

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

APPROX/RANDOM 2022, September 19–21, 2022,
University of Illinois, Urbana-Champaign, USA (Virtual
Conference)

Edited by

Amit Chakrabarti
Chaitanya Swamy



Editors

Amit Chakrabarti 

Dartmouth College, Hanover, NH, USA
amit.chakrabarti@dartmouth.edu

Chaitanya Swamy 

University of Waterloo, Canada
cswamy@uwaterloo.ca

ACM Classification 2012

Theory of computation

ISBN 978-3-95977-249-5

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-249-5>.

Publication date

September, 2022

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):
<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.APPROX/RANDOM.2022.0

ISBN 978-3-95977-249-5

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

LIPICs – Leibniz International Proceedings in Informatics

LIPICs is a series of high-quality conference proceedings across all fields in informatics. LIPICs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Luca Aceto (*Chair*, Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Mikolaj Bojanczyk (University of Warsaw, PL)
- Roberto Di Cosmo (Inria and Université de Paris, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University - Brno, CZ)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (University of Oxford, GB)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

■ Contents


Preface	
<i>Amit Chakrabarti and Chaitanya Swamy</i>	0:ix
Program Committees	
.....	0:xi
Subreviewers	
.....	0:xiii–0:xiv
Authors	
.....	0:xv–0:xix

RANDOM

A Unified Approach to Discrepancy Minimization	
<i>Nikhil Bansal, Aditi Laddha, and Santosh Vempala</i>	1:1–1:22
Fourier Growth of Regular Branching Programs	
<i>Chin Ho Lee, Edward Pyne, and Salil Vadhan</i>	2:1–2:21
Double Balanced Sets in High Dimensional Expanders	
<i>Tali Kaufman and David Mass</i>	3:1–3:17
Fast and Perfect Sampling of Subgraphs and Polymer Systems	
<i>Antonio Blanca, Sarah Cannon, and Will Perkins</i>	4:1–4:18
High Dimensional Expansion Implies Amplified Local Testability	
<i>Tali Kaufman and Izhar Oppenheim</i>	5:1–5:10
Polynomial Bounds on Parallel Repetition for All 3-Player Games with Binary Inputs	
<i>Uma Girish, Kunal Mittal, Ran Raz, and Wei Zhan</i>	6:1–6:17
Local Treewidth of Random and Noisy Graphs with Applications to Stopping Contagion in Networks	
<i>Hermish Mehta and Daniel Reichman</i>	7:1–7:17
Beyond Single-Deletion Correcting Codes: Substitutions and Transpositions	
<i>Ryan Gabrys, Venkatesan Guruswami, João Ribeiro, and Ke Wu</i>	8:1–8:17
Affine Extractors and AC0-Parity	
<i>Xuanguo Huang, Peter Ivanov, and Emanuele Viola</i>	9:1–9:14
Hyperbolic Concentration, Anti-Concentration, and Discrepancy	
<i>Zhao Song and Ruizhe Zhang</i>	10:1–10:19
Improved Local Testing for Multiplicity Codes	
<i>Dan Karliner and Amnon Ta-Shma</i>	11:1–11:19
Unbalanced Expanders from Multiplicity Codes	
<i>Itay Kalev and Amnon Ta-Shma</i>	12:1–12:14

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy

 Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Streaming Algorithms with Large Approximation Factors <i>Yi Li, Honghao Lin, David P. Woodruff, and Yuheng Zhang</i>	13:1–13:23
Local Stochastic Algorithms for Alignment in Self-Organizing Particle Systems <i>Hridayesh Kedia, Shunhao Oh, and Dana Randall</i>	14:1–14:20
Tight Chernoff-Like Bounds Under Limited Independence <i>Maciej Skorski</i>	15:1–15:14
Eigenstripping, Spectral Decay, and Edge-Expansion on Posets <i>Jason Gaitonde, Max Hopkins, Tali Kaufman, Shachar Lovett, and Ruizhe Zhang</i>	16:1–16:24
Accelerating Polarization via Alphabet Extension <i>Iwan Duursma, Ryan Gabrys, Venkatesan Guruswami, Ting-Chun Lin, and Hsin-Po Wang</i>	17:1–17:15
Sketching Distances in Monotone Graph Classes <i>Louis Esperet, Nathaniel Harms, and Andrey Kupavskii</i>	18:1–18:23
Communication Complexity of Collision <i>Mika Göös and Siddhartha Jain</i>	19:1–19:9
Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness <i>Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang</i>	20:1–20:21
Learning Generalized Depth Three Arithmetic Circuits in the Non-Degenerate Case <i>Vishwas Bhargava, Ankit Garg, Neeraj Kayal, and Chandan Saha</i>	21:1–21:22
Lower Bound Methods for Sign-Rank and Their Limitations <i>Hamed Hatami, Pooya Hatami, William Pires, Ran Tao, and Rosie Zhao</i>	22:1–22:24
Black-Box Identity Testing of Noncommutative Rational Formulas of Inversion Height Two in Deterministic Quasipolynomial Time <i>V. Arvind, Abhranil Chatterjee, and Partha Mukhopadhyay</i>	23:1–23:22
Sampling from Potts on Random Graphs of Unbounded Degree via Random-Cluster Dynamics <i>Antonio Blanca and Reza Gheissari</i>	24:1–24:15
Improved Bounds for Randomly Colouring Simple Hypergraphs <i>Weiming Feng, Heng Guo, and Jiaheng Wang</i>	25:1–25:17
Lifting with Inner Functions of Polynomial Discrepancy <i>Yahel Manor and Or Meir</i>	26:1–26:17
Exploring the Gap Between Tolerant and Non-Tolerant Distribution Testing <i>Sourav Chakraborty, Eldar Fischer, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen</i>	27:1–27:23
A Sublinear Local Access Implementation for the Chinese Restaurant Process <i>Peter Mörters, Christian Sohler, and Stefan Walzer</i>	28:1–28:18
A Fully Adaptive Strategy for Hamiltonian Cycles in the Semi-Random Graph Process <i>Pu Gao, Calum MacRury, and Paweł Prałat</i>	29:1–29:22

Cover and Hitting Times of Hyperbolic Random Graphs <i>Marcos Kiwi, Markus Schepers, and John Sylvester</i>	30:1–30:19
Adaptive Sketches for Robust Regression with Importance Sampling <i>Sepideh Mahabadi, David P. Woodruff, and Samson Zhou</i>	31:1–31:21

APPROX

Finding the KT Partition of a Weighted Graph in Near-Linear Time <i>Simon Apers, Paweł Gawrychowski, and Troy Lee</i>	32:1–32:14
Maximum Matching Sans Maximal Matching: A New Approach for Finding Maximum Matchings in the Data Stream Model <i>Moran Feldman and Ariel Szarf</i>	33:1–33:24
Ordered k -Median with Outliers <i>Shichuan Deng and Qianfan Zhang</i>	34:1–34:22
Sketching Approximability of (Weak) Monarchy Predicates <i>Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrabi, Madhu Sudan, and Santhoshini Velusamy</i>	35:1–35:17
Integrality Gap of Time-Indexed Linear Programming Relaxation for Coflow Scheduling <i>Takuro Fukunaga</i>	36:1–36:13
Fair Correlation Clustering in General Graphs <i>Roy Schwartz and Roded Zats</i>	37:1–37:19
On Sketching Approximations for Symmetric Boolean CSPs <i>Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy</i>	38:1–38:23
Massively Parallel Algorithms for Small Subgraph Counting <i>Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Ronitt Rubinfeld, and Slobodan Mitrović</i>	39:1–39:28
Hardness Results for Weaver’s Discrepancy Problem <i>Daniel A. Spielman and Peng Zhang</i>	40:1–40:14
Relative Survivable Network Design <i>Michael Dinitz, Ama Koranteng, and Guy Kortsarz</i>	41:1–41:19
Bypassing the XOR Trick: Stronger Certificates for Hypergraph Clique Number <i>Venkatesan Guruswami, Pravesh K. Kothari, and Peter Manohar</i>	42:1–42:7
Approximating CSPs with Outliers <i>Suprovat Ghoshal and Anand Louis</i>	43:1–43:16
Submodular Dominance and Applications <i>Frederick Qiu and Sahil Singla</i>	44:1–44:21
Online Facility Location with Linear Delay <i>Marcin Bienkowski, Martin Böhm, Jarosław Byrka, and Jan Marcinkowski</i>	45:1–45:17

