sup>, Yiling Elaine Chen<sup>2</sup>, Shihua Zhang<sup>1</sup>, and Jingyi Jessica Li<sup>2(✉)</sup>

<sup>1</sup> Academy of Mathematics and Systems Science, Chinese Academy of Science, Beijing 100190, China

<sup>2</sup> Department of Statistics, University of California, Los Angeles, CA 90095, USA  
lijy03@g.ucla.edu

In real-world multiclass classification problems, outcome labeling ambiguity and subjectivity would deteriorate prediction accuracy. Although outcome labels are often combined in an ad hoc way to train algorithms in practice, there lacks a principled approach to guide the combination by any optimality criterion.

Finding an “optimal” label combination is nontrivial. The reason is that even if prediction is completely random (“random guess”), i.e., assigning data points with random labels irrespective of features, prediction accuracy would still be boosted by label combination. In such an extreme case, the increase in prediction accuracy does not outweigh the decrease in classification resolution. Hence, our rationale is that label combination must be guided by a criterion that reasonably balances prediction accuracy and classification resolution.

Motivated by this rationale, we propose ITCA, a criterion from an information theory perspective to define the optimal balance between prediction accuracy and classification resolution.

Let  $(\mathbf{X}, Y) \sim \mathcal{P}$  be a random pair where  $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^d$  is a feature vector,  $Y \in [K_0] := \{1, \dots, K_0\}$  is a class label indicating one of  $K_0$  observed classes that are potentially ambiguous, and  $\mathcal{P}$  is the joint distribution of  $(\mathbf{X}, Y)$ . For a fixed positive integer  $K (< K_0)$ , a class combination is represented by an onto mapping:  $\pi_K : [K_0] \rightarrow [K]$ . For example, if  $K_0 = 4$  classes are combined into  $K = 3$  classes by merging the original classes 3 and 4, then  $\pi_3(1) = 1$ ,  $\pi_3(2) = 2$ ,  $\pi_3(3) = 3$ , and  $\pi_3(4) = 3$ . Given a class combination  $\pi_K$ , a classification algorithm  $\mathcal{C}$ , and a training dataset  $\mathcal{D}_t$ , we denote by  $\phi_{\pi_K}^{\mathcal{C}, \mathcal{D}_t} : \mathcal{X} \rightarrow [K]$  a multi-class classifier trained by  $\mathcal{C}$  on  $\mathcal{D}_t$  to predict  $K$  combined classes. Then ITCA is defined on a validation dataset  $\mathcal{D}_v$ :

$$\text{ITCA}(\pi_K; \mathcal{D}_t, \mathcal{D}_v, \mathcal{C}) := \sum_{k=1}^K \left[ -p_{\pi_K}^{\mathcal{D}_v}(k) \cdot \log p_{\pi_K}^{\mathcal{D}_v}(k) \right] \cdot \frac{\sum_{(\mathbf{X}_i, Y_i) \in \mathcal{D}_v} \mathbb{1}(\phi_{\pi_K}^{\mathcal{C}, \mathcal{D}_t}(\mathbf{X}_i) = k, \pi_K(Y_i) = k)}{1 \vee \sum_{(\mathbf{X}_i, Y_i) \in \mathcal{D}_v} \mathbb{1}(\pi_K(Y_i) = k)}, \quad (1)$$

where  $p_{\pi_K}^{\mathcal{D}_v}(k) := \frac{1}{|\mathcal{D}_v|} \sum_{(\mathbf{X}_i, Y_i) \in \mathcal{D}_v} \mathbb{1}(\pi_K(Y_i) = k)$  is the proportion of  $\pi_K$ 's  $k$ -th combined class in  $\mathcal{D}_v$ , and the expression  $a \vee b$  means the maximum of  $a$  and  $b$ , preventing the denominator of (1) from being zero. In (1),  $\pi_K$ 's  $k$ -th combined class has weight  $-p_{\pi_K}^{\mathcal{D}_v}(k) \cdot \log p_{\pi_K}^{\mathcal{D}_v}(k)$ , i.e., the class's contribution to

the “classification resolution” defined as the entropy of the  $K$  combined class labels:  $\sum_{k=1}^K -p_{\pi_K}^{\mathcal{D}_v}(k) \cdot \log p_{\pi_K}^{\mathcal{D}_v}(k)$ . Note that (1) becomes the classification accuracy (ACC) if we set the  $k$ -th combined class’s weight as  $p_{\pi_K}^{\mathcal{D}_v}(k)$ . Hence, ITCA overweighs minor classes so it would be less dominated by major classes than ACC is. We also propose two heuristic search strategies, i.e., greedy search and breadth-first search (BFS), to find the optimal class combination that maximizes ITCA.

We evaluate ITCA extensively on both simulated and real-world data.

First, we demonstrate the effectiveness of ITCA and the two search strategies in multiple simulation settings. ITCA consistently outperforms alternative class combination criteria (including our newly proposed alternative criteria and the commonly used ACC and mutual information) and clustering-based class combination algorithms (Figs. 1–3 in Supplementary Material). We also analyze the theoretical properties of ITCA at the population level with the oracle algorithm and the linear discriminant analysis (LDA) algorithm. As expected, when used with the oracle algorithm, ITCA has a much stronger ability to find the true class combination than when it is used with the LDA algorithm (Fig. 4 in Supplementary Material). We also find that, when the LDA is used as a soft classification algorithm, ITCA is more likely to find the true class combination (Fig. 5 in Supplementary Material).

Second, we apply ITCA to improve prediction accuracy in a disease prognosis application. We use the random forest classification algorithm to predict rehabilitation outcomes of traumatic brain injury patients, which were evaluated and recorded by physical therapists for 17 activities with seven Functional Independence Measure levels: 1 (patient requires total assistance to perform an activity) to 7 (patient can perform the activity with complete independence). We apply ITCA as a data-driven approach to guide the combination of outcome levels for each activity. Compared with the expert-suggested combination and the hierarchical-clustering-based combinations, ITCA-guided combinations consistently lead to more balanced levels and more significant improvement in prediction accuracy from random guess (Fig. 6 in Supplementary Material).

Third, we demonstrate how ITCA identifies biologically similar cell types in a single-cell RNA-seq dataset of hydra. Since our goal is to discover ambiguous cell types instead of achieving high prediction accuracy, we choose the LDA, a weak classification algorithm. In our results, ITCA suggests the combination of two cell types with similar labels and gene expression profiles. Moreover, ITCA with greedy search constructs a cell type hierarchy more biologically meaningful than the result of hierarchical clustering (Fig. 7 in Supplementary Material).

The full version of this manuscript is available at [1].

## Reference

1. Zhang, C., Chen, Y.E., Zhang, S., Li, J.J.: Information-theoretic classification accuracy: a criterion that guides data-driven combination of ambiguous outcome labels in multi-class classification. *J. Mach. Learn. Res.* **23**(341), 1–65 (2022)

# Efficient Minimizer Orders for Large Values of $k$ Using Minimum Decycling Sets

David Pellow<sup>1</sup>, Lianrong Pu<sup>1</sup>, Baris Ekim<sup>2</sup>, Lior Kotlar<sup>3</sup>, Ron Shamir<sup>1</sup>,  
and Yaron Orenstein<sup>4,5</sup>(✉)

<sup>1</sup> Blavatnik School of Computer Science, Tel-Aviv University, Tel Aviv-Yafo, Israel

<sup>2</sup> Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of  
Technology, Cambridge, MA, USA

<sup>3</sup> Department of Computer Science, Ben-Gurion University, Beersheba, Israel

<sup>4</sup> Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

<sup>5</sup> The Mina and Everard Goodman Faculty of Life Sciences, Bar-Ilan University,  
Ramat Gan, Israel

[yaron.orenstein@biu.ac.il](mailto:yaron.orenstein@biu.ac.il)

**Introduction.** Efficient methods to map, store, and search DNA sequences have become critical in the analysis of ever-growing sequencing data. Sequence sketching, which is necessary to reduce memory and runtime overhead, is a fundamental building block in many of these sequence analysis tasks. The common principle in all sketching techniques is the consistent selection of a  $k$ -mer representative from a long DNA sequence for indexing the sequences in data structures or algorithms. A key parameter for evaluating and comparing sketching schemes is their density, which is defined as the fraction of  $k$ -mers selected from a sequence by the scheme.

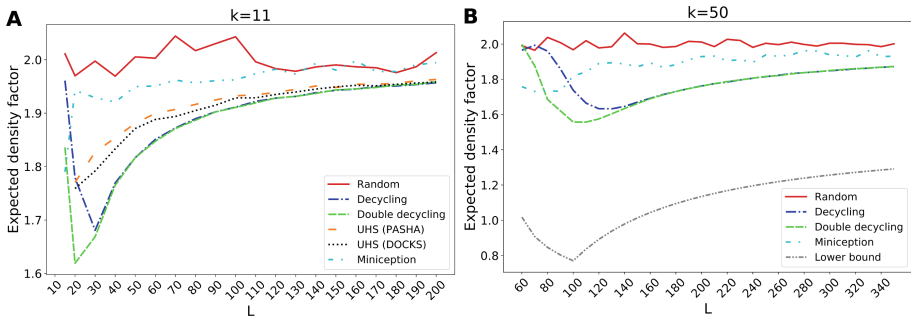
Minimizers are the most common sequence sketching techniques and have been widely applied across many different bioinformatics applications. In this technique, the minimum  $k$ -mers from every  $L$ -long window of a sequence are selected, based on some pre-defined order. However, the commonly used lexicographic and random  $k$ -mer orders have been shown to achieve density that is far from optimal. Minimizer orders based on universal hitting sets (UHS) have achieved lower density [1], but UHS construction becomes infeasible for  $k > 13$ , making UHS-based orders of limited use. In this work, we present minimizer orders that are compatible with a minimum decycling set (MDS) of a de Bruijn graph (DBG). These orders have density that is as low or even lower than UHS-compatible orders and can be computed efficiently for large values of  $k$ .

**Methods.** A complete DBG of order  $k$  is a directed graph in which each  $k$ -mer is represented by a node, and directed edges exist for every pair  $(u, v)$  such that the  $(k-1)$ -long suffix of  $u$  is identical to the  $(k-1)$  long prefix of  $v$ . Paths in the DBG represent all possible sequences, and a path of  $w$  nodes represents a sequence of  $w$  overlapping  $k$ -mers. An MDS is a minimum set of nodes necessary to remove all cycles from a graph, and finding it in a general graph is NP-hard. However, an MDS of a complete DBG can be constructed in linear time using an algorithm due to Mykkeltveit [2]. We introduce an algorithm to test the membership of a

$k$ -mer in this MDS in  $O(k)$  time based on Mykkeltveit’s construction. We define a decycling-set-based minimizer order compatible with an MDS such that every  $k$ -mer in the MDS precedes all other  $k$ -mers, and the order of two  $k$ -mers within or outside of the MDS is determined by a random hash function. Using our new membership algorithm and the new order we defined enables computing minimizers in a sequence on the fly for any value of  $k$ .

We further extended the decycling-set-based order by defining two symmetric MDSs, which are based on the fact that Mykkeltveit’s construction uses symmetric properties of the DBG. We prove that the union of these two sets leaves shorter remaining paths in the DBG compared to each of the symmetric sets separately. We thus define a minimizer order compatible with a *double* decycling set, which we hypothesize will achieve even lower density.

**Results.** We compared the density factors of our new minimizer orders to UHS-based orders, a Miniception-based order [3], and a random minimizer order (Fig. 1). The density factor is density normalized by length to a unitless value such that the random order has an expected density factor of 2. The results show that the decycling- and double decycling-set-based orders achieved the lowest expected density factor, and could be extended efficiently to large  $k$ . Moreover, the double-decycling-set-based order achieves lower density factors than the decycling-based order for smaller values of  $L = w + k - 1$ . Our full paper presents extensive results over a range of  $k$  and  $L$  and various real genome sequences.



**Fig. 1. Decycling-set-based minimizer orders achieved the lowest expected density.** The expected density factors of different minimizer orders are compared over a range of  $L$  values for  $k = 11$  (A) and  $k = 50$  (B). Average density factor are reported over ten runs with 10M nt random sequences.

**Conclusion.** Our new decycling-set-based orders are the first to yield much lower density than random that can be efficiently extended to larger  $k$  without being tailored to a specific sequence. By implementing our new decycling-set-based minimizer orders in data structures and algorithms of high-throughput



DNA sequencing analysis, we expect to see reductions in runtime and memory usage, beyond what was previously demonstrated using UHS-based minimizer orders.

Our manuscript is available on bioRxiv at <https://doi.org/10.1101/2022.10.18.512682> and the code is freely available from [github.com/OrensteinLab/DecyclingSetBasedMinimizerOrder](https://github.com/OrensteinLab/DecyclingSetBasedMinimizerOrder).

## References

1. Marçais, G., Pellow, D., Bork, D., Orenstein, Y., Shamir, R., Kingsford, C.: Improving the performance of minimizers and winnowing schemes. *Bioinformatics* **33**(14), i110–i117 (2017). <https://doi.org/10.1093/bioinformatics/btx235>
2. Mykkeltveit, J.: A proof of Golomb’s conjecture for the de Bruijn graph. *J. Comb. Theory Ser. B* **13**(1), 40–45 (1972). <http://www.sciencedirect.com/science/article/pii/0095895672900068>
3. Zheng, H., Kingsford, C., Marçais, G.: Improved design and analysis of practical minimizers. *Bioinformatics* **36**(Supplement\_1), i119–i127 (2020). <https://doi.org/10.1093/bioinformatics/btaa472>

# Dashing 2: Genomic Sketching with Multiplicities and Locality-Sensitive Hashing

Daniel N. Baker  and Ben Langmead <sup>(✉)</sup> 

Johns Hopkins University, Baltimore, MD 21218, USA  
{dnb, langmea}@cs.jhu.edu

**Abstract.** A genomic sketch is a small, probabilistic representation of the set of  $k$ -mers in a sequencing dataset. Sketches are building blocks for large-scale analyses that consider similarities between many pairs of sequences or sequence collections. While existing tools can easily compare 10,000s of genomes, relevant datasets can reach millions of sequences and beyond. Popular tools also fail to consider  $k$ -mer multiplicities, making them less applicable in quantitative settings. We describe a method called Dashing 2 that builds on the SetSketch data structure. SetSketch is related to HyperLogLog, but discards use of leading zero count in favor of a truncated logarithm of adjustable base. Unlike HLL, SetSketch can perform multiplicity-aware sketching when combined with the ProbMin-Hash method. Dashing 2 integrates locality-sensitive hashing to scale all-pairs comparisons to millions of sequences. Dashing 2 is free, open source software available at <https://github.com/dnbaker/dashing2>.

**Keywords:** Genomics · Sequencing · Sketching

## 1 Introduction

Sketching, e.g. based on MinHash or HyperLogLog, is a key building block for scaling sequence comparison. Sketches built over all the  $k$ -mers in a sequence have been applied in clustering [8], phylogenetic inference [2], strain-level profiling [3, 6], species delineation [5] and summarization of genomic collections [1, 7]. While existing tools can easily cluster 10,000s of genomes, many relevant biological datasets are much larger, reaching millions of sequences and beyond. Further, these tools fail to consider multiplicities of the  $k$ -mers, limiting their applicability in settings where quantities matter, e.g. when analyzing collections of sequence reads, or summaries from quantitative sequencing assays.

Dashing 2 builds on the recent SetSketch structure [4]. SetSketch is related to HyperLogLog (HLL), but replaces the HLL's leading zero count (LZC) operation with a truncated logarithm of adjustable base. This addresses a major disadvantage of the HLL as implemented in Dashing, since the LZC wastes about 2 bits of space out of every 8-bit estimator ("register") stored. SetSketch is also capable of multiplicity-aware sketching. Finally, SetSketch admits a simple, accurate algorithm for computing similarity between sketches in a joint fashion.

## 2 Methods Summary

Ertl's SetSketch [4] addresses this issue by replacing the HyperLogLog's LZC operation with a logarithm of configurable base  $b$ . Given a data item  $d$ , the update rule for each register  $K_i$ , is:

$$K_i = \max(K_i, \lfloor 1 - \log_b h_i(d) \rfloor) \quad (1)$$

Since each register is a function of every input item, each addition may require  $O(m)$  work where  $m$  is the number of registers. Building on Ertl's work, we propose (in the full paper) several practice improvements that allow this computational to be done much more quickly.

Further, this approach can compute a weighted version of the Jaccard similarity, known as the Probability Jaccard similarity.