Prophet Matching in the Probe-Commit Model <i>Allan Borodin, Calum MacRury, and Akash Rakheja</i>	46:1–46:24
The Biased Homogeneous r -Lin Problem <i>Suprovat Ghoshal</i>	47:1–47:14
Asymptotically Optimal Bounds for Estimating H-Index in Sublinear Time with Applications to Subgraph Counting <i>Sepehr Assadi and Hoai-An Nguyen</i>	48:1–48:20
Maximizing a Submodular Function with Bounded Curvature Under an Unknown Knapsack Constraint <i>Max Klimm and Martin Knaack</i>	49:1–49:19
Some Results on Approximability of Minimum Sum Vertex Cover <i>Aleksa Stanković</i>	50:1–50:16
$(1 + \epsilon)$ -Approximate Shortest Paths in Dynamic Streams <i>Michael Elkin and Chhaya Trehan</i>	51:1–51:23
Caching with Reserves <i>Sharat Ibrahimpur, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang</i>	52:1–52:16
Space Optimal Vertex Cover in Dynamic Streams <i>Kheeran K. Naidu and Vihan Shah</i>	53:1–53:15
Approximating LCS and Alignment Distance over Multiple Sequences <i>Debarati Das and Barna Saha</i>	54:1–54:21
A Primal-Dual Algorithm for Multicommodity Flows and Multicuts in Treewidth-2 Graphs <i>Tobias Friedrich, Davis Issac, Nikhil Kumar, Nadym Mallek, and Ziena Zeif</i>	55:1–55:18

Preface

This volume contains the papers presented at the 25th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2021) and the 26th International Conference on Randomization and Computation (RANDOM 2021), which due to COVID-19 were organized as parallel virtual conferences from September 19–21, 2022. APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally-difficult problems, and the 2022 edition was the 25th in the series. RANDOM is concerned with applications of randomness to computational and combinatorial problems, and the 2022 edition was the 26th in the series. Prior to 2003, APPROX took place in Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), and Rome (2002), while RANDOM took place in Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), and Harvard (2002). Since 2003, APPROX and RANDOM have been co-located, taking place in Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014), Princeton (2015), Paris (2016), Berkeley (2017), Princeton (2018), Boston (2019), and online (2020, 2021).

Topics of interest for APPROX and RANDOM are: approximation algorithms, hardness of approximation, small space, sub-linear time and streaming algorithms, online algorithms, approaches that go beyond worst case analysis, distributed and parallel approximation, embeddings and metric-space methods, mathematical-programming methods, spectral methods, combinatorial optimization, algorithmic game theory, mechanism design and economics, computational-geometry problems, approximate learning, design and analysis of randomized algorithms, randomized complexity theory, pseudorandomness and derandomization, random combinatorial structures, random walks/Markov chains, expander graphs and randomness extractors, probabilistic proof systems, random projections and embeddings, error-correcting codes, average-case analysis, smoothed analysis, property testing, and computational learning theory.

The volume contains 24 contributed papers, selected by the APPROX Program Committee out of 46 submissions, and 31 contributed papers, selected by the RANDOM Program Committee out of 60 submissions. We would like to thank all the authors who submitted papers, the members of the program committees, and the external reviewers. We are grateful for the guidance of the steering committees: Jarosław Byrka, Samir Khuller, Monaldo Mastrolili, Laura Sanità, László Végh, Virginia Vassilevska Williams, and David P. Williamson for APPROX, and Oded Goldreich, Raghu Meka, Cris Moore, Anup Rao, Omer Reingold, Dana Ron, Ronitt Rubinfeld, Amit Sahai, Ronen Shaltiel, Alistair Sinclair, and Paul Spirakis for RANDOM.

■ Program Committees

APPROX

Nikhil Bansal	University of Michigan
Deeparnab Chakrabarty	Dartmouth College
Parinya Chalermsook	Aalto University
Karthekeyan Chandrasekaran	UIUC
Moses Charikar	Stanford University
Zachary Friggstad	University of Alberta
Sungjin Im	UC, Merced
Thomas Kesselheim	University of Bonn
Ravishankar Krishnaswamy	Microsoft Research India
Pasin Manurangsi	Google Research
Neil Olver	London School of Economics and Political Science
Chaitanya Swamy (PC chair)	University of Waterloo
Vera Traub	ETH Zurich
Seeun William Umboh	The University of Sydney
Jan Vondrak	Stanford University
Rico Zenklusen	ETH Zurich

RANDOM

Sepehr Assadi	Rutgers University
Arnab Bhattacharyya	National University of Singapore
Andrej Bogdanov	The Chinese University of Hong Kong
Amit Chakrabarti (PC Chair)	Dartmouth College
Zongchen Chen	MIT
Talya Eden	Boston University and MIT
Thomas Hayes	The University of New Mexico
William Hoza	University of California, Berkeley
Fotis Iliopoulos	Google Research
Haim Kaplan	Tel Aviv University
Pravesh Kothari	Carnegie Mellon University
Reut Levi	Reichman University
Or Meir	University of Haifa
Dor Minzer	MIT
Debmalya Panigrahi	Duke University
Eric Price	The University of Texas at Austin
Cynthia Rush	Columbia University
Rahul Santhanam	Oxford University
Srikanth Srinivasan	Aarhus University
Justin Thaler	Georgetown University
Samson Zhou	Carnegie Mellon University



Subreviewers

APPROX

Jason Li
Jakab Tardos
Sepehr Assadi
Laszlo Kozma
Rudy Zhou
Krzysztof Sornat
Joachim Spoerhase
Euiwoong Lee
Prashanth Amireddy
Aaron Potechin
Chhaya Trehan
Richard Santiago
Lars Rohwedder
Kevin Sun
Dishant Goyal
Ioana Bercea
Joachim Spoerhase
Sai Surya
Ali Vakilian
Richard Santiago
Michael Kapralov
C. Seshadhri
Franziska Eberle
Syamantak Das
Ramin Mousavi
Melanie Schmidt
Konstantin Makarychev
Sepehr Assadi
Bundit Laekhanukit
Samuel Hopkins
Adam Kasperski
Ali Kemal Sinop
Charlie Carlson
Vijay Bhattiprolu
Hossein Jowhari
Soheil Behnezhad
Chris Schwiegelshohn
Jeck Lim
Pavel Valtr
Chandra Chekuri
Qingyun Chen
Franziska Eberle
Ruben Hoeksma
Alexander Braun

Mohammad Roghani
Amey Bhangale
Aleksa Stankovic
Madhur Tulsiani
Tijn de Vos
Taisuke Yasuda
Kamyar Khodamoradi
Yi Hao
C. Seshadhri
Adam Kurpisz
Yasushi Kawase
Euiwoong Lee
Marco Caoduro
Hung Le
Piotr Skowron
Ameet Gadekar
Prantar Ghosh
Rajesh Chitnis
Aviad Rubinstein
Chien-Chung Huang
Mathieu Mari
Clement Canonne

RANDOM

Tarun Kathuria
Eshan Chattopadhyay
Paul Goldberg
Argyrios Deligkas
Emmanouil Zampetakis
Siqi Liu
Vedat Levi Alev
Amartya Shankha Biswas
Ali Vakilian
Merav Parter
Xiaoyu Chen
Ewan Davies
Noga Ron-Zewi
Max Hopkins
Igor Shinkar
Yotam Dikstein
Nathaniel Harms
Erik Waingarten
Alec Sun
Catherine Greenhill
Guy Blanc