Dashing 2 has other notable modes and features, discussed in the full paper.

## 3 Results Summary

Our results show that Dashing 2 is capable of sketching very large collections of genomics data very quickly. Importantly, it can also compute the similarity between two sketches much more quickly than the original version of Dashing. Its estimates for the Jaccard similarity and for the related Mash distance are generally better than other competing methods like Mash [8], Dashing [1] and BinDash [9]. Details can be found in the full paper.

## References

1. Baker, D.N., Langmead, B.: Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biol.* **20**(1), 265 (2019). <https://doi.org/10.1186/s13059-019-1875-0>
2. Criscuolo, A.: On the transformation of MinHash-based uncorrected distances into proper evolutionary distances for phylogenetic inference. *F1000Res* **9**, 1309 (2020)
3. Dilthey, A.T., Jain, C., Koren, S., Phillippy, A.M.: Strain-level metagenomic assignment and compositional estimation for long reads with MetaMaps. *Nat. Commun.* **10**(1), 3066 (2019)
4. Ertl, O.: SetSketch: filling the gap between MinHash and HyperLogLog. *Proc. VLDB Endow.* **14**(11), 2244–2257 (2021). <https://doi.org/10.14778/3476249.3476276>
5. Gostinčar, C.: Towards genomic criteria for delineating fungal species. *J. Fungi (Basel)* **6**(4) (2020)
6. LaPierre, N., Alser, M., Eskin, E., Koslicki, D., Mangul, S.: Metalign: efficient alignment-based metagenomic profiling via containment min hash. *Genome Biol.* **21**(1), 242 (2020). <https://doi.org/10.1186/s13059-020-02159-0>
7. Ondov, B.D., et al.: Mash screen: high-throughput sequence containment estimation for genome discovery. *Genome Biol.* **20**(1), 232 (2019). <https://doi.org/10.1186/s13059-019-1841-x>
8. Ondov, B.D., et al.: Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.* **17**(1), 132 (2016). <https://doi.org/10.1186/s13059-016-0997-x>
9. Zhao, X.: Bindash, software for fast genome distance estimation on a typical personal laptop. *Bioinformatics*, bty651 (2018)

# *Startle*: A Star Homoplasy Approach for CRISPR-Cas9 Lineage Tracing

Palash Sashittal<sup>1</sup>, Henri Schmidt<sup>1</sup>, Michelle Chan<sup>2</sup>,  
and Benjamin J. Raphael<sup>1</sup>(✉)

<sup>1</sup> Department of Computer Science, Princeton University, Princeton, NJ, USA  
[braphael@princeton.edu](mailto:braphael@princeton.edu)

<sup>2</sup> Department of Molecular Biology, Princeton University, Princeton, NJ, USA

**Abstract.** CRISPR-Cas9 based genome editing combined with single-cell sequencing enables the tracing of the history of cell divisions, or cellular lineage, in tissues and whole organisms. While standard phylogenetic approaches may be applied to reconstruct cellular lineage trees from this data, the unique features of the CRISPR-Cas9 editing process motivate the development of specialized models that describe the evolution of cells with CRISPR-Cas9 induced mutations. Here, we introduce the *star homoplasy* model, a novel evolutionary model that constrains a phylogenetic character to mutate at most once along a lineage, capturing the *non-modifiability* property of CRISPR-Cas9 mutations. We derive a combinatorial characterization of star homoplasy phylogenies by identifying a relationship between the star homoplasy model and the binary perfect phylogeny model. We use this characterization to develop an algorithm, *Startle* (Star tree lineage estimator), that computes a maximum parsimony star homoplasy phylogeny. We demonstrate that *Startle* infers more accurate phylogenies on simulated CRISPR-based lineage tracing data compared to existing methods; particularly on data with high amounts of dropout and homoplasy. *Startle* also infers more parsimonious phylogenies with fewer metastatic migrations on a lineage tracing dataset from mouse metastatic lung adenocarcinoma.

**Code availability:** Software is available at <https://github.com/raphael-group/startle>.

**Preprint availability:** Full preprint is available at <https://www.biorxiv.org/content/10.1101/2022.12.18.520935v1>.

---

P. Sashittal and H. Schmidt—These authors contributed equally to this work, author order was decided alphabetically.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023  
H. Tang (Ed.): RECOMB 2023, LNBI 13976, p. 229, 2023.  
<https://doi.org/10.1007/978-3-031-29119-7>



# A Fast and Scalable Method for Inferring Phylogenetic Networks from Trees by Aligning Lineage Taxon Strings (Extended Abstract)

Louxin Zhang<sup>1</sup>(✉), Niloufar Abhari<sup>2</sup>, Caroline Colijn<sup>2</sup>, and Yufeng Wu<sup>3</sup>

<sup>1</sup> Department of Mathematics and Centre for Data Science and Machine Learning,  
National University of Singapore, 119076 Singapore, Singapore

matzlx@nus.edu.sg

<sup>2</sup> Department of Mathematics, Simon Fraser University,  
Burnaby, B.C. V5A 1S6, Canada

{nabhari,ccolijn}@sfu.ca

<sup>3</sup> Department of Computer Science and Engineering, University of Connecticut,  
Storrs 06269, USA

yufeng.wu@uconn.edu

**Abstract.** A method is developed for inferring the minimum tree-child network from multiple trees by aligning the lineage taxon strings that are obtained from the trees for each taxon. This algorithmic innovation enables us to get around the scalable limitation of the existing programs for phylogenetic network inference.

**Keywords:** Phylogenetic tree · Phylogenetic network · Shortest common supersequence

Now that a variety of genomic projects have been completed, reticulate evolutionary events have been demonstrated to play important roles in genome evolution [3, 5]. In this study, phylogenetic networks are rooted, directed acyclic graphs in which the leaves are labeled with taxa, the non-leaf indegree-1 nodes represent speciation events and the nodes with multiple incoming edges represent reticulation events. Phylogenetic trees are phylogenetic networks with no reticulate nodes. Although phylogenetic networks are more appealing than trees for modeling reticulate events, it is challenging to apply phylogenetic networks in the study of genome evolution.

One approach to phylogenetic network inference is that phylogenetic trees are first inferred from biomolecular sequences and then used to reconstruct a phylogenetic network with the smallest hybridization number (HN) that displays all the trees (see [2]), where the HN is defined as the sum over all the reticulate nodes of the difference between the indegree and outdegree of each reticulate

---

Supported partially by Singapore MOE Tier 1 grant R-146-000-318-114 (LX Zhang) and U.S. National Science Foundation grants CCF-1718093 and IIS-1909425 (Y Wu). For correspondence: matzlx@nus.edu.sg.

node. This approach takes advantage of the fact that the theory of phylogenetic trees is mature and there are excellent tools available for inferring trees from a large number of sequences. But, inferring a phylogenetic network with the smallest HN from multiple trees is NP-hard even for the special case when there are only two input trees. None of the existing programs can be used for inferring a network from a set of 30 trees on 30 taxa in which the trees do not contain any non-trivial common clusters, i.e. no proper cluster appears in all trees.

Since the general inference problem is hard, attention has been switched to the inference of the tree-child networks, in which every non-leaf node has at least one child that is not reticulate, or, recently, a tree-based network [8]. Tree-child networks have a completeness property that for any set of binary trees, there is always a tree-child network (whose reticulate nodes can be of indegree 2 or more) that displays all the trees [6].

The program we introduce here, named ALTS, takes a different approach from other studies [1, 4, 7] for tree-child network inference. In each input tree, we first label the non-leaf nodes with taxa w.r.t. an ordering of the taxa and, for each taxon, compute its lineage taxon string (LTS) that consists of the labels of certain ancestors of that taxon. Then, we compute the shortest common supersequence (SCS) of the LTSs obtained for each taxon and use all the SCSs to construct a tree-child network. Here, a string is said to be a common supersequence of a set of strings if each of the latter can be obtained from the former by the removal of one or more letters. The smallest tree-child network displaying all trees can be found by examining all orderings of the taxa. This algorithmic innovation enables us to get around the scalable limitation associated with the parsimonious inference by efficiently sampling the taxon orderings and progressively computing the SCSs. We also added a feature of inferring a weighted tree-child network if the input trees are weighted.

The full version of this work can be found on arXiv (Id: 2301.00992).

## References

1. Albrecht, B., Scornavacca, C., Cenci, A., Huson, D.H.: Fast computation of minimum hybridization networks. *Bioinformatics* **28**(2), 191–197 (2012)
2. Elworth, R.A.L., Ogilvie, H.A., Zhu, J., Nakhleh, L.: Advances in computational methods for phylogenetic networks in the presence of hybridization. In: Warnow, T. (ed.) *Bioinformatics and Phylogenetics*. CB, vol. 29, pp. 317–360. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-10837-3\\_13](https://doi.org/10.1007/978-3-030-10837-3_13)
3. Fontaine, M.C., Pease, J.B., Steele, A., et al.: Extensive introgression in a malaria vector species complex revealed by phylogenomics. *Science* **347**(6217), 1258524 (2015). <https://doi.org/10.1126/science.1258524>
4. van Iersel, L., Janssen, R., Jones, M., Murakami, Y., Zeh, N.: A practical fixed-parameter algorithm for constructing tree-child networks from multiple binary trees. *Algorithmica* **84**(4), 917–960 (2022). <https://doi.org/10.1007/s00453-021-00914-8>
5. Koonin, E.V., Makarova, K.S., Aravind, L.: Horizontal gene transfer in prokaryotes: quantification and classification. *Annu. Rev. Microbiol.* **55**(1), 709–742 (2001)

6. Linz, S., Semple, C.: Attaching leaves and picking cherries to characterise the hybridisation number for a set of phylogenies. *Adv. Appl. Math.* **105**, 102–129 (2019)
7. Mirzaei, S., Wu, Y.: Fast construction of near parsimonious hybridization networks for multiple phylogenetic trees. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**(3), 565–570 (2015)
8. Pickrell, J., Pritchard, J.: Inference of population splits and mixtures from genome-wide allele frequency data. *Nat. Precedings* (2012). <https://doi.org/10.1038/npre.2012.6956.1>

# Aligning Distant Sequences to Graphs Using Long Seed Sketches

Amir Joudaki<sup>1,2</sup>, Alexandru Meterez<sup>1</sup>, Harun Mustafa<sup>1,2,3</sup>,  
Ragnar Groot Koerkamp<sup>1</sup>, André Kahles<sup>1,2,3</sup>(✉), and Gunnar Rätsch<sup>1,2,3,4</sup>(✉)

<sup>1</sup> Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland  
{amir.joudaki,alexandru.meterez,harun.mustafa,ragnargroot.koerkamp,  
andre.kahles,gunnar.ratsch}@inf.ethz.ch

<sup>2</sup> Biomedical Informatics Research, University Hospital Zurich, 8091 Zurich,  
Switzerland

<sup>3</sup> Swiss Institute of Bioinformatics, 1015 Lausanne, Switzerland

<sup>4</sup> ETH AI Center, 8092 Zurich, Switzerland

*Sequence-to-Graph Alignment.* Sequence-to-graph alignment is a key step in various bioinformatics applications, such as variant genotyping [1] and genome assembly [2]. The goal of sequence-to-graph alignment is to determine the minimum number of editing operations needed to transform a query sequence into a reference sequence represented in a graph. Seed-and-extend is an approximate sequence-to-graph alignment approach [3], that typically uses  $k$ -mer matches, i.e. substrings of length  $k$  of the original sequence, to find candidate locations in the graph to start the alignment from. However, using a large  $k$  lowers the recall, while using shorter  $k$  in large graphs with high levels of variation can lead to an exponential increase in false positive matches [4].

*Limitations of Exact Seeds.* To highlight these limitations, consider the following setting: we draw reference sequence Ref uniformly from the set of DNA nucleotides, and create a query sequence by copying a substring of the reference sequence at some offset position and substituting each nucleotide with probability  $r$ . To determine the offset between the reference and query using  $k$ -mer matches, at least one  $k$ -mer in the query must be copied without mutations. The probability of this occurring is  $(|q| - k + 1)(1 - r)^k$ , and the expected number of matches between reference and query  $k$ -mers due to chance is  $\mathcal{O}(|\text{Ref}||q|4^{-k})$ . This shows that while lowering  $k$  can improve recall, it also leads to more matches and more time needed to process them. To address this issue, a novel seeding approach that relies on long inexact matches instead of short exact matches is proposed.

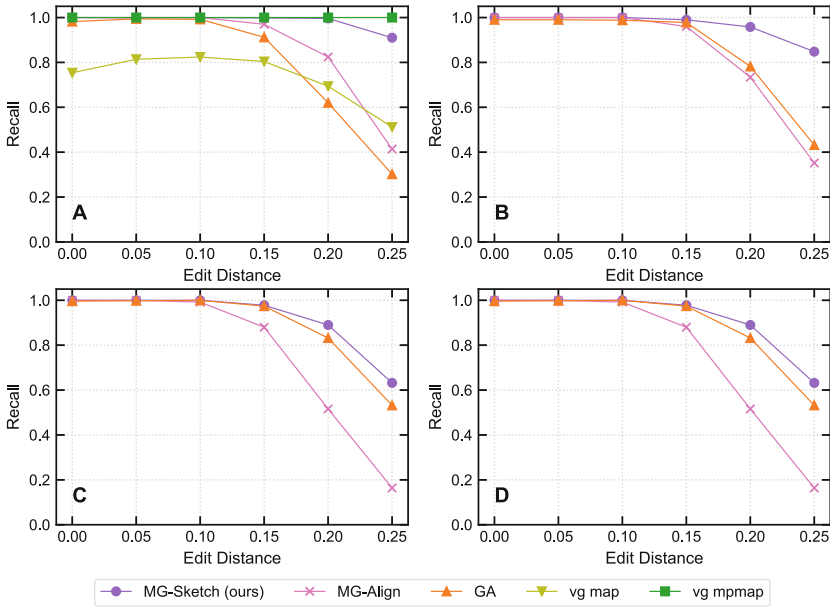
*Finding Inexact Seed Matches.* For finding seeds we use TENSOR SKETCHING [5], which in contrast to  $k$ -mers, relies on subsequences that are not necessarily contiguous. TENSOR EMBEDDING of a sequence  $a$  is a  $|\Sigma|^t$ -dimensional tensor that counts the number of subsequences of length  $t$  in  $a$ , and the embedding distance between  $a$  and  $b$  is defined as the Euclidean distance between their respective tensors. TENSOR SKETCHING is a Euclidean norm-preserving dimensionality

---

A. Joudaki and A. Meterez—Equal contribution.

reduction for TENSOR EMBEDDING, but does so in linear time and memory complexity. We can prove that under uniformly random sequences and substitution, the expected TENSOR SKETCHING distance is proportional to the edit distance. This enables us to recover sequence similarity even at high edit distance. Finally, we used a  $K$ -nearest neighbor index to anchor seeds from a query sequence to nodes in a graph. The index maps vectors in a vector space to lists of graph nodes and is used to anchor seeds in the query to the nearest neighbors returned by it. To improve scalability and avoid indexing redundant information, a subset of the nodes are sketched and stored in a *Hierarchical Navigable Small Worlds* (HNSW) index [6].

*Experimental Results.* We evaluate the performance of our TENSOR SKETCHING approach on synthetic and real genomic datasets and compare it to state-of-the-art seed-and-extend methods. Our results, presented in Fig. 1, show that our approach significantly reduces the number of false positive matches and produces more accurate alignments, particularly at high edit distance.



**Fig. 1.** The recall of different baselines was evaluated across different mutation rates and increasing graph sizes. The number of nodes in the graph for each plot is 100K (A), 10M (B), 100M (C) and 1B (D). The values were measured on a De Bruijn graph generated by METAGRAPH and MG-SKETCH was run with  $K = 40$  neighbors and  $D = 14, w = 16, s = 8, t = 6$ .

*Summary of Contributions.* We propose a novel seeding approach for sequence-to-graph alignment that relies on long inexact matches and a K-nearest neighbor index. Our approach significantly improves the accuracy and speed of alignment at high edit distance, and has the potential to significantly impact various bioinformatics applications that rely on sequence-to-graph alignment.

**Acknowledgement.** A. J. is funded through Swiss National Science Foundation Project Grant #200550 to A. K. H. M. is funded as part of Swiss National Research Programme (NRP) 75 “Big Data” by the SNSF grant #407540\_167331. A. J., H. M., and A. K. are also partially funded by ETH core funding (to G. R.). R.G. was funded through an ETH Research Grant (# ETH-17 21-1 ) to G.R.

We declare no conflicts of interest.

## References

1. Garrison, E., et al.: Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat. Biotechnol.* **36**(9), 875–879 (2018)
2. Rautiainen, M., Marschall, T.: GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biol.* **21**(1), 1–28 (2020)
3. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. *J. Mol. Biol.* **215**(3), 403–410 (1990)
4. Jain, C., Zhang, H., Gao, Y., Aluru, S.: On the complexity of sequence-to-graph alignment. *J. Comput. Biol.* **27**(4), 640–654 (2020)
5. Joudaki, A., Rättsch, G., Kahles, A.: Fast alignment-free similarity estimation by tensor sketching. *bioRxiv*, pp. 2020–11 (2021)
6. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **42**(4), 824–836 (2018)

# MD-Cat: Phylogenetic Dating Under a Flexible Categorical Model Using Expectation-Maximization

Uyen Mai<sup>1</sup>, Eduardo Charvel<sup>2</sup>, and Siavash Mirarab<sup>3</sup>(✉)

<sup>1</sup> Department of Computer Science and Engineering,  
UC San Diego, La Jolla, CA, USA

<sup>2</sup> Bioinformatics and Systems Biology Graduate Program, UC San Diego, La Jolla,  
CA, USA

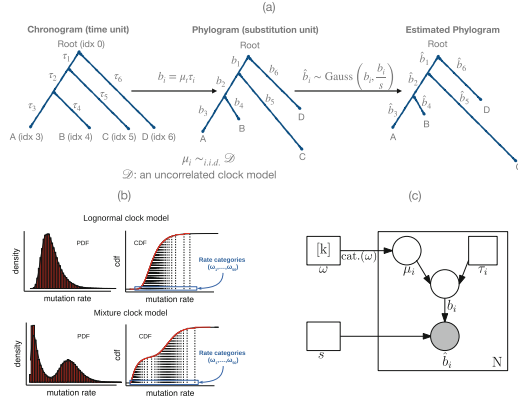
<sup>3</sup> Department of Electrical and Computer Engineering,  
UC San Diego, La Jolla, CA, USA  
smirarab@ucsd.edu

**Keywords:** Categorical model · EM algorithm · Molecular dating ·  
Phylogenetic dating · Time tree

## Extended Abstract

Substitution rates change through time, creating a major challenge for dating phylogenetic trees. Ideally, we need to know a *molecular clock* model that assigns probabilities to substitution rates throughout the tree. The strict clock model, while being simple and convenient to use, is insufficient to model phylogenies at long evolutionary time horizons. Alternatives to the strict clock exist, but none is universally accepted. While Maximum Likelihood (ML) methods require a parametric model of the molecular clock, Bayesian methods need a proper prior. Therefore, both ML and Bayesian methods are sensitive to model misspecification. We introduce a new method - Molecular Dating with Categorical rates (MD-Cat) - that uses a categorical distribution (CAT model) to approximate the distribution of the rates (Fig. 1). Our method is non-parametric, as we do not assume any predefined parametric model of the clock but use a CAT model to approximate it. We co-estimate the rate categories and branch lengths in time units by formulating a ML problem and solving it using Expectation-Maximization (EM) algorithm. We test MD-Cat on simulated and real datasets of Angiosperms and HIV. Under a wide selection of rate distributions, we show that MD-Cat is more accurate than the alternatives.

**The Model.** We assume the input tree has the correct topology and branch lengths are given in substitution unit with uncertainties. Let  $B_i$  be a random variable denoting the estimated length of branch  $e_i$  in substitution unit,  $\hat{b}_i$  be its estimate,  $\tau_i$  be the unknown length in time unit, and  $s$  be the sequence length (Fig. 1a). We assume  $B_i \sim_{i.i.d.} N(\mu_i \tau_i, \frac{\hat{b}_i}{s})$  and let  $f(\hat{b}_i | \mu_i, \tau_i)$  denote the Gaussian pdf of this model. To model the molecular clock, we discretize the rates



**Fig. 1.** (a) The generative model. (b) Examples of clock models and their categorical approximation. Rate categories are identified by the projection of the steps onto the x-axis of the CDF plot. (c) The hierarchical model in plate notation.

into  $k$  categories  $\omega = [\omega_1, \omega_2, \dots, \omega_k]$  each with the same probability mass  $\frac{1}{k}$  and use this CAT model to approximate the unknown distribution of the rates.

**The Log-Likelihood Function.** Putting all elements together, we obtain a hierarchical probabilistic model (Fig. 1c). The log-likelihood function is

$$l(\tau, \omega) = \sum_{i=1}^N \log L_i(\hat{b}_i; \tau_i, \omega) = \sum_{i=1}^N \log \left( \frac{1}{k} \sum_{j=1}^k f(\hat{b}_i; \omega_j, \tau_i) \right) \quad (1)$$

where  $L_i(\hat{b}_i; \tau_i, \omega)$  denotes the density of  $\hat{b}_i$  on branch  $i$  and  $f$  is the density function of the Gaussian model. Our goal is to find  $\tau$  and  $\omega$  that maximize  $l(\tau, \omega)$  and satisfy the linear constraints defined by calibration points.

**The EM Algorithm.** We start with an initial of  $\tau$  and  $\omega$  and iteratively improve the log-likelihood function by alternating between the E-step and M-step. In the **E-step**, we compute the posterior of the latent variables:

$$q_{ij} = Pr(\mu_i = \omega_j | \hat{b}_i; \tau, \omega) = \frac{f(\hat{b}_i | \mu_i = \omega_j, \tau_i)}{\sum_{m=1}^k f(\hat{b}_i | \mu_i = \omega_m, \tau_i)}. \quad (2)$$

In the **M-step**, we find  $\omega$  and  $\tau$  to maximize

$$\sum_{i=1}^N \sum_{j=1}^k q_{ij} \log f(\hat{b}_i | \omega_j, \tau_i) \quad (3)$$

Using the Gaussian pdf and removing constants, the problem is reduced to:

$$\min_{\tau > 0, \omega > 0} \sum_{i=1}^N \sum_{j=1}^k \frac{sq_{ij}}{\hat{b}_i} (\hat{b}_i - \omega_j \tau_i)^2 \quad (4)$$

such that the calibration constraints are satisfied.



**Open Source Software.** <https://github.com/uym2/MD-Cat>.

**Data Availability.** <https://github.com/uym2/MD-Cat-paper>.

**Preprint.** <https://www.biorxiv.org/content/10.1101/2022.10.06.511147v1>.

**Acknowledgement.** This work was supported by the National Institutes of Health (1R35GM142725).

# Phenotypic Subtyping via Contrastive Learning

Aditya Gorla<sup>1</sup>, Sriram Sankararaman<sup>2,3,4</sup>, Esteban Burchard<sup>5,6</sup>,  
Jonathan Flint<sup>7</sup>, Noah Zaitlen<sup>3,4,8</sup>, and Elior Rahmani<sup>3(✉)</sup>

<sup>1</sup> Bioinformatics Interdepartmental Program, University of California, Los Angeles,  
Los Angeles, CA, USA

<sup>2</sup> Department of Computer Science, University of California, Los Angeles, Los  
Angeles, CA, USA

<sup>3</sup> Department of Computational Medicine, David Geffen School of Medicine,  
University of California, Los Angeles, Los Angeles, CA, USA

<sup>4</sup> Department of Human Genetics, University of California, Los Angeles, Los  
Angeles, CA, USA

<sup>5</sup> Department of Medicine, University of California, San Francisco, San Francisco,  
CA, USA

<sup>6</sup> Department of Bioengineering and Therapeutic Sciences, University of California  
San Francisco, San Francisco, CA, USA

<sup>7</sup> Department of Psychiatry and Behavioral Sciences, Brain Research Institute,  
University of California, Los Angeles, Los Angeles, CA, USA

<sup>8</sup> Department of Neurology, David Geffen School of Medicine, University of  
California, Los Angeles, Los Angeles, CA, USA

[elior.rahmani@gmail.com](mailto:elior.rahmani@gmail.com)

**Abstract.** Defining and accounting for subphenotypic structure has the potential to increase statistical power and provide a deeper understanding of the heterogeneity in the molecular basis of complex disease. Existing phenotype subtyping methods primarily rely on clinically observed heterogeneity or metadata clustering. However, they generally tend to capture the dominant sources of variation in the data, which often originate from variation that is not descriptive of the mechanistic heterogeneity of the phenotype of interest; in fact, such dominant sources of variation, such as population structure or technical variation, are, in general, expected to be independent of subphenotypic structure. We instead aim to find a subspace with signal that is unique to a group of samples for which we believe that subphenotypic variation exists (e.g., cases of a disease). To that end, we introduce Phenotype Aware Components Analysis (PACA), a contrastive learning approach leveraging canonical correlation analysis to robustly capture weak sources of subphenotypic variation. In the context of disease, PACA learns a gradient of variation unique to cases in a given dataset, while leveraging control samples for accounting for variation and imbalances of biological and technical confounders between cases and controls. We evaluated PACA using an extensive simulation study, as well as on various subtyping tasks using genotypes, transcriptomics, and DNA methylation data. Our results provide multiple strong evidence that PACA allows us to robustly capture weak unknown variation of interest while being calibrated and well-powered, far superseding the performance of alternative methods. This renders

PACA as a state-of-the-art tool for defining *de novo* subtypes that are more likely to reflect molecular heterogeneity, especially in challenging cases where the phenotypic heterogeneity may be masked by a myriad of strong unrelated effects in the data.

A preprint of the full paper is available at:

<https://www.biorxiv.org/content/10.1101/2023.01.05.522921v1>.

# HOGVAX: Exploiting Peptide Overlaps to Maximize Population Coverage in Vaccine Design with Application to SARS-CoV-2

Sara C. Schulte<sup>1,2</sup> , Alexander T. Dilthey<sup>2</sup> , and Gunnar W. Klau<sup>1</sup> 

<sup>1</sup> Algorithmic Bioinformatics, Heinrich Heine University Düsseldorf, Düsseldorf, Germany  
gunnar.klau@hhu.de

<sup>2</sup> Institute of Medical Microbiology and Hospital Hygiene, University Clinic Düsseldorf, Düsseldorf, Germany

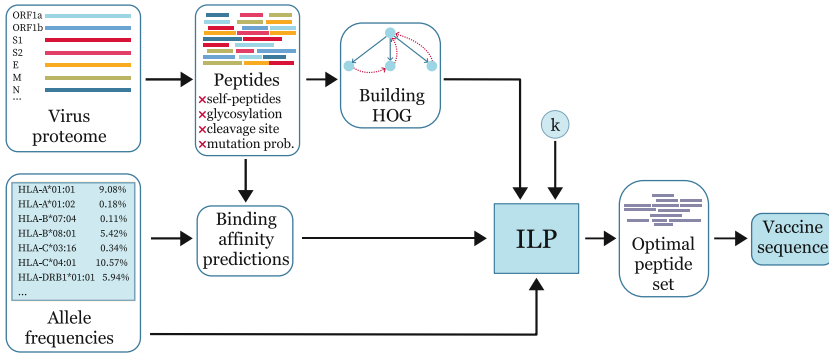
Vaccination provides an effective way to develop immunity against many diseases and is the key component to overcoming the global COVID-19 pandemic. Peptide vaccines present a safe and cost-efficient alternative to traditional vaccines with fewer side effects and with application to non-infectious diseases like cancer or Alzheimer's disease. They are rapidly produced and adapted for new viral variants. Their efficacy depends on the peptides included in the vaccine and the ability of major histocompatibility complex (MHC) molecules to bind and present these peptides to cells of the immune system. Due to the high diversity of MHC alleles, their diverging peptide binding specificities, and physical constraints on the maximum length of peptide vaccine constructs, choosing a set of peptides that effectively achieve immunization across a large proportion of the population is challenging.

Bioinformatics has long been well established in peptide vaccine design to optimize population coverage or the number of peptides presented by the human leukocyte antigen (HLA) molecules per individual of a target population. Usually, selected peptides are concatenated with optional spacer sequences in between. Due to the restricted vaccine capacities such approaches have difficulty covering the full population or being robust to viral mutations. Especially rare MHC alleles are often not targeted by the vaccine formulation.

Here, we present HOGVAX, a combinatorial optimization approach to select peptides that maximize population coverage (Fig. 1). Our model requires (i) candidate peptide sequences, (ii) HLA allele frequencies, and (iii) binding affinity predictions between peptides and HLA alleles. The key idea behind HOGVAX is to exploit overlaps between peptide sequences to include a large number of peptides in limited space and thereby also cover rare MHC alleles. We model this task as a theoretical problem, which we call the *Maximum Scoring k-Superstring Problem* (MSKS). We show that MSKS is NP-hard and introduce a reformulation as a graph problem using the hierarchical overlap graph (HOG) data structure. This rooted directed tree-like structure contains the input peptides and all pairwise maximal overlaps as nodes. Weighted tree edges model prefixes

---

A. T. Dilthey, G. W. Klau—Shared last author.



**Fig. 1.** HOGVAX workflow.

while special edges called suffix links connect each node with its longest proper suffix in the HOG. We give an integer linear programming (ILP) formulation for the graph that selects a closed walk equivalent to a vaccine formulation of length  $k$ . Each MHC allele covered by peptides included in the walk increases the population coverage. Furthermore, HOGVAX is able to consider haplotype frequencies to take linkage disequilibrium between MHC loci into account.

We compare HOGVAX to the state-of-the-art tool OptiVax by Liu et al. [Cell Systems **11**(2), 2020] using the same SARS-CoV-2 case study proposed by Liu et al. Candidate peptides derive from using sliding windows over the entire SARS-CoV-2 proteome. HLA allele frequencies are available from databases, and allele linkage can be considered by combining the frequencies of allele combinations with haplotype or genotype data of typed individuals. Binding affinities result from in silico prediction methods as they are difficult to obtain experimentally and are represented as a binary matrix expressing whether a specific candidate peptide binds to an allele (combination).

We first run OptiVax to select a number of peptides which we concatenate to define the maximum vaccine length  $k$ . We then run HOGVAX using this  $k$ . HOGVAX creates vaccines with maximal population coverage based on single allele data in under 15 min and based on haplotype frequencies in about 1.5 h. In addition, our vaccine formulations contain significantly more peptides compared to vaccine sequences built from concatenated peptides, which has been the state-of-the-art. We predict over 98% population coverage for our vaccine candidates of MHC class I and II based on single-allele and haplotype frequencies. Moreover, we predict high numbers of per-individual presented peptides leading to a robust immunity in the face of new virus variants.

Note that we are constructing an overlap-vaccine sequence that must be degraded by the proteasome into presentable peptide fragments to induce immunity. The proteasomal cleavage process is, however, not fully understood yet. Thus, only experimental investigations will show how well our approach and competing methods will work in practice.

We provide a publicly available open source implementation of HOGVAX and all datasets required for reproduction at <https://gitlab.cs.uni-duesseldorf.de/schulte/hogvax> and a full version of the paper at <https://doi.org/10.1101/2023.01.09.523288>.



# Ultra-Fast Genome-Wide Inference of Pairwise Coalescence Times

Regev Schweiger<sup>(✉)</sup> and Richard Durbin

Department of Genetics, University of Cambridge, Cambridge, UK  
{rs2145,rd109}@cam.ac.uk

**Keywords:** Sequentially Markovian coalescent · Coalescent theory · Population genetics · Selection

Each of an individual's two alleles, at each genomic position, is inherited from a chain of ancestors going back in time. These two chains must eventually coalesce at an identical common ancestor. The length of these chains, measured by the number of generations until coalescence, is called the *coalescence time*, or the *time to the most recent common ancestor* (TMRCA). Coalescence times along two homologous chromosomes can be well approximated with a Markov chain using the sequentially Markovian coalescent (SMC) framework [1, 2]. This framework underlies the PSMC [3] method and its extensions [3–6], which estimate a posterior distribution of the TMRCA at each genomic position.

With increasing dataset sizes, scalable inference of coalescence times between each pair of haplotypes in a large dataset is of great interest, as they may provide otherwise unattainable rich information about the population structure and history of the sample.

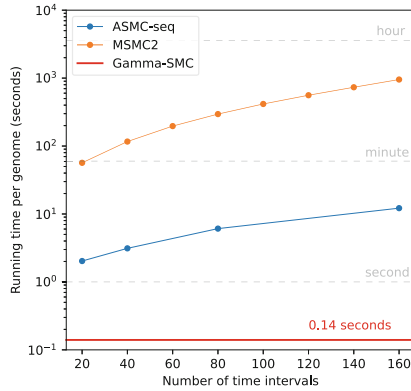
In SMC-based methods, hidden states along the genome correspond to pairwise TMRCA. However, standard HMMs require a discrete hidden state space, while coalescence time is naturally continuous. Therefore, current SMC-based methods discretize time into  $T$  intervals, and so have at least  $O(T)$  computational complexity per step.