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

D. Ellis Hershkowitz
Sidhant Saraogi
Deeparnab Chakrabarty
Jan Hązła
Yong Gao
Rachit Nimavat
Nikhil Srivastava
Jonathan Leake
Praladh Harsha
Tali Kaufman-Halman
Chris Umans
Enric Boix-Adserà
Konstantinos Zampetakis
Janani Sundaresan
Erik Waingarten
Nithin Varma
Vishwas Bhargava
Paul Liu
Ewan Davies
Anthimos-Vardis Kandiros
Andreas Galanis
Varsha Dani
Madhur Tulsiani
Mitali Bafna
Gil Cohen
Taisuke Yasuda
Makrand Sinha
Igor Pak
Antonio Blanca
Min Ye
Jakab Tardos
Dmitry Itsykson
Marc Vinyals
Sandeep Silwal
Shyam Narayanan
Navid Ardeshtir
Oliver Korten
Sidhant Saraogi
Vishwas Bhargava
Ramprasad Saptharishi
Prerona Chatterjee
Pranjal Dutta
Ben Lee Volk
Ankit Garg
Weiming Feng
Eric Vigoda
Charilaos Efthymiou
Thuy Duong Vuong

Huy Tuan Pham
Aditya Potukuchi
Praladh Harsha
Parth Mittal
Sajin Koroth
Jakub Tětek
Clément Canonne
Gautam Kamath
Udi Wieder
Edward Pyne
Moti Medina
Yuval Dagan
Ainesh Bakshi
Lyuben Lichev
Omri Ben-Eliezer
Nikolaos Fountoulakis
Kyriakos Axiotis
Rong Ge
Advait Parulekar

■ List of Authors


Simon Apers (32)
CNRS and IRIF, Paris, France

V. Arvind (23)
The Institute of Mathematical Sciences, HBNI,
Chennai, India

Sepehr Assadi (48)
Department of Computer Science, Rutgers
University, Piscataway, NJ, USA

Nikhil Bansal (1)
University of Michigan, Ann Arbor, MI, USA

Vishwas Bhargava (21)
Department of Computer Science, Rutgers
University, Piscataway, NJ, USA


Marcin Bienkowski  (45)
Institute of Computer Science, University of
Wrocław, Poland


Amartya Shankha Biswas (39)
CSAIL, MIT, Cambridge, MA, USA

Antonio Blanca (4, 24)
Department of Computer Science and
Engineering, Pennsylvania State University,
University Park, PA, USA

Allan Borodin (46)
Department of Computer Science, University of
Toronto, Canada

Joanna Boyland (38)
Harvard College, Harvard University, Cambridge,
MA, USA

Jarosław Byrka  (45)
Institute of Computer Science, University of
Wrocław, Poland

Martin Böhm  (45)
Institute of Computer Science, University of
Wrocław, Poland

Sarah Cannon (4)
Department of Mathematical Sciences,
Claremont McKenna College, CA, USA

Sourav Chakraborty (27)
Indian Statistical Institute, Kolkata, India


Abhranil Chatterjee (23)
Indian Institute of Technology Bombay, India

Chi-Ning Chou (35)
School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA, USA


Debarati Das (54)
Pennsylvania State University, University Park,
PA, USA


Shichuan Deng (34)
IIIS, Tsinghua University, Beijing, China


Michael Dinitz (41)
Johns Hopkins University, Baltimore, MD, USA


Iwan Duursma  (17)
University of Illinois Urbana-Champaign, IL,
USA

Talya Eden (39)
CSAIL, MIT, Cambridge MA, USA; Boston
University, MA, USA


Michael Elkin  (51)
Ben-Gurion University of the Negev, Beer-Sheva,
Israel


Louis Esperet  (18)
Univ. Grenoble Alpes, CNRS, Laboratoire
G-SCOP, Grenoble, France


Moran Feldman  (33)
Department of Computer Science, University of
Haifa, Israel

Weiming Feng  (25)
School of Informatics, University of Edinburgh,
UK

Eldar Fischer (27)
Technion - Israel Institute of Technology, Haifa,
Israel

Tobias Friedrich  (55)
Hasso Plattner Institute, Universität Potsdam,
Germany

Takuro Fukunaga  (36)
Faculty of Science and Engineering, Chuo
University, Tokyo, Japan

Ryan Gabrys  (8, 17)
ECE Department, University of California, San
Diego, CA, USA

Jason Gaitonde (16)
Cornell University, Ithaca, NY, USA

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Pu Gao (29)
Department of Combinatorics and Optimization,
University of Waterloo, Canada
- Ankit Garg (21)
Microsoft Research, Bangalore, India
- Paweł Gawrychowski (32)
Institute of Computer Science, University of
Wrocław, Poland
- Reza Gheissari (24)
Department of Statistics and EECS, University
of California, Berkeley, CA, USA
- Arijit Ghosh (27)
Indian Statistical Institute, Kolkata, India
- Suprovat Ghoshal (43, 47)
University of Michigan, Ann Arbor, MI, USA
- Uma Girish (6)
Princeton University, NJ, USA
- Alexander Golovnev (35)
Department of Computer Science, Georgetown
University, Washington, D.C., USA
- Heng Guo  (25)
School of Informatics, University of Edinburgh,
UK
- Venkatesan Guruswami  (8, 17, 20, 42)
EECS Department, University of California,
Berkeley, CA, USA
- Mika Göös (19)
EPFL, Lausanne, Switzerland
- Nathaniel Harms  (18)
University of Waterloo, Canada
- Hamed Hatami (22)
School of Computer Science, McGill University,
Montreal, Canada
- Pooya Hatami  (22)
Department of Computer Science and
Engineering, The Ohio State University,
Columbus, OH, USA
- Max Hopkins (16)
University of California, San Diego, La Jolla,
CA, USA
- Xuangui Huang (9)
Northeastern University, Boston, MA, USA
- Michael Hwang (38)
Harvard College, Harvard University, Cambridge,
MA, USA
- Sharat Ibrahimpur (52)
University of Waterloo, Canada
- Davis Issac  (55)
Hasso Plattner Institute, Universität Potsdam,
Germany
- Peter Ivanov (9)
Northeastern University, Boston, MA, USA
- Siddhartha Jain  (19)
EPFL, Lausanne, Switzerland
- Itay Kalem (12)
Department of Computer Science, Tel Aviv
University, Israel
- Dan Karliner (11)
Department of Computer Science, Tel Aviv
University, Israel
- Tali Kaufman (3, 5, 16)
Department of Computer Science, Bar-Ilan
University, Ramat-Gan, Israel
- Neeraj Kayal (21)
Microsoft Research, Bangalore, India
- Hriddesh Kedia (14)
Georgia Institute of Technology, Atlanta, GA,
USA
- Marcos Kiwi  (30)
Department of Industrial Engineering and
Center for Mathematical Modeling, Universidad
de Chile, Santiago, Chile
- Max Klimm (49)
Institute for Mathematics, Technische
Universität Berlin, Germany
- Martin Knaack (49)
Institute for Mathematics, Technische
Universität Berlin, Germany
- Ama Koranteng (41)
Johns Hopkins University, Baltimore, MD, USA
- Guy Kortsarz (41)
Rutgers University, Camden, NJ, USA
- Pravesh K. Kothari (42)
Carnegie Mellon University, Pittsburgh, PA,
USA
- Nikhil Kumar  (55)
Hasso Plattner Institute, Universität Potsdam,
Germany