We introduce *Gamma-SMC*, which avoids discretizing time by utilizing the observation that SMC posterior coalescence time distributions are very well approximated by a gamma distribution [7]. If we constrain the posterior to be a member of the family  $\Gamma(\alpha, \beta)$ , then the hidden state is described by only two parameters ( $\alpha$  and  $\beta$ ) per step, which leads to  $O(1)$  time-complexity per step.

To support HMM operations, we describe how to update the forward posterior density from one step to the next. For the emission step, we exploit the conjugacy between Poisson and gamma to derive a simple update rule for the  $\alpha, \beta$  parameters. For the transition step, we derived a closed-form expression of the coalescence time at step  $i + 1$ , given that the forward density at step  $i$  is gamma, which is approximately also gamma. For each possible  $\Gamma(\alpha, \beta)$  distribution, we thus map a new approximate subsequent  $\Gamma(\alpha', \beta')$  distribution. We

---

R. Durbin—Contributing author.



**Fig. 1. Speed.** Running times of Gamma-SMC vs. MSMC2 and ASMC-seq.

evaluate this mapping over a log-spaced grid, and use lookup and bilinear interpolation for  $O(1)$  evaluation of off-grid values. Importantly, while calculating this flow field is computationally expensive, it needs only be calculated once.

To evaluate the accuracy of Gamma-SMC, we simulated a whole genome using realistic population genetic parameters, used ASMC-seq [6], MSMC2 [8] and Gamma-SMC to infer the posterior mean of the coalescence times, and compared them to the true TMRCA. We observed that all methods have similar inference accuracy ( $r^2$ : ASMC-seq = 0.85, MSMC2 = 0.82, Gamma-SMC = 0.80), with Gamma-SMC slightly less accurate.

We next evaluated the running time of Gamma-SMC, observing that it is at least  $\times 14$  faster than ASMC-seq (Fig. 1), depending on  $T$ . For example, analysis of all pairs using  $T = 69$  discrete time intervals (ASMC-seq default), required 29 min for a single processor for ASMC-seq, compared to  $\sim 1$  min for Gamma-SMC. Extrapolated to a genome size of 3.2 Gbp, Gamma-SMC can process a whole genome in  $\sim 0.14$  s.

Finally, we applied Gamma-SMC to 505,515 pairs of haploid genomes from the 1000 Genomes Project data, and detected evidence for recent positive selection at multiple sites across the genome.

Link to the bioRxiv preprint: <https://www.biorxiv.org/content/10.1101/2023.01.06.522935v1>.

## References

1. McVean, G.A.T., Cardin, N.J.: Approximating the coalescent with recombination. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **360**(1459), 1387–1393 (2005)
2. Marjoram, P., Wall, J.D.: Fast “coalescent” simulation. *BMC Genet.* **7**(1), 16 (2006)
3. Li, H., Durbin, R.: Inference of human population history from individual whole-genome sequences. *Nature* **475**(7357), 493–496 (2011)



4. Schiffels, S., Wang, K.: MSMC and MSMC2: the multiple sequentially Markovian coalescent. In: Dutheil, J.Y. (ed.) *Statistical Population Genomics*. MMB, vol. 2090, pp. 147–166. Springer, New York (2020). [https://doi.org/10.1007/978-1-0716-0199-0\\_7](https://doi.org/10.1007/978-1-0716-0199-0_7)
5. Terhorst, J., Kamm, J.A., Song, Y.S.: Robust and scalable inference of population history from hundreds of unphased whole genomes. *Nat. Genet.* **49**(2), 303–309 (2017)
6. Palamara, P.F., Terhorst, J., Song, Y.S., Price, A.L.: High-throughput inference of pairwise coalescence times identifies signals of selection and enriched disease heritability. *Nat. Genet.* **50**(9), 1311–1317 (2018)
7. Albers, P.K., McVean, G.: Dating genomic variants and shared ancestry in population-scale sequencing data. *PLoS Biol.* **18**(1), 3000586 (2020)
8. Wang, K., Mathieson, I., O’Connell, J., Schiffels, S.: Tracking human population structure through time from whole genome sequences. *PLoS Genet.* **16**(3), 1008552 (2020)

# Leveraging Family Data to Design Mendelian Randomization That is Provably Robust to Population Stratification

Nathan LaPierre<sup>1</sup>(✉), Boyang Fu<sup>1</sup>, Steven Turnbull<sup>2</sup>, Eleazar Eskin<sup>1,3,4</sup>, and Sriram Sankararaman<sup>1,3,4</sup>(✉)

<sup>1</sup> Department of Computer Science, UCLA, Los Angeles, CA, USA  
nathanl2012@gmail.com

<sup>2</sup> Department of Statistics, UCLA, Los Angeles, CA, USA

<sup>3</sup> Department of Computational Medicine, UCLA, Los Angeles, CA, USA

<sup>4</sup> Department of Human Genetics, UCLA, Los Angeles, CA, USA  
sriram@cs.ucla.edu

Mendelian Randomization (MR) [1] is a widely-used analytical tool that uses genetic variants (“genetic instruments”) to determine whether one trait (the “exposure”) has a causal effect on another (the “outcome”). The validity of MR rests on three key assumptions [1]: (i) that the genetic instrument affects the exposure; (ii) that the genetic instrument is independent of confounders of the exposure-outcome relationship; (iii) that the genetic instrument affects the outcome only through its effect on the exposure. Unfortunately, the latter two assumptions are often violated in practice, due to several factors including horizontal pleiotropy and population stratification (and related phenomena such as assortative mating and dynastic effects). These biases can lead to bias and false positive findings in MR studies [1].

In this abstract, we introduce MR-Twin, a test for causal effects between pairs of traits that is able to leverage family-based genetic data to provably control for population stratification, cross-trait assortative mating, and dynastic effects, and avoids false positives due to weak instrument bias. MR-Twin tests for a causal effect by comparing an appropriate statistic computed on the offspring in the families to their “digital twins” [2], which are simulated offspring created by sampling offspring genotypes from parental genotypes. By testing for causal effects while conditioning on parental genotypes, MR-Twin avoids confounding caused by population stratification, since offspring genotypes are independent of population information given the genotypes of their parents.

We now outline the algorithm in the context of a trio design in which we have genetic data on the parents and the offspring. Let  $\mathbf{X}$  and  $O$  denote the genotypes and outcome phenotype values respectively for some individual, and let  $(\mathbf{X}_n; O_n)_{n=1}^N$  denote these across  $N$  trios. Also let  $\mathbf{P1}$  and  $\mathbf{P2}$  denote the genotypes of the parents of the individual with genotypes  $\mathbf{X}$ , and let  $\mathbf{A} := (\mathbf{P1}, \mathbf{P2})$  refer to the set of parental genotypes. Let  $\mathbf{Z}$  denote the set of external confounders measured on the same individual, which we define as the set of confounders that satisfy  $\mathbf{X} \perp\!\!\!\perp \mathbf{Z} \mid \mathbf{A}$ . Thus, population stratification is an exter-

---

N. LaPierre and B. Fu—These authors contributed equally to this work.

nal confounder (as are assortative mating and dynastic effects) while horizontal pleiotropy is not. The key idea is that we can formulate a hypothesis test of a causal effect conditional on the parental haplotypes  $\mathbf{A}$ . Bates et al. [2] show that such a test is also a test of the stronger null hypothesis of a causal effect conditional on  $(\mathbf{A}, \mathbf{Z})$ .

The way that this is accomplished is through a conditional randomization test, similar to the Digital Twin Test proposed by Bates et al. in the context of GWAS [2]. The idea is to sample so-called “digital twins”  $\tilde{\mathbf{X}}$  from each set of parents  $\mathbf{A}$  such that  $\tilde{\mathbf{X}} \mid \mathbf{A}$  has the same distribution as  $\mathbf{X} \mid \mathbf{A}$ , which can easily be accomplished using the laws of Mendelian inheritance. We construct  $B$  such random samples across all trios,  $(\tilde{\mathbf{X}}_n, O_n)_{n=1}^N$ , and for each set  $b$  of twins we compute a test statistic  $t_b = t((\tilde{\mathbf{X}}_n; O_n)_{n=1}^N; \hat{\beta})$  representing the strength of association between the genetically-predicted exposure and the outcome. We also construct a test statistic for the true offspring of the trios,  $t^* = t((\mathbf{X}_n; O_n)_{n=1}^N; \hat{\beta})$ . We can then obtain a p-value for a non-zero causal effect of the exposure on the outcome,  $p = \frac{1 + \mathbf{1}\{t_b \geq t^*\}}{1 + B}$ .

In order to assess the performance of MR-Twin, we simulated genotypes from two populations with allele frequency differences modeled according to the standard Balding-Nichols model [3], and then simulated phenotypes with confounding due to population stratification. We show that while standard MR methods had inflated false positive rates (FPRs) under this stratification model, MR-Twin did not have an inflated FPR regardless of the strength of confounding. We also demonstrate that a recent family-based method [1] controls FPR under stratification but, unlike MR-Twin, can still yield false positives due to weak instrument bias. We show that, given the same sample size, the methods have similar power. We then applied the methods to 121 pairs of traits measured in 959 White British trios from the UK Biobank [4]. MR-Twin identified trait pairs that are expected to be causal (such as Weight  $\rightarrow$  BMI) and did not identify trait pairs that are not expected to be causal (such as Height  $\rightarrow$  Body Fat).

MR-Twin should be a useful tool for researchers who want to verify that their significant MR findings are not confounded by population or familial effects. MR-Twin is freely available at: <https://github.com/nlapier2/MR-Twin>.

## References

1. Brumpton, B., et al.: Avoiding dynastic, assortative mating, and population stratification biases in mendelian randomization through within-family analyses. *Nat. Commun.* **11**(1), 1–13 (2020)
2. Bates, S., Sesia, M., Sabatti, C., Candès, E.: Causal inference in genetic trio studies. *Proc. Nat. Acad. Sci.* **117**(39), 24117–24126 (2020)
3. Balding, D.J., Nichols, R.A.: A method for quantifying differentiation between populations at multi-allelic loci and its implications for investigating identity and paternity. *Genetica* **96**(1), 3–12 (1995). <https://doi.org/10.1007/BF01441146>
4. Bycroft, C., et al.: The UK biobank resource with deep phenotyping and genomic data. *Nature* **562**(7726), 203–209 (2018)

# Minimal Positional Substring Cover: A Haplotype Threading Alternative to Li & Stephens Model

Ahsan Sanaullah<sup>1</sup>, Degui Zhi<sup>2</sup>, and Shaojie Zhang<sup>1</sup>(✉)

<sup>1</sup> Department of Computer Science, University of Central Florida, Orlando, FL  
32816, USA

asanauallah2019@knights.ucf.edu shzhang@cs.ucf.edu

<sup>2</sup> Center for AI and Genome Informatics, School of Biomedical Informatics,  
University of Texas Health Science Center at Houston, Houston, TX 77030, USA  
degui.zhi@uth.tmc.edu

The Li & Stephens (LS) hidden Markov model (HMM) [1] models the process of reconstructing a haplotype as a mosaic copy of haplotypes in a reference panel (haplotype threading). For small panels the probabilistic parameterization of LS enables modeling the uncertainties of such mosaics, and has been the foundational model for haplotype phasing and imputation. However, LS becomes inefficient when sample size is large (tens of thousands to millions), because of its linear time complexity ( $O(MN)$ , where  $M$  is the number of haplotypes and  $N$  is the number of sites in the panel).

Recent work has attempted to obtain the optimal haplotype threading in the Li & Stephens model in time sublinear to the size of the panel. Gerton Lunter described “fastLS”, an algorithm that implements the Li & Stephens model and obtains the optimal haplotype threading through the use of the Burrows-Wheeler transform [2]. His algorithm obtains the optimal haplotype threading orders of magnitude faster than the Viterbi algorithm. Yohei Rosen and Benedict Paten also described an algorithm that achieved runtime orders of magnitude faster than the Viterbi algorithm [3]. Their method achieves this using the efficient sparse representation of haplotypes and the lazy evaluation of dynamic programming. These algorithms have also been claimed to have runtime sublinear to the size of the reference panel. However, this claim has only been shown empirically.

Previously, we introduced a new formulation of haplotype threading, the Minimal Positional Substring Cover (MPSC) [4]. The MPSC problem is, given a query haplotype  $z$  and a set of haplotypes  $X$ , find a smallest set of segments of haplotypes in  $X$  that covers  $z$ . This formulation corresponds to LS model with 0 mismatch rate and the only objective is to minimize the switch rate. Obviously, the MPSC formulation is too limited as it does not tolerate mismatches. In a sense, MPSC is a more general formulation of haplotype threading. The traditional haplotype threading is a special case of MPSC with non-overlapping segments. MPSC formulation captures the fact that while the switching of templates indicates some recombination events, the exact breakpoint of the recombination events may be anywhere within the overlapping region between the templates. Furthermore, this combinatorial formulation is theoretically attractive because

**Table 1.** Summary of algorithms on haplotype threading

	Functionalities	Time complexity
Lunter [2]	fastLS	Expt. $o(MN)$
Rosen and Paten [3]	LS Lazy Evaluation	Expt. $o(MN)$
Sanaullah et al. [4]	MPSC	$O(N)$
	Set Maximal MPSC	$O(N)$
	$h$ -MPSC	$O(h \mathcal{C}  + N)$
<b>This work</b>	MPSC Graph	$O(N)$
	Length Maximal MPSC	$O(N)$
	Set Max. MPSC Solution Space Count	$O(N)$
	Set Max. MPSC Solution Space Enumerate	$O(N + S_c)$
	$h$ -MPSC	$O(N)$

$M$  is the number of sequences.  $N$  is the number of sites.  $\mathcal{C}$  is the outputted cover.  $h$  is the coverage guaranteed.  $S_c$  is the number of MPSCs outputted.

it is can be solved in worst case  $O(N)$  time [4]. I.E. given a PBWT of the reference panel, a MPSC haplotype threading of a query haplotype can be done in time independent to the number of haplotypes in the reference panel.

In this work, we present new results on the solution space of the MPSC by first identifying a property that any MPSC will have a set of required regions, and then proposing a MPSC graph. In addition, we derived a number of optimal algorithms for MPSC, including solution enumerations, the Length Maximal MPSC, and  $h$ -MPSC solutions. In doing so, our algorithms reveal the solution space of LS for large panels. Even though we only solved an extreme case of LS where the emission probability is 0, our algorithms can be made more robust by PBWT smoothing. We show that our method is informative in terms of revealing the characteristics of biobank-scale data sets and can improve genotype imputation. Table 1 summarizes the major algorithmic contributions of this paper. The preprint of the full paper is available at <https://www.biorxiv.org/content/10.1101/2023.01.04.522803v1>.

## References

1. Li, N., Stephens, M.: Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics* **165**(4), 2213–2233 (2003)
2. Lunter, G.: Haplotype matching in large cohorts using the Li and Stephens model. *Bioinformatics* **35**(5), 798–806 (2019)
3. Rosen, Y.M., Paten, B.J.: An average-case sublinear forward algorithm for the haploid Li and Stephens model. *Algorithms Mol. Biol.* **14**(1), 1–12 (2019)
4. Sanaullah, A., Zhi, D., Zhang, S.: Haplotype threading using the positional burrows-wheeler transform. In: Boucher, C., Rahmann, S. (eds.) 22nd International Workshop on Algorithms in Bioinformatics (WABI 2022). *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 242, pp. 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022)

# Cell Segmentation for High-Resolution Spatial Transcriptomics

Hao Chen<sup>1</sup>, Dongshunyi Li<sup>1</sup>, and Ziv Bar-Joseph<sup>1,2</sup>(✉)

<sup>1</sup> Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

<sup>2</sup> Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

zivbj@cs.cmu.edu

**Introduction.** Spatial transcriptomics provides information on both the expression and the locations of cells in a sample, enabling the analyses of cell-cell signaling and cell type organization. Spatial transcriptomics uses a set of bar-coded spots placed at regular intervals on the sample to profile the expression of genes. While most current platforms for spatial transcriptomics only provide multi-cellular resolution, newer spatial technologies, such as Stereo-seq [3] and Seq-scope [4], achieve a spot-to-spot distance of 0.5  $\mu\text{m}$  on average, resulting in more than 1000 placed spots per cell on average.

A challenging step in the analysis of such new data is cell segmentation and the integration of spots to assign an expression profile to each cell. Recently, new segmentation methods were developed for spatial proteomics or *in situ* fluorescent hybridization (FISH) based spatial transcriptomics data. However, the predefined spot locations, the large number of genes profiled, and the sparseness of the expression captured by each spot make these methods not appropriate for sequencing-based technologies. Most standard cell segmentation methods developed to date rely on nucleus or membrane staining to identify cell boundaries. While successful, these methods do not fully utilize the information provided by spatial transcriptomics data, leading to less accurate results.

**Methods.** Here we developed SCS (Sub-cellular spatial transcriptomics Cell Segmentation), which combines sequencing and staining data to improve cell segmentation in high-resolution spatial transcriptomics. SCS performs segmentation in three key steps. It first identifies cell nuclei from staining images using the Watershed algorithm [2]. Second, a transformer model infers for each spot whether it is part of a cell or part of the extracellular matrix (background), and its relative position w.r.t. the center of its cell, by adaptively aggregating high-dimensional but sparse gene expression information from neighboring spots via an attention mechanism. To train the model, we used as positive samples spots within the identified nuclei and as negative samples spots sampled from highly confident background regions. Finally, spots that are determined to be part of the cell are grouped by tracking the gradient flow from spots to nucleus centers.

**Results.** We applied SCS to two different high-resolution spatial transcriptomics platforms: a mouse brain dataset profiled using Stereo-seq [3] with nucleus staining, and a mouse liver dataset from Seq-scope [4] with H&E staining. To evaluate

performance, we compared SCS with Watershed cell segmentation, and several popular deep learning-based methods, including Cellpose [6], DeepCell [1], and StarDist [5]. Since the ground truth for cell segmentation does not exist, we compare the expression of regions where two methods (SCS and another method) agree to regions where they disagree using Pearson's  $r$ . We expect that the more correlated the difference region for a method is with the intersection region, the more accurate the segmentation of that method.

Our results indicate that SCS greatly outperforms others methods for segmenting spatial transcriptomics sub-cellular data. On the Stereo-seq dataset, SCS segmentation achieved an average correlation that is 24% higher than Watershed (0.61 vs. 0.49), and at least 13% higher than all other deep learning methods (0.60 vs. 0.53 of DeepCell). While image-based methods on nucleus staining images tend to underestimate cell sizes, SCS can accurately capture cytoplasm regions of cells, leading to more realistic cell size estimation. In addition, some cells were completely missed by image-based methods due to their low staining signal intensity. However, SCS identified such cells by integrating transcriptomics data, leading to at least 1.5% more identified cells when compared to other methods. On the Seq-scope data, the differences were less dramatic due to the use of H&E images. Still, SCS had a higher correlation of 0.88 vs. 0.86 for Watershed and higher correlations than all the other deep learning-based methods. For this data, we observed that when the boundaries of two cells are unclear in the staining, image-based methods tend to merge them, while SCS can correctly separate them relying on transcriptomics data, leading to at least 2.3% more cells identified by SCS.

To characterize molecular heterogeneity within individual cells, we used SCS to investigate how RNAs are distributed within cells. Specifically, we divided each SCS identified cell into two regions, the nucleus region and the cytoplasm region, and identified genes whose RNAs localize differentially between two groups of regions using t-test. Interestingly, in both datasets, RNAs that have been experimentally shown to reside in nucleus or cytoplasm are significantly enriched in our identified RNAs in the corresponding regions.

**Conclusion.** Applications of SCS to two datasets generated using state-of-the-art spatial transcriptomics platforms demonstrate the advantage of our method over the existing image-based methods. While sub-cellular spatial transcriptomics is still very new, we believe that its advantages and ability to provide spatially resolved single-cell information would make it very popular going forward. The ability of SCS to identify more accurate cell boundaries and more cells would make it a useful tool for analyzing such data.

**Code:** <https://github.com/chenhcs/SCS>.

**Preprint:** <https://doi.org/10.1101/2023.01.11.523658>.

**Acknowledgements.** This work was partially supported by NIH grants 1U54AG075931, and 1U24CA268108 and by NSF grant CBET-2134998 to Z.B.J.

## References

1. Bannon, D., et al.: DeepCell Kiosk: scaling deep learning-enabled cellular image analysis with Kubernetes. *Nat. Methods* **18**(1), 43–45 (2021)
2. Beucher, S.: Use of watersheds in contour detection. In: *Proceedings of the International Workshop on Image Processing*. CCETT (1979)
3. Chen, A., et al.: Spatiotemporal transcriptomic atlas of mouse organogenesis using DNA nanoball-patterned arrays. *Cell* **185**(10), 1777–1792 (2022)
4. Cho, C.S., et al.: Microscopic examination of spatial transcriptome using Seq-Scope. *Cell* **184**(13), 3559–3572 (2021)
5. Schmidt, U., Weigert, M., Broaddus, C., Myers, G.: Cell detection with star-convex polygons. In: Frangi, A.F., Schnabel, J.A., Davatzikos, C., Alberola-López, C., Fichtinger, G. (eds.) *MICCAI 2018, Part II*. LNCS, vol. 11071, pp. 265–273. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-00934-2\\_30](https://doi.org/10.1007/978-3-030-00934-2_30)
6. Stringer, C., Wang, T., Michaelos, M., Pachitariu, M.: Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods* **18**(1), 100–106 (2021)



# Unsupervised Deep Peak Caller for ATAC-seq

Yudi Zhang<sup>1</sup>, Ha T. H. Vu<sup>2,3</sup>, Geetu Tuteja<sup>2,3</sup>, and Karin Dorman<sup>1,2,3</sup>(✉)

<sup>1</sup> Department of Statistics, Iowa State University, Ames, IA 50011, USA  
kdorman@iastate.edu

<sup>2</sup> Bioinformatics and Computational Biology Program, Iowa State University, Ames, IA 50011, USA

<sup>3</sup> Department of Genetics, Development and Cell Biology, Iowa State University, Ames, IA 50011, USA

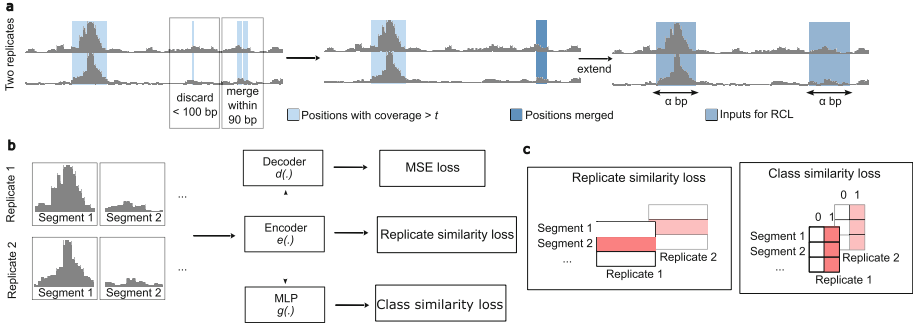
The assay for transposase-accessible chromatin using sequencing (ATAC-seq) identifies accessible chromatin regions using a Tn5 transposase that can access, cut, and ligate adapters to DNA fragments for subsequent amplification and sequencing. These sequenced regions are quantified and tested for enrichment in a process referred to as “peak calling”. Downstream analyses, including motif detection, differential binding analysis or footprint identification, require accurate peak calls. Most unsupervised peak calling methods are based on traditional statistical models and suffer from elevated false positive rates. Such errors can be reduced by masking repetitive regions and using control samples, but input controls for ATAC-seq are typically unavailable due to high sequencing costs. Newly developed supervised deep learning methods can be successful, but they are trained on high quality labeled data, which can be difficult and costly to obtain. Moreover, though biological replicates are recognized to be important, there are no established approaches for using replicates in the deep learning tools. However, joint analysis of multiple biological replicates could improve the power to distinguish true transcription factor binding events, since some weak or highly variable peak signals may only become evident across multiple replicates.

To better address these important challenges, we developed a novel peak caller named Replicative Contrastive Learner (RCL), which uses unsupervised contrastive learning to extract shared peak signals from multiple replicates (Fig. 1). Our method separates peak calling into 1) identifying candidate regions of possible enrichment, then 2) learning to score and classify the regions as peaks. To identify candidate regions (Fig. 1a), genome positions with coverage greater than a threshold  $t$  in all replicates are retained. An algorithm then processes these sites into variable-length *candidate regions* and  $\alpha$  bp *genomic segments* centered roughly on candidate peak summits. Candidate regions are classified and scored via the genomic segments they overlap. The raw coverage vectors of the segments are encoded to a low-dimensional embedding while optimizing a *replicate similarity loss* and a segment *class similarity loss* over biological replicates, as well as an autoencoder *MSE loss* on denoised data (Fig. 1b).

---

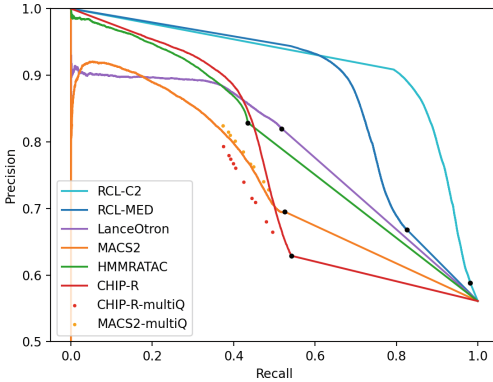
Y. Zhang, H. T.H. Vu—These authors contributed equally to this work.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023  
H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 254–256, 2023.  
<https://doi.org/10.1007/978-3-031-29119-7>



**Fig. 1.** RCL model. (a) Raw coverage is processed to extract  $\alpha$ -length input segments. (b) The segments are fed to encoder  $e(\cdot)$  for cross-replicate contrastive loss. The embedding is fed to a multilayer perceptron (MLP) for class similarity loss and a decoder (MSE) loss. (c) Shaded red boxes represent the elements contrasted in the respective losses.

We use ResNET (He et al. 2016) as the backbone of the encoder. The losses encourage both the low dimensional embeddings and peak probabilities to be similar for the same segment across replicates and distinct for different segments (Fig. 1c).



**Fig. 2.** Precision-Recall curves for ChromHMM regions. Black dots denote the ChromHMM region with lowest called score. RCL-C2, coverage threshold 2; RCL-MED, default coverage threshold; MACS2-multiQ and CHIP-R-multiQ dots indicate performance at typical  $q$ -value cut-offs. (Color figure online)

peaks with little cost to precision (RCL-C2 vs. RCL-MED). On A549, RCL finds twice as many true peaks while maintaining higher precision than either MACS or LanceOtron ever attains. Moreover, peaks called only by RCL (not

We compared RCL to three unsupervised methods, MACS (Zhang et al. 2008), ChIP-R (Newell et al. 2021), and HMM-RATAC (Tarbell and Liu 2019), as well as the pre-trained supervised deep learner LanceOtron (Hentges et al. 2021) on five ATAC-seq datasets from human cell lines and mouse tissues. To evaluate performance we compared peak calls to open and closed chromatin regions identified by ChromHMM (Ernst and Kellis 2017) in the same cells and tissues. RCL consistently dominates the other methods (data A549 in Fig. 2). Decreasing threshold  $t$  exposes the learner to lower coverage candidate regions, allowing RCL to identify weaker

overlapping with peaks called by any other method) are associated with genes expected to be functionally relevant in the examined cell lines and tissues. For most relevant terms, RCL peaks are associated with at least five genes and more than two-fold enrichment, while other methods find no relevant associations.

We developed a novel peak caller for ATAC-seq data using contrastive learning to extract signals shared across biological replicates and identify open chromatin regions. We have focused on ATAC-seq data, where peak calling has been particularly difficult because of the lack of control samples and reliable truth labels. However, our model assumes nothing particular to ATAC-seq data and can be applied to ChIP-seq, CUT&RUN and other techniques requiring peak calling. Run times were acceptable on all tested datasets. For the largest dataset of three replicates with 60 million 1,000 bp segments per replicate, the training time (25 epochs) took about one hour on two GPUs. Because RCL can predict more peaks with higher precision, it will facilitate future epigenome and chromatin accessibility studies in various biological contexts.

Link to the bioRxiv: <https://www.biorxiv.org/content/10.1101/2023.01.07.523108v1.full.pdf>.

## References

- Ernst, J., Kellis, M.: Chromatin-state discovery and genome annotation with ChromHMM. *Nat. Protoc.* **12**(12), 2478–2492 (2017)
- He, K. et al.: Deep residual learning for image recognition. In: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV*, pp. 770–778 (2016)
- Hentges, L.D., et al.: LanceOtron: a deep learning peak caller for ATAC-seq, ChIP-seq, and DNase-seq. *Bioinformatics* **38**(18), 4255–4263 (2021)
- Newell, R., et al.: ChIP-R: assembling reproducible sets of ChIP-seq and ATAC-seq peaks from multiple replicates. *Genomics* **113**(4), 1855–1866 (2021)
- Tarbell, E., Liu, T.: HMMRATAC: a hidden Markov Modeler for ATAC-seq. *Nucleic Acids Res.* **47**(16), e91 (2019)
- Zhang, Y., et al.: Model-based analysis of ChIP-Seq (MACS). *Genome Biol.* **9**(9), R137 (2008)

# Unraveling Causal Gene Regulation from the RNA Velocity Graph Using Velorama

Rohit Singh<sup>1</sup>(✉), Alexander P. Wu<sup>1</sup>, Anish Mudide<sup>2</sup>, and Bonnie Berger<sup>1,3</sup>(✉)

<sup>1</sup> Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA

{rsingh, alexwu, bab}@mit.edu

<sup>2</sup> Phillips Exeter Academy, Street, Exeter, NH 03883, USA

amudide@exeter.edu

<sup>3</sup> Department of Mathematics, MIT, Cambridge, MA 02139, USA

**Motivation.** Gene regulatory network (GRN) inference that incorporates single-cell RNA-seq (scRNA-seq) differentiation trajectories or RNA velocity can reveal causal links between transcription factors and their target genes. However, current GRN inference methods require a total ordering of cells along a linear pseudotemporal axis, which is biologically inappropriate since trajectories with branches cannot be reduced to a single time axis. Such orderings are especially difficult to derive from RNA velocity studies since they characterize each cell's state transition separately.

**Methods.** Here, we present Velorama, a novel method for GRN inference on scRNA-seq data that does not require a total ordering over the differentiation landscape. Velorama takes advantage of recent advances in cell fate mapping using RNA velocity and is the first method to fully incorporate cell-to-cell transition probabilities for GRN inference. Our key conceptual advance is to perform causal GRN inference while modeling the differentiation landscape as a *partial*, rather than total, ordering of cells (Fig. 1). Thus, we model cell-differentiation in a local, rather than global, context, which enables us to better capture cell dynamics.

**Results.** On a standard set of synthetic datasets, we first demonstrate Velorama's use with just pseudotime, finding that it improves area under the precision-recall curve (AUPRC) by 1.5–3x over state-of-the-art approaches. Using RNA velocity instead of pseudotime as the input to Velorama further improves AUPRC by an additional 1.8–3x. We also applied Velorama to study cell differentiation in pancreas, dentate gyrus, and bone marrow from real datasets and obtained intriguing evidence for the relationship between regulator interaction speeds and mechanisms of gene regulatory control during differentiation. We expect Velorama to be a critical part of the RNA velocity toolkit for investigating the causal drivers of differentiation and disease.

**Implementation.** <https://github.com/rs239/velorama>.

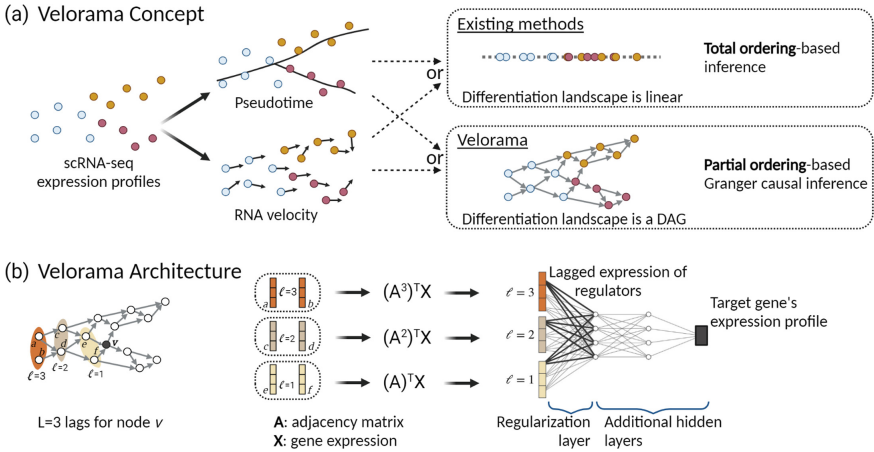
**Full Text Preprint.** <https://www.biorxiv.org/content/10.1101/2022.10.18.512766v>.