- Andrey Kupavskii (18)
Univ. Grenoble Alpes, CNRS, Laboratoire
G-SCOP, Grenoble, France; Moscow Institute of
Physics and Technology, Russia; Huawei R&D
Moscow, Russia
- Aditi Laddha (1)
Georgia Tech, Atlanta, GA, USA
- Chin Ho Lee (2)
Harvard University, Cambridge, MA, USA
- Troy Lee (32)
Centre for Quantum Software and Information,
University of Technology Sydney, Australia
- Yi Li  (13)
Division of Mathematical Sciences, Nanyang
Technological University, Singapore
- Honghao Lin (13)
Computer Science Department, Carnegie Mellon
University, Pittsburgh, PA, USA
- Ting-Chun Lin  (17)
University of California San Diego, CA, USA;
Hon Hai (Foxconn) Research Institute, Taipei,
Taiwan
- Quanquan C. Liu (39)
Northwestern University, Evanston, IL, USA
- Anand Louis (43)
Indian Institute of Science, Bangalore, India
- Shachar Lovett (16)
University of California, San Diego, La Jolla,
CA, USA
- Xin Lyu (20)
University of California Berkeley, CA, USA
- Calum MacRury (29, 46)
Department of Computer Science, University of
Toronto, Canada
- Sepideh Mahabadi (31)
Microsoft Research, Redmond, WA, USA
- Nadym Mallek  (55)
Hasso Plattner Institute, Universität Potsdam,
Germany
- Peter Manohar (42)
Carnegie Mellon University, Pittsburgh, PA,
USA
- Yahel Manor (26)
Department of Computer Science, University of
Haifa, Israel
- Jan Marcinkowski  (45)
Institute of Computer Science, University of
Wrocław, Poland
- David Mass (3)
Department of Computer Science, Bar-Ilan
University, Ramat-Gan, Israel
- Hermish Mehta (7)
Citadel Securities, Chicago, IL, USA
- Or Meir  (26)
Department of Computer Science, University of
Haifa, Israel
- Gopinath Mishra (27)
University of Warwick, Coventry, UK
- Slobodan Mitrović (39)
University of California Davis, CA, USA
- Kunal Mittal (6)
Princeton University, NJ, USA
- Partha Mukhopadhyay (23)
Chennai Mathematical Institute, India
- Peter Mörters  (28)
Universität zu Köln, Germany
- Kheeran K. Naidu  (53)
Department of Computer Science, University of
Bristol, UK
- Hoai-An Nguyen (48)
Department of Computer Science, Rutgers
University, Piscataway, NJ, USA
- Shunhao Oh (14)
Georgia Institute of Technology, Atlanta, GA,
USA
- Izhar Oppenheim (5)
Department of Mathematics, Ben-Gurion
University of the Negev, Be'er-Sheva, Israel
- Will Perkins (4)
Department of Computer Science, Georgia
Institute of Technology, Atlanta, GA, USA
- William Pires (22)
School of Computer Science, McGill University,
Montreal, Canada
- Tarun Prasad  (38)
Harvard College, Harvard University, Cambridge,
MA, USA
- Paweł Prałat (29)
Department of Mathematics, Toronto
Metropolitan University, Canada

Manish Purohit (52)
Google Research, Mountain View, CA, USA

Edward Pyne (2)
Harvard University, Cambridge, MA, USA


Frederick Qiu (44)
Department of Computer Science, Princeton University, NJ, USA

Akash Rakheja (46)
Department of Computer Science, University of Toronto, Canada

Dana Randall  (14)
Georgia Institute of Technology, Atlanta, GA, USA

Ran Raz (6)
Princeton University, NJ, USA


Daniel Reichman (7)
Department of Computer Science, Worcester Polytechnic Institute, MA, USA

João Ribeiro  (8)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Ronitt Rubinfeld (39)
CSAIL, MIT, Cambridge, MA, USA

Barna Saha (54)
University of California, San Diego, CA, USA

Chandan Saha (21)
Indian Institute of Science, Bangalore, India


Markus Schepers  (30)
Institut für Medizinische Biometrie,
Epidemiologie und Informatik,
Johannes-Gutenberg-University Mainz, Germany

Roy Schwartz (37)
The Henry and Marilyn Taub Faculty of
Computer Science, Technion, Haifa, Israel


Sayantan Sen (27)
Indian Statistical Institute, Kolkata, India


Vihan Shah (53)
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Amirbehshad Shahrasbi (35)
Microsoft, Redmond, WA, USA

Noah Singer  (38)
Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA;
Harvard College, Harvard University, Cambridge, MA, USA


Sahil Singla (44)
School of Computer Science, Georgia Tech, Atlanta, GA, USA

Maciej Skorski  (15)
University of Luxembourg, Luxembourg

Christian Sohler  (28)
Universität zu Köln, Germany


Zhao Song (10)
Adobe Research, Seattle, WA, USA

Daniel A. Spielman (40)
Yale University, New Haven, CT, USA

Aleksa Stanković  (50)
Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

Madhu Sudan (35)
School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA USA


Zoya Svitkina (52)
Google Research, Mountain View, CA, USA

John Sylvester  (30)
School of Computing Science, University of Glasgow, UK

Ariel Szarf (33)
Department of Mathematics and Computer Science, Open University of Israel, Ra'anana, Israel

Amnon Ta-Shma (11, 12)
Department of Computer Science, Tel Aviv University, Israel

Ran Tao (22)
Department of Mathematics and Statistics,
McGill University, Montreal, Canada

Chhaya Trehan  (51)
London School of Economics & Political Science, UK

Salil Vadhan (2)
Harvard University, Cambridge, MA, USA


Erik Vee (52)
Google Research, Mountain View, CA, USA

Santhoshini Velusamy (35, 38)
School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA, USA


Samson Zhou  (31)
Carnegie Mellon University, Pittsburgh, PA,
USA

Santosh Vempala (1)
Georgia Tech, Atlanta, GA, USA

Emanuele Viola (9)
Northeastern University, Boston, MA, USA

Stefan Walzer  (28)
Universität zu Köln, Germany


Hsin-Po Wang  (17)
University of California San Diego, CA, USA

Jiaheng Wang  (25)
School of Informatics, University of Edinburgh,
UK


Joshua R. Wang (52)
Google Research, Mountain View, CA, USA

Xiuhan Wang (20)
Tsinghua University, Beijing, China

David P. Woodruff (13, 31)
Computer Science Department, Carnegie Mellon
University, Pittsburgh, PA, USA

Ke Wu  (8)
Computer Science Department, Carnegie Mellon
University, Pittsburgh, PA, USA

Roded Zats (37)
The Henry and Marilyn Taub Faculty of
Computer Science, Technion, Haifa, Israel

Ziena Zeif  (55)
Hasso Plattner Institute, Universität Potsdam,
Germany

Wei Zhan (6)
Princeton University, NJ, USA

Peng Zhang (40)
Rutgers University, Piscataway, NJ, USA

Qianfan Zhang (34)
IIIS, Tsinghua University, Beijing, China

Ruizhe Zhang (10, 16)
The University of Texas at Austin, TX, USA

Yuheng Zhang (13)
Zhiyuan College, Shanghai Jiao Tong University,
China

Rosie Zhao (22)
School of Computer Science, McGill University,
Montreal, Canada

A Unified Approach to Discrepancy Minimization

Nikhil Bansal ✉

University of Michigan, Ann Arbor, MI, USA

Aditi Laddha ✉

Georgia Tech, Atlanta, GA, USA

Santosh Vempala ✉

Georgia Tech, Atlanta, GA, USA

Abstract

We study a unified approach and algorithm for constructive discrepancy minimization based on a stochastic process. By varying the parameters of the process, one can recover various state-of-the-art results. We demonstrate the flexibility of the method by deriving a discrepancy bound for smoothed instances, which interpolates between known bounds for worst-case and random instances.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Discrepancy theory, smoothed analysis

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2022.1

Category RANDOM

Related Version Full Version: <https://arxiv.org/abs/2205.01023>

Funding *Nikhil Bansal*: Supported in part by the NWO VICI grant 639.023.812.

Aditi Laddha: Supported in part by NSF awards CCF-2007443 and CCF-2134105.

Santosh Vempala: Supported in part by NSF awards CCF-2007443 and CCF-2134105.

Acknowledgements We are grateful to Yin Tat Lee and Mohit Singh for helpful discussions.

1 Introduction

Given a universe of elements $U = \{1, \dots, n\}$ and a collection $\mathcal{S} = \{S_1, \dots, S_m\}$ of subsets $S_i \subseteq U$, the discrepancy of the set system \mathcal{S} is defined as

$$\text{disc}(\mathcal{S}) = \min_{x: U \rightarrow \{-1, 1\}} \max_{i \in [m]} \left| \sum_{j \in S_i} x(j) \right|.$$

That is, the discrepancy is the minimum imbalance that must occur in at least one of the sets in \mathcal{S} over all bipartitions of U . More generally for an $m \times n$ matrix A , the discrepancy of A is defined as $\text{disc}(A) = \min_{x \in \{-1, 1\}^n} \|Ax\|_\infty$. Note that the definition for set systems corresponds to choosing A as the incidence matrix of \mathcal{S} , i.e., $A_{ij} = 1$ if $j \in S_i$ and 0 otherwise. Discrepancy is a well-studied area with several applications in both mathematics and theoretical computer science (see [14, 17, 28]).

Spencer’s problem. In a celebrated result, Spencer [34] showed that the discrepancy of any set system with $m = n$ sets is $O(\sqrt{n})$, and more generally $O(\sqrt{n \log(2m/n)})$ for $m \geq n$. To show this, he developed a general partial-coloring method (a.k.a. the entropy method), building on a counting argument of Beck [13], that has since been used widely for various other problems. A similar approach was developed independently by Gluskin [20]. Roughly, here the elements are colored in $O(\log n)$ phases. In each phase, an $\Omega(1)$ fraction of the elements get colored while incurring a small discrepancy for each row.



© Nikhil Bansal, Aditi Laddha, and Santosh Vempala;
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 1; pp. 1:1–1:22



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Beck-Fiala and Komlós problems. Another central question is the Beck-Fiala problem where each element appears in at most k sets in \mathcal{S} . Equivalently, every column of the incidence matrix is k -sparse. The long-standing Beck-Fiala conjecture [15] states that $\text{disc}(\mathcal{S}) = O(\sqrt{k})$. A further generalization is the Komlós problem, also called the vector balancing problem, about the discrepancy of matrices A with column ℓ_2 -norms at most 1. Komlós conjectured that $\text{disc}(A) = O(1)$ for any such matrix. Note that the Komlós conjecture implies the Beck-Fiala conjecture.

Banaszczyk showed an $O(\sqrt{\log n})$ bound for the Komlós problem based on a deep geometric result [3]. Here, the full coloring is constructed directly (in a single phase), and this result has also found several applications. The resulting $O(\sqrt{k \log n})$ bound for the Beck-Fiala problem is also the best known bound for general k .¹

In contrast, the partial coloring method only gives weaker bounds of $O(\log n)$ and $O(k^{1/2} \log n)$ for these problems – the $O(\log n)$ loss is incurred due to the $O(\log n)$ phases of partial coloring.

Limitations of Banaszczyk’s result. Even though Banaszczyk’s method gives better bounds for the Komlós problem, it is not necessarily stronger, and is incomparable to the partial coloring method. E.g., it is not known how to obtain Spencer’s $O(\sqrt{n})$ result (or anything better than the trivial $O(\sqrt{n \log n})$ random-coloring bound) using Banaszczyk’s result. A very interesting question is whether there is a common generalization that unifies both these results and techniques.

Algorithmic approaches. Both the partial coloring method and Banaszczyk’s result were originally non-algorithmic, and a lot of recent progress has resulted in their algorithmic versions. Starting with the work of [4], several different algorithmic approaches are now known for the partial coloring method [27, 33, 21, 18], based on various elegant ideas from linear algebra, random walks, optimization and convex geometry.

In further progress, an algorithmic version of the $O(\sqrt{\log n})$ bound for the Komlós problem was obtained by [5], see also [7], and [6] for the more general algorithmic version of Banaszczyk’s result. In related work, Levy et al. [26] gave deterministic polynomial time constructive algorithms for the Spencer and Komlós settings matching $O(\sqrt{n \log(2m/n)})$ and $O(\sqrt{\log n})$ respectively.

A key underlying idea behind many of these results is to perform a discrete Brownian motion (random walk with small steps) in the $\{-1, 1\}^n$ cube, where the update steps are correlated and chosen to lie in some suitable subspace. However, the way in which these subspaces are chosen for the partial coloring method and the Komlós problem are quite different. We give a high level description of these approaches as this will be crucial later on.

In the partial coloring approach, the walk is performed in a subspace orthogonal to the *tight discrepancy constraints*. If the discrepancy for some row A_i reaches its target discrepancy bound, the update Δx to the coloring satisfies $A_i \cdot \Delta x = 0$. As the walk continues over time, the subspace dimension gets smaller and smaller until the walk is stuck. At this point, the subspace is reset and the *next phase* resumes.

On the other hand, the algorithm for the Komlós problem does not consider the discrepancy constraints at all, and chooses a different subspace with a certain sub-isotropic property which ensures the discrepancy incurred for a row is roughly proportional to its ℓ_2 norm,

¹ For $k = o(\log n)$ an improved bound follows from the $2k - 1$ bound by [15].

while ensuring that the rows with large ℓ_2 -norm incur zero-discrepancy. In particular, in contrast to the partial coloring method, all the elements are colored in a *single phase*, and the discrepancy constraints are ignored.

The need for a combined approach. Even though the $O(\sqrt{k \log n})$ bound for the general Beck-Fiala problem is based on Banaszczyk’s method, all the important special cases where the conjectured $O(\sqrt{k})$ bound holds are based on the partial coloring method. For example, Spencer’s problem with $m = O(n)$ sets corresponds to special case of the Beck-Fiala problem with $k = O(n)$. So Spencer’s six-deviations result resolves the Beck-Fiala conjecture for this case, which we do not know how to obtain from Banaszczyk’s result.

The Beck-Fiala conjecture also holds for the case of random set systems with $m \geq n$. In particular, Potukuchi [32] considers the model where each column has 1’s in k randomly chosen rows and shows that the discrepancy is $O(\sqrt{k})$ with high probability. See also [19, 9, 22, 1] for related results. Potukuchi’s result crucially relies on the partial coloring approach, and it is not clear at all how to exploit the properties of random instances in Banaszczyk’s approach.

Thus a natural question and a first step towards resolving the Beck-Fiala and Komlós conjecture, and making progress on other discrepancy problems, is whether there exist more general techniques to obtain both Spencer’s and Potukuchi’s result and the $O(\sqrt{k \log n})$ bound for the Beck-Fiala problem in a unified way.

1.1 Our results

We present a new unified framework that recovers all the results mentioned above, and various other state-of-the-art results as special cases. Our algorithm is based on a derandomization of a stochastic process that is guided by a barrier-based potential function. We were inspired by an elegant idea of Lee and Singh [23] who showed how the barrier function approach can be used to give a proof of Spencer’s result without any partial coloring phases. A related idea was also explored in [21]. The barrier function approach itself has been used extensively in various settings such as graph and matrix sparsification [12, 24], covariance estimation [35], isoperimetric inequalities [25], bandit algorithms [2] and also in the context of discrepancy minimization [10, 21, 11].

Given a matrix A , the algorithm starts with the all-zero coloring x_0 . Let $x_t \in [-1, 1]^n$ be the coloring at time t . The algorithm maintains a barrier $b_t > 0$ over time and defines the slack of row i at time t as

$$s_i(t) = b_t - \underbrace{\sum_{j=1}^n a_i(j)x_t(j)}_{\text{current discrepancy}} - \lambda \underbrace{\sum_{j=1}^n a_i(j)^2(1 - x_t(j)^2)}_{\text{remaining variance}}. \quad (1)$$

Notice that when all $x_t(j)$ eventually reach ± 1 , the *remaining variance* term is zero and the slack measures the gap between the discrepancy and the barrier.

We define the potential

$$\Phi(t) = \sum_i s_i(t)^{-p} \quad (2)$$

for some fixed $p > 1$, that penalizes the rows with small slacks and blows up to infinity if some slack approaches zero. If we can ensure that the slacks are always positive and the potential is bounded, then the discrepancy is upper bounded by value of the barrier when the algorithm terminates.

At each time step, the algorithm picks a random direction v_t that is orthogonal to some of the rows with the least slack, and satisfies some additional properties, and updates the coloring by a small amount in the direction v_t . The barrier b_t is also updated. These updates are chosen to ensure that the potential does not increase in expectation, and hence all the slacks stay bounded away from 0. We give a more detailed overview in Section 2.