R. Singh, A. P. Wu and A. Mudide—These authors contributed equally to this work.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 257–258, 2023.

<https://doi.org/10.1007/978-3-031-29119-7>



**Fig. 1.** Schematic of Velorama: Existing methods need to impose a total (i.e., linear) ordering on cells in order to perform gene regulatory network (GRN) inference. When supplied with RNA velocity, they convert it to a global pseudotime measure and do not make use of the cell transition matrix. **(a)** Our key insight is that the differentiation landscape is more faithfully captured by a partial ordering, or directed acyclic graph (DAG), of cells constructed from RNA velocity (or pseudotime, as available) data. **(b)** We perform Granger causal inference on this DAG, taking advantage of a recent advance in generalizing Granger causality with the use of graph neural networks. At each node (cell), we collect the expression of putative regulators (e.g., three genes per cell) at the cell’s ancestor nodes (e.g., we look back three lags). These are applied to predict the target gene’s expression. A key component of our neural network is the first hidden layer where regularization of the weights is performed to identify statistically significant regulator–target relationships.

**Acknowledgements.** AW, RS and BB were supported by the NIH grant R35GM141861. Some figures were made using [biorender.com](https://biorender.com).

# PIsToN: Evaluating Protein Binding Interfaces with Transformer Networks

Vitalii Stebliankin<sup>1</sup>, Azam Shirali<sup>1</sup>, Prabin Baral<sup>2</sup>, Prem Chapagain<sup>2,3</sup>,  
and Giri Narasimhan<sup>1,3</sup>(✉)

<sup>1</sup> Bioinformatics Research Group (BioRG), KFSCIS, Florida International University, 11200 SW 8th St., Miami 33199, USA  
giri@fiu.edu

<sup>2</sup> Department of Physics, College of Arts, Science and Education, Florida International University, 11200 SW 8th St., Miami 33199, USA

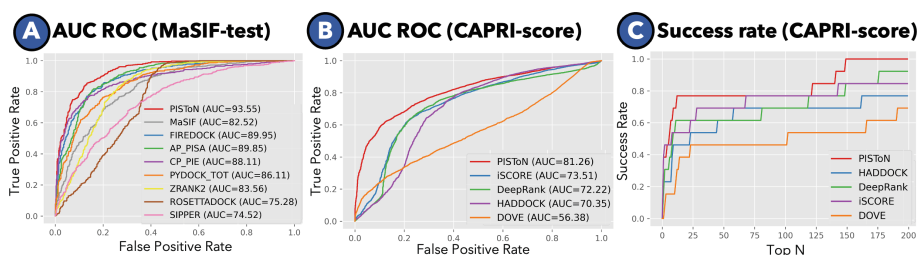
<sup>3</sup> Biomolecular Sciences Institute, Florida International University, 11200 SW 8th St., Miami 33199, USA

**Introduction.** With the advent of tools such as AlphaFold [6] that are based on machine learning, protein structure prediction has become more tractable. The next challenge in this domain is protein docking. The docking programs predict many binding sites for two target proteins, which are then ranked with a scoring function. The best docking tools have been able to ensure that the optimal binding conformation appears within the top 100 predicted docking structures, but rarely do they appear in the top 10, and even more rarely in the top position [9]. Therefore, improving scoring/evaluation functions would greatly benefit existing docking tools.

**Results.** We developed a tool called **Protein Interface Scoring with Transformer Network (PIsToN)** that has the following novel contributions:

- *Data representation.* We represent interfaces of protein complexes as 2D multi-channel images. Similar to the MaSIF approach [4], we compute circular “patches” on protein surfaces [4]. Next, we convert patches into images with pixel intensities corresponding to feature values associated with surface points (geometric and physico-chemical). Unlike the single-patch approach of MaSIF, we consider pairs of patches from protein binding interfaces, allowing us to compute essential interaction properties: distance between atoms, relative accessible surface area (*RASA*) [10], Van der Waals interactions [1], and more.
- *Vision transformer network.* We provide novel adaptations to the vision transformer (ViT) model [3], thus improving predictive performance and providing explainability. Since ViTs are best suited for image classification, the choice of images for the representation of the features is an ideal complement.
- *Hybrid machine learning.* The ViT is enhanced with a hybrid component that combines energy terms with surface feature representations.
- *Multi-attention and explainability.* We introduce dual attention: spatial attention to highlight essential binding sites and feature attention to identifying the relative importance of specific protein properties. The embeddings of each protein property are combined using the transformer encoder.

- *Contrastive mini-batch training.* Each training iteration consists of multiple views of acceptable and incorrect binding poses of the same protein complex, labeled using Critical Assessment of Predicted Interactions (CAPRI) criteria [5]. The combination of supervised contrastive [7], margin ranking [2], and binary cross-entropy terms in the loss function help to cluster the correct docking models in the embedding space while pulling apart incorrect predictions.
- *Superior Performance.* PISToN had a superior AUC ROC measure and success rate on MaSIF test [4] and CAPRI Score [8] datasets (Fig. 1).



**Fig. 1. PISToN performance.** **A)** AUC ROC on MaSIF test dataset; **B)** AUC ROC on CAPRI score set; **C)** The success rate on the CAPRI score set (% of complexes for which at least one model of acceptable quality is found in the top  $N$  selected models).

**Conclusion.** The PISToN deep learning model outperforms the state-of-the-art scoring functions in identifying viable protein binding interfaces. Given docking models for two protein targets, PISToN is more accurate in placing the correct configuration among the top predictions.


## References

1. Andrusier, N., et al.: FireDock: fast interaction refinement in molecular docking. *Proteins Struct. Funct. Bioinf.* **69**(1), 139–159 (2007)
2. Chen, C., et al.: This looks like that: deep learning for interpretable image recognition. *Adv. Neural Inf. Process. Syst.* **32** (2019)
3. Dosovitskiy, A., et al.: An image is worth  $16 \times 16$  words: Transformers for image recognition at scale. *arXiv* (2020)
4. Gainza, P., et al.: Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nat. Methods* **17**(2), 184–192 (2020)
5. Janin, J., et al.: CAPRI: a critical assessment of predicted interactions. *Proteins Struct. Funct. Bioinf.* **52**(1), 2–9 (2003)
6. Jumper, J., et al.: Highly accurate protein structure prediction with AlphaFold. *Nature* **596**(7873), 583–589 (2021)
7. Khosla, P., et al.: Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **33**, 18661–18673 (2020)

8. Lensink, M.F., et al.: Score\_set: a CAPRI benchmark for scoring protein complexes. *Proteins Struct. Funct. Bioinf.* **82**(11), 3163–3169 (2014)
9. Moal, I.H., et al.: The scoring of poses in protein-protein docking: current capabilities and future directions. *BMC Bioinf.* **14**(1), 1–15 (2013)
10. Touw, W.G., et al.: A series of PDB-related databanks for everyday needs. *Nucleic Acids Res.* **43**(D1), D364–D368 (2015)



# DebiasedDTA: A Framework for Improving the Generalizability of Drug-Target Affinity Prediction Models

Rıza Özçelik<sup>1</sup> , Alperen Bağ<sup>1</sup> , Berk Atıl<sup>1</sup> , Melih Barsbey<sup>1</sup> ,  
Arzucan Özgür<sup>1</sup> , and Elif Ozkirimli<sup>2</sup>  

<sup>1</sup> Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey  
arzucan.ozgur@boun.edu.tr

<sup>2</sup> Data and Analytics Chapter, Pharma International Informatics, F. Hoffmann-La Roche AG, Kaiseraugst, Switzerland  
elif.ozkirimli@roche.com

## 1 Introduction

Proteins and chemicals interact with each other by following fundamental physicochemical principles, and a complete understanding of these principles would allow the accurate prediction of protein-chemical interactions. Without such an understanding, machine learning approaches that rely on the available knowledge space of large interaction datasets can help to rapidly identify high-affinity protein-chemical pairs in the vast combination space by learning affinity patterns from measurements for millions of protein-chemical pairs. However, a representative sampling of the entire combination space is impossible or infeasible, and thus the available datasets explore only portions of this space. Previous work suggests that these limited datasets may contain potentially misleading spurious patterns and may lead to prediction models that do not generalize to molecules outside of the dataset, for which the learned patterns are non-applicable [3]. Here, we propose DebiasedDTA, a novel model training framework to improve the generalizability of DTA prediction models.

## 2 DebiasedDTA

The DebiasedDTA training framework comprises two stages: “guide” and “predictor” models. The guide learns a weighting of the dataset such that a model trained thereupon can learn a robust relationship between biomolecules and binding affinity instead of spurious associations. The predictor then uses the weights produced by the guide and progressively weights the training data during its training to generalize well to unseen biomolecules.

The guide in DebiasedDTA targets the low-complexity spurious relationships in the data and, therefore, should have a limited learning capacity. We design a

---

A. Bağ and B. Atıl—Equal contribution.

weak learner with simple biomolecule representations: ID-DTA. ID-DTA represents the chemicals and proteins in the training set with one-hot-encoded vectors and uses decision trees for regression, as decision trees have limited learning capacity. DebiasedDTA is model-agnostic for the predictor; in that, any DTA prediction model that allows instance weighting can be trained with Debiased-DTA training framework. Here, we experiment with GraphDTA [1].

Here we report the experiment results for BDB dataset [2] which contains four test splits: warm, cold (unseen) ligand, cold (unseen) protein, and cold (unseen) both. We measure concordance index on the test sets and compare DebiasedDTA with a method proposed to improve the generalizability of ligand-based drug-target interaction prediction models, asymmetric validation embedding (AVE) [3], since there is no comparable study for structure-based DTA prediction.

**Table 1.** The effect of DebiasedDTA framework on the generalizability of GraphDTA.

Debiasing	Warm	Cold ligand	Cold protein	Cold both
None	0.824 (0.010)	0.701 (0.024)	0.685 (0.039)	0.558 (0.077)
AVE	0.825 (0.017)	0.716 (0.049)	0.692 (0.027)	0.581 (0.082)
DebiasedDTA	0.832 (0.012)	0.728 (0.039)	0.695 (0.027)	0.603 (0.050)

### 3 Results

The results of experiments are displayed in Table 1. Table 1 shows that DebiasedDTA can improve the performance of GraphDTA on seen and unseen biomolecules (compared to no debiasing), suggesting that DebiasedDTA can improve the generalizability of a DTA prediction model. The results also indicate that DebiasedDTA can outperform AVE, a state-of-the-art method to improve the generalizability of ligand-based affinity prediction models. The results for more datasets, guides, and predictors are available in our preprint.

### 4 Conclusion

DebiasedDTA is a model training framework that aims to improve the generalization performance of DTA prediction models. The out-of-distribution generalization is a notoriously difficult problem and our results point to a promising direction to enhance the toolbox for generalization in DTA prediction.

**Acknowledgements.** This work was supported by the Scientific and Technological Research Council of Turkey [Grant Number 119E133].

## References

1. Nguyen, T., Le, H., Quinn, T.P., Nguyen, T., Le, T.D., Venkatesh, S.: GraphDTA: predicting drug-target binding affinity with graph neural networks. *Bioinformatics* **37**(8), 1140–1147 (2021)
2. Özçelik, R., Öztürk, H., Özgür, A., Ozkirimli, E.: ChemBoost: a chemical language based approach for protein - ligand binding affinity prediction. *Mol. Inform.* **40**(5), 2000212 (2021)
3. Wallach, I., Heifets, A.: Most ligand-based classification benchmarks reward memorization rather than generalization. *J. Chem. Inf. Model.* **58**(5), 916–932 (2018)

# Drug Synergistic Combinations Predictions via Large-Scale Pre-training and Graph Structure Learning

Zhihang Hu<sup>1</sup>, Qinze Yu<sup>1</sup>, Yucheng Guo<sup>2</sup>, Taifeng Wang<sup>2</sup>, Irwin King<sup>1</sup>,  
Xin Gao<sup>3</sup>, Le Song<sup>2</sup>, and Yu Li<sup>1,4</sup>(✉)

<sup>1</sup> Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Hong Kong SAR, China

liyucse@cuhk.edu.hk

<sup>2</sup> BioMap (Beijing) Intelligence Technology Limited, Beijing, China

<sup>3</sup> Computational Bioscience Research Center, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

<sup>4</sup> The CUHK Shenzhen Research Institute, Hi-Tech Park, Nanshan, Shenzhen, China

**Abstract.** Drug combination therapy is a well-established technique for disease treatment that improves efficacy while decreasing safety risks. In this study, we present a computational predictive pipeline that makes use of a wide range of biological resources as well as current breakthroughs in deep learning.

**Keywords:** Drug discovery · Deep learning · Biological networks · Synergistic effect

## 1 Introduction

Drug combination therapy has been widely applied in both traditional and modern medicine due to its diverse merits. Compared with monotherapy, administering drug combinations leads to improvement of efficacy [1], and reduction of side effects [2] and host toxicity [3], further, it even overcomes drug resistance [4]. Considering the fact that a single drug usually cannot be effective, drug combinations are increasingly used to treat a variety of complex diseases, such as human immunodeficiency virus (HIV) [5], virus infections [6], and cancer [7, 8]. Computational methods have emerged to tackle the limitations of traditional wet-lab tests, which are confined to a small number of drugs and thus are unable of exploring the large combinatorial search space. Current computational-based methods often comprise only one modality and rarely make use of recent advances in deep learning research. Although the reported results on Drug Comb are pretty high, their use of drug or cell line features usually result in false predictions on domain-shift datasets.

Z. Hu and Q. Yu—Equal first authorship.

Full-paper arxiv: <https://arxiv.org/abs/2301.05931>.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 265–267, 2023.

<https://doi.org/10.1007/978-3-031-29119-7>

## 2 Methods

In this study, we address the problems mentioned above by proposing an end-to-end deep learning framework that accurately predicts synergistic effects. Our method takes advantage of multi-modal data, graph neural networks, and large-scale unsupervised training to integrate and learn useful information for synergistic prediction. Specifically, our model takes chemical structure graphs of drugs and the protein expression of cell lines as input and applies a pre-trained molecular graph transformer to convert drug graphs into embeddings. Meanwhile, the model generates embeddings for every protein in the expression by utilizing a protein language model [9]. To enrich more features, we also include disease information, particularly, we apply RotatE [10] to get the embedding of disease from PrimeKG [11]. Next, we utilize graph neural networks and take our generated embeddings as node representations. In order to inference on unseen drugs, we include drug-drug similarity edge and drug-target module/ drug-drug interaction module to generate pseudo edges and formed a refined graph with richer information. Finally, a synergistic prediction head is built on top of our graph and acts as a Perceptron (MLP) to predict the synergistic effect. We also incorporate a self-training strategy to benefit from the large amount of information in the combination space.

## 3 Results

Our framework achieves state-of-the-art results in comparison with other deep learning-based methods on synergistic prediction benchmark datasets. We gained 2% AU ROC improvement on DrugComb dataset where well-performing methods exceed over 90%. When inferencing new drug combinations on an independent set released by AstraZeneca, a 10% of improvement over previous methods is observed. In addition, we're robust against unseen drugs and surpass almost 15% AU ROC compared to the second-best model. We also presented the findings of ablation studies, which show that our self-training technique and predictive modules mine more meaningful information about drugs and proteins and that our pretrained embeddings also contributed significantly to our performance improvement.

## 4 Conclusion

We developed an end-to-end model that aggregates various types of drug-related information to aid in discovering drug combinations. Furthermore, certain deep learning advances are first integrated into our framework to improve our performance. Truthfully, there are still many issues to be resolved in this field, including isolated drugs (which exhibit a huge dissimilarity to our database) or perhaps more precise regression values. We will investigate all these aspects further, and we believe this work will benefit the community.