By changing the parameters p, λ depending on the problem at hand, we obtain several results using a unified approach.

1. Set coloring [34]. For any set system on n elements and $m \geq n$ sets, $\text{disc}(\mathcal{S}) = O(\sqrt{n \log(2m/n)})$.
2. Komlós problem [7]. For any $A \in \mathbb{R}^{m \times n}$ with columns norms $\|A^j\|_2 \leq 1$, $\text{disc}(A) = O(\sqrt{\log n})$.
3. Random/Spectral Hypergraphs [32]. Let $A \in \{0, 1\}^{m \times n}$ be the incidence matrix of a set system with n elements and m sets, where element lies in at most k sets and let $\gamma = \max_{v \perp \mathbf{1}, \|v\|=1} \|Av\|$. Then for $m \geq n$, $\text{disc}(\mathcal{S}) = O(\sqrt{k} + \gamma)$.
4. Gaussian Matrix [16]. For a random matrix $A \in \mathbb{R}^{m \times n}$ with each entry $A_{ij} \sim \mathcal{N}(0, \sigma^2)$ independently, with probability at least $1 - (1/m^3)$, $\text{disc}(A) = O\left(\sigma(\sqrt{n} + \sqrt{\log m}) \cdot \sqrt{\log \frac{2m}{n}}\right)$.

More generally, given a matrix A , we state the following result based on optimizing the various parameters of the algorithm, depending on the properties of A . This allows our framework to be applied in a black-box manner to a given problem at hand.

► **Theorem 1.** For a $A \in \mathbb{R}^{m \times n}$ with $\|A^j\|_2 \leq L$ and $|a_i(j)| \leq M$ for all $i \in [m], j \in [n]$, let $h : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be a non-increasing function such that for every subset $S \subseteq [n]$ and $i \in [m]$,

$$\sum_{j \in S} a_i(j)^2 \leq |S| \cdot h(|S|). \quad (3)$$

Then, for any $p > 1$, there exists a vector $x \in \{-1, 1\}^n$ such that $\|Ax\|_\infty \leq 5b_0 + 2M$, where

$$b_0 = \min \left(\sqrt{8(p+1)(48m)^{1/p} \cdot \beta}, 250L\sqrt{\log(2m)} \right). \quad (4)$$

where $\beta = \int_{t=0}^{n-2} h(n-t)(n-t)^{-1/p} dt$.

Let us see how Theorem 1 directly leads to the results stated above.

Set coloring. As $\|A^j\|_2 \leq \sqrt{m}$, we have $L = \sqrt{m}$, and as $\sum_{j \in S} a_i(j)^2 \leq |S|$, we can set $h(t) = 1$ for all $t \in [n]$. Consider (4) and suppose $p \geq 1.1$ so that $p/(p-1) = O(1)$. Then

$$\beta = \int_{t=0}^{n-2} h(n-t) \cdot (n-t)^{-1/p} dt = O(n^{1-1/p}),$$

and the first bound in (4) gives $b_0 = O(pn^{1/2}(m/n)^{1/p})$. Setting $p = \log(2m/n)$ gives Spencer's $O(\sqrt{n \log(2m/n)})$ bound.

Interestingly, the above result gives a new proof of Spencer's six-deviations result based on a direct single-phase coloring. In contrast, all the previously known proofs of this result [4, 27, 33, 18] required multiple partial coloring phases.

Komlós problem. Here $L = 1$ and the second term in (4) directly gives a $O(\sqrt{\log m})$ bound². This also implies an $O(\sqrt{\log n})$ bound as at most n^2 rows can have ℓ_1 -norm more than 1, and we can assume that $m \leq n^2$.

Similarly, bounding $h(t)$ using standard concentration bounds, directly gives the following results for various models of random matrices.

► **Theorem 2** (Sub-Gaussian Matrix). *Let $A \in \mathbb{R}^{m \times n}$ with each column drawn independently from a distribution \mathcal{D} , where the marginal of each coordinate is sub-Gaussian with mean 0 and variance σ^2 . Then, for $n \leq m \leq 2^{O(\sqrt{n})}$, $\text{disc}(A) = O(\sigma \sqrt{n \log(2m/n)})$, with probability at least $1 - (1/m^2)$.*

► **Theorem 3** (Random Matrix). *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$ such that every column of A is drawn independently from the uniform distribution on $\{x \in \mathbb{R}^m : \|x\|_2 \leq 1\}$. Then $\text{disc}(A) = O(1)$ with probability at least $1 - (1/m^2)$.*

1.1.1 Flexibility of the method

An important advantage of the method is its flexibility, which can be used to obtain several additional results.

Subadditivity. Given $A, B \in \mathbb{R}^{m \times n}$, can we bound $\text{disc}(A + B)$ given bounds on $\text{disc}(A)$ and $\text{disc}(B)$? Such questions can be directly handled by this framework by considering a weighted combination of two different potential functions – one for A and another for B .

More precisely, let us define $\text{sdisc}(A)$, the *Stochastic Discrepancy* of a matrix A , to be the upper bound on discrepancy obtained by the Potential Walk described in Algorithm 1. For this notion, we have the following approximate subadditivity for arbitrary matrices.

► **Theorem 4** (Subadditivity of Stochastic Discrepancy). *For any two arbitrary matrices $A, B \in \mathbb{R}^{m \times n}$, there exists $x \in \{-1, 1\}^n$ such that*

$$\begin{aligned} |\langle a_i, x \rangle| &\lesssim \text{sdisc}(A) \quad \text{for every row } a_i \text{ of } A, \text{ and} \\ |\langle b_i, x \rangle| &\lesssim \text{sdisc}(B) \quad \text{for every row } b_i \text{ of } B. \end{aligned}$$

In particular, this implies that $\text{sdisc}(A + B) \lesssim \text{sdisc}(A) + \text{sdisc}(B)$.

Here $a \lesssim b$ means that $a = O(1)b$. The theorem is algorithmic if A, B are given. It also implies that for any matrix A , we have $\text{sdisc}(A) \lesssim \min_B (\text{sdisc}(B) + \text{sdisc}(A - B))$.

Similar questions have been studied previously in the context of understanding the discrepancy of unions of systems [30, 31]. For example, other related quantities such as the γ_2 -norm and the determinant lower bound are also subadditive [30, 31]. We remark that the additive bound cannot hold for the (actual) discrepancy or even hereditary discrepancy³, and a logarithmic loss is necessary. For this reason, the previous additive bounds based on γ_2 -norm and the determinant lower bound lose extra polylogarithmic factors when translated to discrepancy.

A direct application of Theorem 4 is the following.

² It would be interesting to construct an explicit family of examples where the discrepancy obtained by our approach is $\Omega(\sqrt{\log n})$.

³ A classical example due to Hoffman gives two set systems A and B , each with hereditary discrepancy 1, but their union has discrepancy $\Omega(\log n / \log \log n)$ [29].

► **Theorem 5** (Semi-Random Komlós). *Let $C \in \mathbb{R}^{m \times n}$ be an arbitrary matrix with columns satisfying $\|C^j\|_2 \leq 1$ for all $j \in [n]$, and $R \in \mathbb{R}^{m \times n}$ be a matrix with entries drawn i.i.d. from $\mathcal{N}(0, \sigma^2)$. Then, for $n \leq m \leq 2^{O(\sqrt{n})}$, with probability at least $1 - (1/m^2)$,*

$$\text{disc}(C + R) = O\left(\sqrt{\log n} + \sigma\sqrt{n \log(2m/n)}\right).$$

For $m = O(n)$, the bound above is $O(\sqrt{\log n} + \sigma\sqrt{n})$, which is better than the bound of $O(\sqrt{\log n}(1 + \sigma\sqrt{n}))$ obtained by directly applying the best-known bound for the Komlós problem to $C + R$.

As another application, consider a matrix C with n columns and two sets of rows, A and B , where each row in A has entries in $\{0, 1\}$, and the column norm of every column restricted to rows in B is at most 1. Suppose that A has $O(n)$ rows. Applying the framework gives a coloring with $O(\sqrt{n})$ discrepancy for rows in A and $O(\sqrt{\log n})$ for rows in B .⁴ Notice that using previous techniques, if we apply the partial coloring method to get $O(\sqrt{n})$ discrepancy for A , this would give $O(\log n)$ for rows of B . On the other hand, if we apply try to obtain $O(\sqrt{\log n})$ discrepancy for B , all the known methods would incur $O(\sqrt{n \log n})$ discrepancy for A .

Relaxing the function $h(\cdot)$. Recall that the function h in Theorem 1, that controls how the ℓ_2 norms of rows decrease when restricted to subsets S of columns, and plays an important role in the bounds. In many random or pseudo-random instances however, a worst case bound on h can be quite pessimistic. For example, here even though most rows decrease significantly when restricted to S , h can remain relatively high due to a few outlier rows. The following result gives improved bound for such settings where for any subset S of columns, most row sizes restricted to S do not deviate much from their expectation if S is chosen at random.

► **Theorem 6** (Pseudo-Random Bounded Degree Hypergraphs). *Let $A \in \{0, 1\}^{m \times n}$ such that $\|A^j\|_1 \leq k$. Suppose there exists $\beta \leq k$ s.t. for any $S \subseteq [n]$ and any $c > 0$, the number of rows of A with*

$$\left| \sum_{j \in S} a_i(j) - \|a_i\|_1 \cdot (|S|/n) \right| \geq c\beta \quad (5)$$

is at most $c^{-2}|S|$. Then $\text{disc}(A) = O(\sqrt{k} + \beta)$.

As discussed in [32], one can set $\beta \leq \max_{v \perp \mathbf{1}, \|v\|=1} \|Av\|$ in (5), which in particular gives Potukuchi's result [32] for random k -regular hypergraphs as $\beta = O(k^{1/2})$ in this case.

Combining with Theorem 4, this extends to the following semi-random setting. Consider a random k -regular hypergraph A with n vertices and n edges. Suppose an adversary can arbitrarily modify A by adding or deleting vertices from edges such that degree of any vertex changes by at most t . How much can this affect the discrepancy of the hypergraph?

► **Theorem 7** (Semi-Random Hypergraphs). *Consider a random k -regular hypergraph with incidence matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, and let $C \in \{-1, 0, 1\}^{m \times n}$ be an arbitrary matrix with at most t non-zero entries per column. Then $\text{disc}(A + C) = O\left(\sqrt{k} + \sqrt{t \log n}\right)$ with probability $1 - n^{-\Omega(1)}$.*

⁴ This answers a question of Haotian Jiang.

At any time t , the change in potential $\Phi(t+1) - \Phi(t)$ is due to (i) new rows becoming small and entering \mathcal{I}_{t+1} and (ii) and the change slack of rows in \mathcal{I}_t . As each row has discrepancy 0 until it becomes small, the total contribution of step (i) over the entire algorithm is at most $n(2/b_0)^p$. So the main goal will be to show that Φ does not rise due to step (ii). This will ensure that the potential throughout the algorithm is at most $2n(2/b_0)^p$, which gives the $\sum_j a_i(j)x(j) < b_0$ for all i .

Bounding the increase in Φ . We now describe the main ideas of the algorithm and computations for the change in Φ in step (ii). The desired $O(\sqrt{\log n})$ will then follow directly by optimizing the parameters b_0 and p in (1).

Let $e_{t,i}$ denote a vector in \mathbb{R}^n with j -th entry $a_i(j)^2 x_t(j)$. At step t , x_t changes as $x_{t+1} - x_t = \varepsilon \delta \cdot v_t$ and, by a simple calculation, the approximate change in $s_i(t)$ is:

$$s_i(t+1) - s_i(t) \simeq (2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle) \varepsilon \delta + \lambda \langle a_i^{(2)}, v_t^{(2)} \rangle \delta^2,$$

where ε is a Rademacher random variable and $a^{(2)}$ denotes the vector with j -th entry $a(j)^2$. The error terms not included above are all higher powers of δ , and can be ignored for small enough δ as long as all coefficients are bounded. We formalize this in Section 2.2.

Then, up to second order terms in δ , $\Phi(t+1) - \Phi(t) \simeq f(t)\delta^2 + g(t)\varepsilon\delta$ where,

$$f(t) = -p\lambda \sum_{i \in \mathcal{I}} \frac{\langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} + \frac{p(p+1)}{2} \sum_{i \in \mathcal{I}} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+2}},$$

$$g(t) = p \sum_{i \in \mathcal{I}} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)}{s_i(t)^{p+1}}.$$

Note that the expectation of the second term $g(t)\varepsilon\delta$ is zero. So it suffices to prove that there is a choice of v_t such that $f(t) \leq 0$. This will ensure the expected change of Φ is at most *zero*, and there will be a choice of ε that ensures Φ is non-increasing. The difficulty in making $f(t)$ at most zero is that the positive part (the second term of $f(t)$) has an extra factor of $s_i(t)$ in the denominator. So if some $s_i(t)$ becomes very small, the positive term could dominate. To ensure this doesn't happen, we choose v_t to be in a subspace that makes this positive term zero for the smallest slack indices.

Blocking small slacks. Let \mathcal{J}_t be the subset of \mathcal{I} corresponding to all but the $\lfloor n_t/12 \rfloor$ smallest values of $s_i(t)$ at time t . Select v_t such that

$$(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle) = 0 \text{ for all } i \in \mathcal{I} \setminus \mathcal{J}_t, \quad (8)$$

Then as $\sum_i s_i(t)^{-p} \leq \Phi(t)$, and the smallest $n_t/12$ slacks are “blocked”, we have

$$\max_{j \in \mathcal{J}_t} \frac{1}{s_j(t)} \leq \left(\frac{\Phi(t)}{n_t/12} \right)^{1/p},$$

and so,

$$f(t) \leq p \left(\frac{p+1}{2} \sum_{i \in \mathcal{J}_t} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+1}} \max_{j \in \mathcal{J}_t} s_j(t)^{-1} - \lambda \sum_{i \in \mathcal{I}} \frac{\langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} \right)$$

$$\leq p \left(\frac{p+1}{2} \sum_{i \in \mathcal{J}_t} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+1}} \left(\frac{12\Phi(t)}{n_t} \right)^{1/p} - \lambda \sum_{i \in \mathcal{I}} \frac{\langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} \right)$$

In addition to (6) and (8), suppose v_t also satisfies

$$\sum_{i \in \mathcal{J}_t} \frac{\langle 2\lambda e_{t,i} - a_i, v_t \rangle^2}{s_i(t)^{p+1}} \leq 12 \cdot \sum_{i \in \mathcal{J}_t} \frac{\langle a_i^{(2)}, v_t \rangle^2}{s_i(t)^{p+1}}. \quad (9)$$

Choosing the update v_t . Later in Section 2.2, we will see how to find a vector v_t satisfying (6), (8), (7), and (9). Then,

$$f(t) \leq p \sum_{i \in \mathcal{J}_t} \frac{\langle a_i^{(2)}, v_t \rangle^2}{s_i(t)^{p+1}} \left(6(p+1) \left(\frac{12\Phi(t)}{n_t} \right)^{1/p} - \lambda \right).$$

To show that $f(t) \leq 0$, it thus suffices to have $6(p+1) (12\Phi(t)/n_t)^{1/p} - \lambda \leq 0$.

As $\Phi(t)^{1/p} \leq 2(2n)^{1/p}/b_0$ by the inductive hypothesis, and $n_t \geq 1$, it suffices to have $12(p+1) (24n)^{1/p} - \lambda \cdot b_0 \leq 0$. Choosing $p = \log n$ so that $n^{1/p} = O(1)$, and as $\lambda = 2^{-5}b_0$, we can pick $b_0 = O(\sqrt{\log n})$ to satisfy the above. This gives the desired discrepancy bound.