## References

1. Csermely, P., Korcsmáros, T., Kiss, H.J.M., London, G., Nussinov, R.: Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review. *Pharmacol. Ther.* **138**(3), 333–408 (2013)
2. Zhao, S., et al.: Systems pharmacology of adverse event mitigation by drug combinations. *Sci. Transl. Med.* **5**(206), 206ra140 (2013)
3. O’Neil, J., et al.: An unbiased oncology compound screen to identify novel combination strategies. *Mol. cancer Ther.* **15**(6), 1155–1162 (2016)
4. Hill, J.A., Ammar, R., Torti, D., Nislow, C., Cowen, L.E.: Genetic and genomic architecture of the evolution of resistance to antifungal drug combinations. *PLoS Genet.* **9**(4), e1003390 (2013)
5. De Clercq, E.: The design of drugs for HIV and HCV. *Nat. Rev. Drug discovery* **6**(12), 1001–1018 (2007)
6. Zheng, W., Sun, W., Simeonov, A.: Drug repurposing screens and synergistic drug-combinations for infectious diseases. *Brit. J. Pharmacol.* **175**(2), 181–191 (2018)
7. Kim, Y., Zheng, S., Tang, J., Zheng, W.J., Li, Z., Jiang, X.: Anticancer drug synergy prediction in understudied tissues using transfer learning. *J. Am. Med. Inform. Assoc.* **28**(1), 42–51 (2021)
8. Al-Lazikani, B., Banerji, U., Workman, P.: Combinatorial drug therapy for cancer in the post-genomic era. *Nat. Biotechnol.* **30**(7), 679–692 (2012)
9. Rives, A., et al.: Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Nat. Acad. Sci.* **118**(15), e2016239118 (2021)
10. Sun, Z., Deng, Z.-H., Nie, J.-Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint [arXiv:1902.10197](https://arxiv.org/abs/1902.10197) (2019)
11. Chandak, P., Huang, K., Zitnik, M.: Building a knowledge graph to enable precision medicine. *bioRxiv* (2022)

# Pisces: A Cross-Modal Contrastive Learning Approach to Synergistic Drug Combination Prediction

Jiacheng Lin<sup>1</sup>, Hanwen Xu<sup>2</sup>, Addie Woicik<sup>2</sup>, Jianzhu Ma<sup>3</sup>,  
and Sheng Wang<sup>2</sup>(✉)

<sup>1</sup> Department of Automation, Tsinghua University, Beijing, China

<sup>2</sup> Paul G. Allen School of Computer Science and Engineering,  
University of Washington, Seattle, WA, USA

[swang@cs.washington.edu](mailto:swang@cs.washington.edu)

<sup>3</sup> Institute for Artificial Intelligence, Peking University, Beijing, China

Drug combination therapy is a promising solution to many complex diseases, such as breast cancer, colorectal cancer, Alzheimer's disease, and diabetes. However, previous studies have also pointed out the rareness of synergistic drug combinations. Since experimentally testing millions of candidate combinations is not scalable, there is a pressing need to develop computational approaches to identify synergistic drug combinations.

We focus on cancer drug synergistic prediction and follow existing approaches to form the synergistic drug combination prediction problem as a triplet classification problem. Each triplet consists of two drugs and a cancer cell line and will be classified into synergistic or not. Each cell line is represented using its genomics features, such as gene expression and somatic mutation. A key technical challenge is to derive an effective representation for a pair of drugs. Simplified molecular-input line-entry system (SMILES) sequences and molecular graphs are the two major modalities for representing a drug. Both of them have strengths and limitations: SMILES sequences are easier to embed by leveraging the recent progress in natural language processing, but are ambiguous on drugs with complex structures; molecular graphs precisely characterize the molecular information, but large graphs (i.e., large diameter) are often hard to embed. This dilemma is even more severe when we want to embed a pair of drugs since one might be better represented using SMILES sequences and the other might be better represented using molecular graphs. As we showed in the experiments, a simple concatenation of these two kinds of features yields undesirable results.

To address this problem, we propose Pisces, a cross-modal contrastive learning approach for drug synergy prediction. Our intuition is that the SMILES sequence modality and the molecular graph modality complement each other, and thus should be integrated. To realize this intuition, we have developed a cross-modal contrastive learning framework, as illustrated in Fig. 1. We propose to create four augmented views for each drug pair based on the combination of the SMILES sequence and the molecular graph modality. We hypothesize that

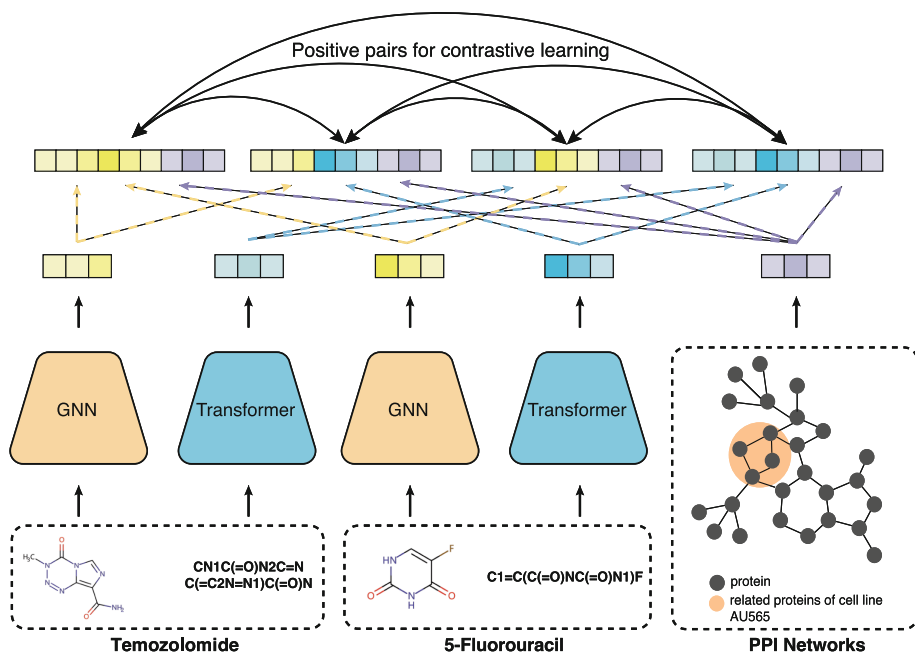
---

J. Lin and H. Xu—Contributed equally.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

H. Tang (Ed.): RECOMB 2023, LNBI 13976, pp. 268–270, 2023.

<https://doi.org/10.1007/978-3-031-29119-7>



**Fig. 1. Flowchart of Pisces.** Pisces considers both the SMILES sequence and the molecular graph of each drug. It first uses Transformer to embed SMILES sequences and graph neural networks to embed molecular graphs. Pisces embeds each cell line by aggregating neighbors of over-expressed genes in the protein-protein interaction network. It then concatenates the SMILES embedding and the graph embedding between two drugs, which creates four different views for each drug combination. These four different views are treated as augmentations in contrastive learning.

these four combinations can offer a comprehensive view of a drug pair, thus enhancing the drug synergistic prediction.

We evaluated our method on a recently published large-scale cancer drug synergy dataset GDSC-Combo, which covers 102,893 drug combinations spanning over 63 drugs and 125 cell lines. We first observed a substantial discrepancy between the prediction performance by using two different modalities. By contrasting these two modalities, Pisces substantially outperformed five existing drug combination prediction approaches under vanilla cross validation setting, stratified cross validation for drug combinations setting, and stratified cross validation for cell lines setting. Finally, we found that two drugs from the top performed drug pairs favored different modalities, again confirming the effectiveness of integrating SMILES and molecular graph modalities. Despite the large amount of triplet combinations that have been measured in GDSC-Combo, it still only covers 20.7% of all possible triplet combinations. Pisces offers an *in silico* solution to massively generalize these *in vitro* measurements. In addition to drug synergy prediction, Pisces can be broadly applied to other applications



that require the modeling of drug pairs, such as drug-drug interaction prediction, as well as further integrating other drug modalities.

Link to the bioRxiv preprint: <https://www.biorxiv.org/content/10.1101/2022.11.21.517439v1.full.pdf>.

# Modeling and Predicting Cancer Clonal Evolution with Reinforcement Learning

Stefan Ivanovic<sup>1</sup>  and Mohammed El-Kebir<sup>2</sup>  

<sup>1</sup> Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61801, USA

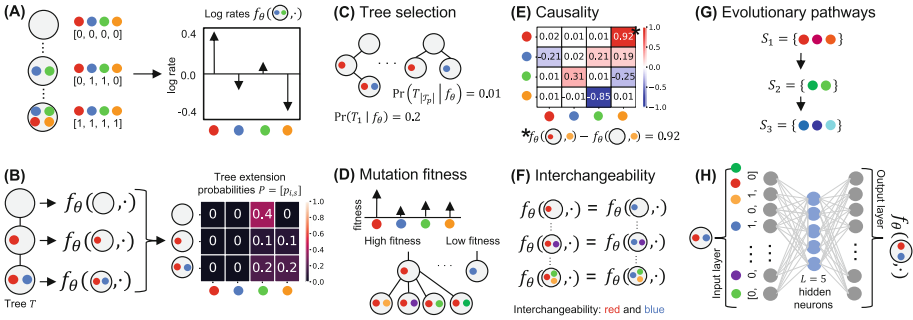
<sup>2</sup> Cancer Center at Illinois, University of Illinois Urbana-Champaign, Champaign, IL 61801, USA

melkebir@illinois.edu

*Introduction:* Cancer forms in an evolutionary process in which somatic mutations occur in cells, forming clones with different sets of mutations. Modeling this genetic process is important for understanding cancer. Existing methods for modeling cancer evolution from cohort-level phylogenies consist of tree generative models [1, 2], tree consensus methods [3–5], and statistical tests [6]. These methods are used to reduce phylogeny uncertainty, detect causal relationships between pairs of mutations, detect interchangeable mutations, detect pathways of interchangeable mutations, and predict mutation fitness. We introduce CloMu (Clone To Mutation) a tree-generative model, which effectively performs all of these tasks while existing methods each complete only a subset of these tasks.

*Methods:* CloMu solves the INDEPENDENT CLONAL EVOLUTION problem, which takes as input sets  $\mathcal{T}_1, \dots, \mathcal{T}_n$  of possible phylogeny trees for each patient on  $m$  total mutations and seeks to identify model parameters  $f_\theta$  that maximize the data probability  $\Pr(\mathcal{T}_1, \dots, \mathcal{T}_n | f_\theta)$ . Like TreeMHN [1] and HINTRA [2], we assume *independent clonal evolution* such that the rate  $f_\theta(\mathbf{c}, s)$  of a clone  $\mathbf{c} \in \{0, 1\}^m$  acquiring a new mutation  $s \in [m]$  only depends on the genotype  $\mathbf{c}$  of that clone. CloMu represents  $f_\theta$  using a two-layer neural network with a small number  $L \ll m$  of hidden neurons. The network is trained using reinforcement learning. After training the parameters once for each dataset, the learned model can be post-processed to complete the previously mentioned tasks.

*Results:* We benchmarked CloMu using simulated and real data. One set of simulations consisted of pairs of mutations with bidirectional causal relationships. Our method outperformed all existing methods, especially in the presence of interchangeable mutations. We also constructed simulations with pathways of interchangeable mutations. CloMu produced latent representations, which accurately identified interchangeable mutations. Additionally, CloMu accurately predicted pathways of interchangeable mutations, greatly outperforming heuristics based on existing methods. On a breast cancer cohort [7], CloMu’s fitness predictions matched known driver mutations, and CloMu’s causal relationships predictions matched TreeMHN in cases where this is theoretically expected. On an AML cohort [8], CloMu’s causal relationship predictions matched TreeMHN



**Fig. 1. Overview of CloMu.** (A) Under the independent clonal evolution assumption, our model determines a log rate  $f_{\theta}(\mathbf{c}, s)$  of any clone  $\mathbf{c} \in \{0, 1\}^m$  acquiring a mutation  $s \in [m]$ . (B) This in turn enables us to compute probabilities  $P = [p_{i,s}]$  that the next mutation to occur on a tree  $T$  is  $s$  at node/clone  $\mathbf{c}_i$ . (C-G) We use the model for five prediction tasks. (H) CloMu represents  $f_{\theta}$  using a low-parameter, two-layer neural network trained via reinforcement learning.

when expected and CloMu’s fitness predictions were orthogonally validated by clonal prevalence measurements (Fig. 1).

*Conclusion:* We introduced CloMu, a tree generative model of cancer evolution trained using reinforcement learning. CloMu outperformed existing methods on simulations in a wide variety of tasks and gave validated predictions on real data. In the future, our work may be expanded by supporting additional types of mutations such as copy-number aberrations, or by utilizing our latent representations for patient outcome predictions. Larger future data sets may also enable accurate prediction of complex mutation interactions on real data.

*Availability:* The code for CloMu and the data used are both available at <https://github.com/elkebir-group/CloMu>.

## References

1. Luo, X.G., et al.: Joint inference of exclusivity patterns and recurrent trajectories from tumor mutation trees. bioRxiv (2022)
2. Khakabimamaghani, S., et al.: Collaborative intra-tumor heterogeneity detection. Bioinformatics **35**(14), i379–i388 (2019)
3. Christense, G., et al.: Detecting repeated cancer evolution from multi-region tumor sequencing data. Nat. Methods **15**, 707–714 (2018)
4. Christensen, S., et al.: Detecting evolutionary patterns of cancers using consensus trees. Bioinformatics **36**(Supplement\_2), i684–i691 (2020)
5. Hodzic, E., et al.: Identification of conserved evolutionary trajectories in tumors. Bioinformatics **36**(Supplement\_1), i427–i435 (2020)
6. Kuipers, J., et al.: Statistical tests for intra-tumour clonal co-occurrence and exclusivity. PLOS Comput. Biol. **17**(12), e1009036 (2021)

7. Razavi, P., et al.: The genomic landscape of endocrine-resistant advanced breast cancers. *Cancer Cell* **34**(3), 427–438 (2018)
8. Morita, K., et al.: Clonal evolution of acute myeloid leukemia revealed by high-throughput single-cell genomics. *Nat. Commun.* **11**(1), 1–17 (2020)

# Enabling Trade-Offs in Privacy and Utility in Genomic Data Beacons and Summary Statistics

Rajagopal Venkatesaramani<sup>1</sup>(✉), Zhiyu Wan<sup>2</sup>, Bradley A. Malin<sup>2</sup>,  
and Yevgeniy Vorobeychik<sup>1</sup>(✉)

<sup>1</sup> Washington University in St. Louis, St. Louis, MO 63130, USA  
{rajagopal,yvorobeychik}@wustl.edu

<sup>2</sup> Vanderbilt University Medical Center, Nashville, TN 37203, USA  
zhiyu.wan@vanderbilt.edu, b.malin@vumc.org

The increased sharing of genomic data over the past decade has sparked acute privacy concerns. To address these concerns, data custodians often resort to only releasing summary-level information, such as alternate allele frequencies (AAF) for single nucleotide variants (SNVs) in a given dataset, which was initially thought to sufficiently protect individual privacy. A notable example is the Beacon service [3] introduced by the Global Alliance for Genomics and Health (GA4GH), which is a web service that only allows queries about the presence or absence of alternate alleles in a dataset. However, research [5, 6] has shown that even such summary statistics are vulnerable to membership inference attacks. These attacks leverage likelihood ratio test (LRT) scores, whereby an attacker, given a target genome, computes the log-likelihood that the target individual was part of the dataset  $D$  over which the summary statistics were computed, compared to the likelihood that they were not. Typically, the attacker identifies a threshold  $\theta$  and claims that the target individual was part of the dataset (i.e.,  $i \in D$ ) if the target's LRT score lies below  $\theta$ . Membership inference allows the attacker to potentially infer sensitive information about the individual such as underlying medical conditions, based on dataset metadata. Approaches used to prevent such attacks typically involve adding noise to the summary information using heuristics or differentially-private mechanisms (flipping bits in the case of Beacons, and adding real-valued noise to AAF releases), or suppressing information release for a subset of SNVs (masking).

In this work, we consider both forms of summary statistics releases: a) the Beacon service, where the system's responses are binary indicators of the presence of certain alleles, and b) the release of alternate allele frequencies for a given set of single nucleotide variants (SNVs). We present a rigorous optimization framework that enables the data custodian to combine the addition of noise (either binary or real-valued, depending on the release) with masking and finely tune the privacy-utility tradeoff as desired, in contrast to state-of-the-art methods which rely on only one mode of data obfuscation. Following the model

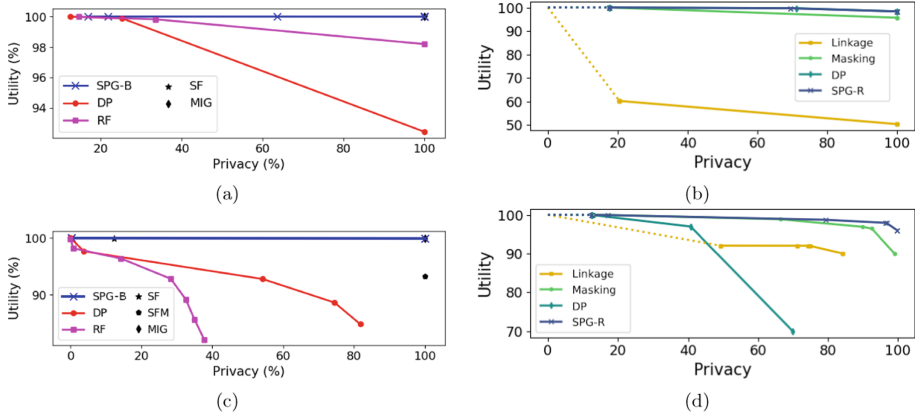
---

We acknowledge support of this work by the National Institutes of Health (NIH) under grant RM1HG009034 and the National Science Foundation (NSF) CAREER award program under grant IIS-1905558.

presented in [8], we further consider two attack models: a) a *fixed-threshold* model where the attacker identifies an LRT score threshold *a priori*, using either simulated or external data according to a maximum allowable false positive rate, and b) an *adaptive-threshold* model where the attacker identifies a threshold which best separates the LRT scores of individuals in the dataset from those who are not *after* the defense is implemented. We capture the latter as the threshold being set to the mean of the lowest  $K$  percentile of LRT scores for a set of individuals not in the dataset  $D$ . Our goal is to solve an optimization problem capturing the privacy-utility tradeoff, taking into account the relative impact of adding noise to masking SNVs on utility, as well as the relative importance of utility versus privacy using exogenous parameters. We call this the SUMMARY STATS PRIVACY PROBLEM (SSPP).

Solving the SSPP problem optimally has exponential worst-case runtime. Therefore, to approximately solve SSPP, we propose highly scalable algorithms which leverage greedy heuristics (which we call SOFT PRIVACY GREEDY (SPG)). In the case of Beacons, we compute the average marginal contribution of flipping each SNV, as well as the average marginal impact of masking each SNV on the LRT score, normalized by the relative costs of flipping and masking (user specifies the cost of flipping as  $\alpha$ , and the cost of masking is  $1 - \alpha$ ). We then rank the SNVs in decreasing order of their marginal contributions, and greedily either flip or mask SNVs until a desired level of privacy is achieved (using a user-specified parameter,  $w$ ). We assume that the subset of flipped SNVs and the subset of masked SNVs are disjoint. In the case of AAF releases, we propose a greedy heuristic that alternates between masking SNVs, and adding real-valued noise to the remaining release using a differentially-private Laplace mechanism.

Our experiments were conducted on a dataset based on the 1000 Genomes Project [1] made available as part of the 2016 iDash workshop on Privacy and Security [7], consisting of 800 individuals and over 1.3 million SNVs. We compare our approach to state-of-the-art baseline approaches, including differentially private mechanisms [2], randomized noise on rare alleles [4], strategic flipping using differential discriminative power [9], linkage-equilibrium based suppression [5], and prior greedy methods [8]. We also evaluate performance against a more powerful adversary that attempts to infer missing or flipped SNVs using linkage disequilibrium as a metric for correlation between SNVs. Our approach scales easily to 1.3 million SNVs, and the results presented in Fig. 1 demonstrate that our approach outperforms prior art in both utility and privacy.



**Fig. 1.** Utility-Privacy plots for the various attack models. a) Fixed threshold model, Beacon release,  $\theta = -250$ , baselines flip SNVs, b) Fixed threshold model, AAF release,  $\theta = 0$ , c) Adaptive threshold model, Beacon release,  $K = 10$ , baselines flip SNVs, d) Adaptive threshold model, AAF release,  $K = 10$ .

## References

1. Genomes Project Consortium, et al.: A global reference for human genetic variation. *Nature* **526**(7571), 68 (2015)
2. Cho, H., Simmons, S., Kim, R., Berger, B.: Privacy-preserving biomedical database queries with optimal privacy-utility trade-offs. *Cell Syst.* **10**(5), 408–416 (2020)
3. Fiume, M., et al.: Federated discovery and sharing of genomic data using beacons. *Nat. Biotechnol.* **37**(3), 220–224 (2019)
4. Raisaro, J.L., et al.: Addressing beacon re-identification attacks: quantification and mitigation of privacy risks. *J. Am. Med. Inform. Assoc.* **24**(4), 799–805 (2017)
5. Sankararaman, S., Obozinski, G., Jordan, M.I., Halperin, E.: Genomic privacy and limits of individual detection in a pool. *Nat. Genet.* **41**(9), 965–967 (2009)
6. Shringarpure, S.S., Bustamante, C.D.: Privacy risks from genomic data-sharing beacons. *Am. J. Hum. Genet.* **97**(5), 631–646 (2015)
7. Tang, H., Wang, X., Wang, S., Jiang, X.: idash privacy and security workshop (2016). [www.humangenomeprivacy.org/2016/](http://www.humangenomeprivacy.org/2016/)
8. Venkatesaramani, R., Wan, Z., Malin, B.A., Vorobeychik, Y.: Defending against membership inference attacks on beacon services. arXiv preprint [arXiv:2112.13301](https://arxiv.org/abs/2112.13301) (2021)
9. Wan, Z., Vorobeychik, Y., Kantarcioglu, M., Malin, B.: Controlling the signal: practical privacy protection of genomic data sharing through beacon services. *BMC Med. Genomics* **10**(2), 87–100 (2017)

# Accurate Evaluation of Transcriptomic Re-identification Risks Using Discriminative Sequence Models

Shuvom Sadhuka<sup>1,2</sup>, Daniel Fridman<sup>2,3</sup>, Bonnie Berger<sup>1,2</sup>,  
and Hyunghoon Cho<sup>2(✉)</sup>

<sup>1</sup> Massachusetts Institute of Technology, Cambridge, MA 02139, USA

<sup>2</sup> Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA  
[hhcho@broadinstitute.org](mailto:hhcho@broadinstitute.org)

<sup>3</sup> Harvard University, Cambridge, MA 02138, USA

## 1 Introduction

Our understanding of genomic privacy is rapidly evolving as new data modalities expose new routes for potential breaches. Transcriptomic data, such as gene expression measurements shared in databases like NCBI GEO, is a prominent example of biomedical data for which our understanding of privacy implications is incomplete. Prior works have shown that knowledge of expression quantitative trait loci (eQTLs) could be used to match genotypes to gene expression profiles, also known as a linking attack [1, 2]. Such a linkage could in turn lead to re-identification of individuals. However, existing methods can analyze only a fraction of known independent eQTLs due to restrictive model assumptions, leaving the full extent of this risk incompletely understood. Our work introduces *discriminative sequence models* (DSMs), a novel probabilistic framework for predicting a sequence of genotypes based on gene expression data. By modeling the joint distribution over all variants in a genomic region with linkage disequilibrium, DSMs enable an accurate assessment of the power of linking attacks that leverage all known eQTLs with calibration for redundant predictive signals.

## 2 Methods

Our discriminative sequence models (DSMs) enable *sequence-level* inference of genotypes given a gene expression profile. DSMs score the likelihood of a target genetic sequence belonging to the same individual as a query gene expression profile. The predictive probabilities of DSMs are calibrated during training to correctly adjust for correlation, i.e., linkage disequilibrium (LD), among nearby genetic variants as well as redundant predictive signals across different eQTLs and eGenes (genes associated with eQTLs), thus allowing the model to leverage the full range of information captured by known eQTL associations. To achieve this, DSMs model the joint distribution over the genotype sequence and the gene expression profile by extending the popular Li-Stephens model of genetic

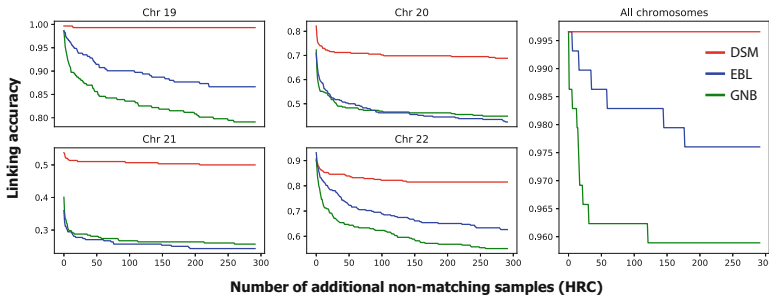


sequences to include probabilistic factors that capture the correlation between each eQTL and a set of known eGenes. Furthermore, we introduce a haplotype-based approximate inference for our model to allow the use of large reference panels needed for accurate prediction.

### 3 Results

To evaluate DSMs, we emulated a linking attack scenario in which the attacker identifies eQTL associations and trains the model on one dataset (GTEx;  $n = 588$ ) and uses the model to perform a linking attack on a separate dataset including gene expression profiles and genotypes (FUSION;  $n = 292$ ), shuffled to obscure the links. We also included genotype profiles from a third dataset (HRC) to increase the number of candidate profiles to distinguish from the match.

We compared DSM against two methods: GNB [1] and EBL [2]. GNB uses an independent Bayesian model with a Gaussian distribution over gene expression for each eQTL. EBL additionally leverages an extremity-based correction to set SNP prediction probabilities to 1 for extreme values of gene expression. The DSM consistently obtains higher linking accuracy, revealing a greater risk of re-identification than previously known (Fig. 1). The largest improvement was seen in chromosome 21, where the DSM correctly links 146 (out of 292) individuals, while GNB and EBL respectively link 75 and 71 individuals. Our work provides a new framework for understanding the privacy risks of functional genomic data in relation to protected genetic information.



**Fig. 1. DSMs enhance the accuracy of linking gene expression profiles to corresponding genotype profiles.** DSM’s linking accuracy is compared to that of prior approaches (EBL and GNB) as more non-matching target genotypes are considered.

**Acknowledgements.** This work is supported by the Hertz Fellowship and NSF GRFP under grant number 2141064 (to S.S.); NIH R01 HG010959 (to B.B.); and NIH DP5 OD029574 and RM1 HG011558 (to H.C.).

## References

1. Schadt, E.E., Woo, S., Hao, K.: Bayesian method to predict individual SNP genotypes from gene expression data. *Nat. Genet.* **44**(5), 603–608 (2012)
2. Harmanci, A., Gerstein, M.: Quantification of private information leakage from phenotype-genotype data: linking attacks. *Nat. Methods* **13**(3), 251–256 (2016)

# Author Index

## A

Abhari, Niloufar 230  
Ahmed, Omar 214  
Aldape, Kenneth 219  
Atıl, Berk 262

## B

Bağ, Alperen 262  
Baker, Daniel N. 227  
Bar-Joseph, Ziv 251  
Baral, Prabin 259  
Barsbey, Melih 262  
Berger, Bonnie 197, 257, 277  
Bhadra, Sahely 104  
Boucher, Christina 214  
Burchard, Esteban 239

## C

Chan, Michelle 229  
Chandra, Ghanshyam 58  
Chapagain, Prem 259  
Charvel, Eduardo 236  
Chen, Hao 251  
Chen, Yiling Elaine 222  
Chen, Ziqi 174  
Cheng, Chao 174  
Chikhi, Rayan 197  
Cho, Hyunghoon 277  
Chu, Athena H. Y. 139  
Clancy, Trevor 174  
Colijn, Caroline 230  
Cracco, Andrea 208  
Crawford, David R. 219

## D

de Wiel, Mark A van 120  
Dilthey, Alexander T. 241  
Dorman, Karin 254  
Du, Bing-Xue 85  
Durbin, Richard 244

## E

Ekim, Barış 197, 224  
El-Kebir, Mohammed 271  
Eskin, Eleazar 247

## F

Fan, Jason 21  
Flint, Jonathan 239  
Fong, John H. C. 139  
Fridman, Daniel 277  
Fu, Boyang 247

## G

Gao, Xin 265  
Gopalan, Vishaka 219  
Gorla, Aditya 239  
Guo, Yucheng 265  
Guo, Hongyu 174

## H

Han, Yunheng 195  
Hera, Mahmudur Rahman 200  
Ho, Joshua W. K. 139  
Hu, Zhihang 265  
Huang, Yuanhua 139

## I

Ivanovic, Stefan 271

## J

Jain, Chirag 58  
Joudaki, Amir 233

## K

Kahles, André 233  
Kececioglu, John 155  
Khan, Jamshed 21  
King, Irwin 265  
Klau, Gunnar W. 241  
Koerkamp, Ragnar Groot 233

Koslicki, David 200  
 Kotlar, Lior 224  
 Krieger, Spencer 155

**L**

Langmead, Ben 214, 227  
 LaPierre, Nathan 247  
 Li, Dongshunyi 251  
 Li, Jingyi Jessica 222  
 Li, Xuan Cindy 219  
 Li, Yu 265  
 Li, Zhiqiang 74  
 Lin, Jiacheng 268  
 Lin, Yu 3  
 Liu, Xinhao 210  
 Liu, Yuelin 219  
 Loog, Marco 120  
 Luo, Runpeng 3

**M**

Ma, Jianzhu 268  
 Mai, Uyen 236  
 Maier, Benjamin Dominik 203  
 Malikić, Salem 219  
 Mallick, Shikha 104  
 Malin, Bradley A. 274  
 McWhite, Claire 217  
 Medvedev, Paul 197  
 Mehrabadi, Farid Rashidi 219  
 Meterez, Alexandru 233  
 Min, Martin Renqiang 174  
 Mirarab, Siavash 236  
 Molloy, Erin K. 195, 219  
 Mount, Stephen M. 219  
 Mourragui, Soufiane M. C. 120  
 Mudide, Anish 257  
 Mustafa, Harun 233

**N**

Narasimhan, Giri 259  
 Naseri, Ardalan 212  
 Ning, Xia 174

**O**

Orenstein, Yaron 224  
 Özçelik, Rıza 262  
 Özgür, Arzucan 262  
 Ozkirimli, Elif 262

**P**

Patro, Rob 21  
 Pellow, David 224  
 Pibiri, Giulio Ermanno 21  
 Pierce-Ward, N. Tessa 200  
 Pratt, Drew 219  
 Pu, Lianrong 224

**R**

Rahmani, Elior 239  
 Raphael, Benjamin J. 210, 229  
 Rättsch, Gunnar 233  
 Reinders, Marcel J. T. 120  
 Roch, Sébastien 41  
 Rossi, Massimiliano 214  
 Ruppin, Eytan 219

**S**

Sadhuka, Shuvom 277  
 Sahinalp, S. Cenk 219  
 Sahlin, Kristoffer 197, 203  
 Sanaullah, Ahsan 249  
 Sankaraman, Sriram 239, 247  
 Sashittal, Palash 229  
 Schäffer, Alejandro A. 219  
 Schmidt, Henri 229  
 Schulte, Sara C. 241  
 Schweiger, Regev 244  
 Shamir, Ron 224  
 Shaw, Jim 205  
 Shi, Jian-Yu 85  
 Shirali, Azam 259  
 Singh, Mona 217  
 Singh, Rohit 257  
 Song, Le 265  
 Stebliankin, Vitalii 259

**T**

Tabatabaee, Yasamin 41  
 Tomescu, Alexandru I. 208  
 Tong, Tong 74  
 Turnbull, Steven 247  
 Tuteja, Geetu 254

**V**

van Nee, Mirrelijjn 120  
 Venkatesaramani, Rajagopal 274

Vorobeychik, Yevgeniy 274  
Vu, Ha T. H. 254

**W**

Wan, Yuk Kei 139  
Wan, Zhiyu 274  
Wang, Sheng 268  
Wang, Taifeng 265  
Warnow, Tandy 41  
Wessels, Lodewyk F. A. 120  
Woicik, Addie 268  
Wong, Alan S. L. 139  
Wu, Alexander P. 257  
Wu, Yufeng 230

**X**

Xu, Hanwen 268  
Xu, Yi 85

**Y**

Yiu, Siu-Ming 85  
Yu, Hui 85  
Yu, Qinze 265  
Yu, Yun William 205  
Yue, William 212

**Z**

Zaitlen, Noah 239  
Zeira, Ron 210  
Zhang, Chihao 222  
Zhang, Louxin 230  
Zhang, Shaojie 212, 249  
Zhang, Shihua 222  
Zhang, Yudi 254  
Zhi, Degui 249  
Zheng, Weizhong 139  
Zhi, Degui 212  
Zhou, Xiaogen 74