2.2 The General Framework

We now describe the algorithm more formally. Given a matrix $A \in \mathbb{R}^{m \times n}$ with $\|A^j\|_2 \leq 1$ for all $j \in [n]$, extend A such that for each original row a_i of A , there are two rows a_i and $-a_i$ in A . Additionally, partition every row a_i into 2 rows, a_i^S and a_i^L , with small and large entries, as follows:

$$a_i^S(j) = \begin{cases} 0 & \text{if } |a_i(j)| > 1/2\lambda \\ a_i(j) & \text{otherwise} \end{cases}, \quad a_i^L(j) = \begin{cases} a_i(j) & \text{if } |a_i(j)| > 1/2\lambda \\ 0 & \text{otherwise} \end{cases}$$

where λ is a parameter to be determined later. After this transformation, for any $x \in \mathbb{R}^n$, $\|Ax\|_\infty = \max_i \langle a_i^S + a_i^L, x \rangle$, and the squared 2-norm of any column of A is at most 2.

Let \mathcal{I} denote the index set of all rows of A , and \mathcal{I}^S denote the index set of rows of the first type above.

The step-size of the algorithm is δ and the algorithm will run for $T = \frac{n-2}{\delta^2}$ steps. Starting with $x_0 = 0$, let $v_t \in \mathbb{R}^n$ with $\langle x_t, v_t \rangle = 0$. For $t \in [T]$,

$$x_t = \begin{cases} x_{t-1} + \delta v_{t-1} & \text{w.p. } 1/2, \\ x_{t-1} - \delta v_{t-1} & \text{w.p. } 1/2. \end{cases}$$

As t increases, some variables will start approaching 1 in magnitude. To ensure that $x_t \in [-1, 1]^n$, we restrict v_t to be in the space of *alive* variables, defined as $\mathcal{V}_t = \{i \in [n] : |x_t(i)| < 1 - 1/(2n)\}$.

For any $t \in [T]$, $\|x_t\|^2 = \delta^2 t$ as

$$\|x_t\|^2 = \|x_{t-1} + \delta v_{t-1}\|^2 = \|x_{t-1}\|^2 + \delta^2 \|v_{t-1}\|^2 = \delta^2(t-1) + \delta^2 = \delta^2 t. \quad (10)$$

Let $n_t = |\mathcal{V}_t|$ denote the number of alive variables at t . By (10), $(n - n_t)(1 - \epsilon)^2 \leq \delta^2 t$, which gives $n_t \geq n - \frac{\delta^2 t}{(1-1/(2n))^2} > n - \delta^2 t - 1$.

To select a v_t such that for all $t \in [T]$, $x_t \in [-1, 1]^n$ and $\langle a_i, x_t \rangle$ is bounded for all rows, we classify the rows according to how many variables are still “uncolored” in a row.

Let the set of *s-Alive* rows at time t be defined as $\mathcal{I}_t = \{i \in \mathcal{I}^S : \sum_{j \in \mathcal{V}_t} a_i(j)^2 \leq 20\}$. The choice of 20 here is arbitrary, and large enough constant works.

We can now define the slack and the potential function.

1:10 A Unified Approach to Discrepancy Minimization

Slack. For any $i \in \mathcal{I}$, the slack function is defined as

$$s_i(t) = b_t - \langle a_i, x_t \rangle - \lambda \cdot \sum_{j=1}^n a_i(j)^2 (1 - x_t(j)^2).$$

We call b_t the barrier, and for $t \in [T]$, we also move it as $b_t = b_{t-1} + \delta^2 d_{t-1}$, for some function d_t . We set $\lambda = cb_0$ where $c = 1/42$ and b_0 is the initial barrier.

Potential function. The potential function has a parameter $p > 1$ and is defined as

$$\Phi(t) = \sum_{i \in \mathcal{I}_t} s_i(t)^{-p}.$$

We will only consider slacks for *alive* rows and ensure that they are always positive. Moreover, we will consider only the small *s-Alive* rows as the rows in \mathcal{I}^L will be easily handled. To ensure that $s_i(t)$ does not become too “small” for any *s-Alive* row, the choice of v_t should not decrease the smallest slacks. This motivates the following definitions.

- *Blocked* rows: Let \mathcal{C}_t be the subset of \mathcal{I}_t corresponding to the $\lfloor n_t/12 \rfloor$ smallest values of $s_i(t)$.
- Let $\mathcal{J}_t = \mathcal{I}_t \setminus \mathcal{C}_t$. These are the “large slack” rows.

To prove that all the slacks are positive, we will upper bound the potential throughout by bounding the change in $\Phi(t)$ at each step. Note that $\Phi(t)$ will experience jumps whenever a new index gets added to \mathcal{I}_t , however the total contribution of jumps is easily shown to be bounded (see Lemma 19) and can essentially be ignored. To bound the one-step change in Φ , we use the second order Taylor expansion of $\Phi(t+1)$ centered at $\Phi(t)$. Details of this can be found in the arXiv version of this paper [8].

2.3 Algorithm and Analysis

Recall that $e_{t,i}$ denotes the vector in \mathbb{R}^n with j -th entry $a_i(j)^2 x_t(j)$. We can now state the algorithm for selecting v_t .

■ **Algorithm 2** Algorithm for Selecting v_t .

-
- 1 Initialize $x_0 \leftarrow 0$
 - 2 **for** $t = 1, \dots, T = \frac{n-2}{\delta^2}$ **do**
 - 3 Let $\mathcal{W}_t = \{w \in \mathbb{R}^n : w(i) = 0, \forall i \notin \mathcal{V}_t\}$ // restrict to alive variables
 - 4 Let $\mathcal{U}_t = \{w \in \mathcal{W}_t : \langle w, 2\lambda e_{t,i} - a_i \rangle = 0, \forall i \in \mathcal{C}_t \text{ and } \langle w, x_t \rangle = 0\}$ // restrict to large slack rows
 - 5 Let $\mathcal{Y}_t = \{w \in \mathcal{W}_t : \langle w, a_i \rangle = 0, \forall i \in \mathcal{I} \setminus \mathcal{I}_t\}$ // restricted to s-Alive rows
 - 6 Let \mathcal{G}_t denote the subspace

$$\mathcal{G}_t = \left\{ w \in \mathcal{W}_t : \sum_{i \in \mathcal{J}_t} \frac{\langle (2\lambda e_{t,i} - a_i), w \rangle^2}{s_i(t)^{p+1}} \leq 40 \sum_{i \in \mathcal{J}_t} \frac{\langle a_i^{(2)}, w^{(2)} \rangle}{s_i(t)^{p+1}} \right\} \quad (11)$$

- 7 Consider the subspace $\mathcal{Z}_t = \mathcal{U}_t \cap \mathcal{Y}_t \cap \mathcal{G}_t$ and let $W = \{w_1, w_2, \dots, w_k\}$ be an orthonormal basis for \mathcal{Z}_t . Choose

$$v_t = \arg \min_{w \in W} \sum_{i \in \mathcal{J}_t} \langle 2\lambda e_{t,i} - a_i, w \rangle^2 s_i(t)^{-(p+1)}. \quad (12)$$

We now re-state our main theorem. In words, the assumption of the theorem is that there is a non-decreasing function $h(\cdot)$ such that for any row, the squared norm in any subset of coordinates S is proportional to $h(|S|)$ times the size of the subset S . Under this condition, we can bound the discrepancy as a function of h .

► **Theorem 1.** *For a $A \in \mathbb{R}^{m \times n}$ with $\|A^j\|_2 \leq L$ and $|a_i(j)| \leq M$ for all $i \in [m], j \in [n]$, let $h : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be a non-increasing function such that for every subset $S \subseteq [n]$ and $i \in [m]$,*

$$\sum_{j \in S} a_i(j)^2 \leq |S| \cdot h(|S|). \quad (3)$$

Then, for any $p > 1$, there exists a vector $x \in \{-1, 1\}^n$ such that $\|Ax\|_\infty \leq 5b_0 + 2M$, where

$$b_0 = \min \left(\sqrt{8(p+1)(48m)^{1/p} \cdot \beta}, 250L\sqrt{\log(2m)} \right). \quad (4)$$

where $\beta = \int_{t=0}^{n-2} h(n-t)(n-t)^{-1/p} dt$.

The case when $h(t) = h$ is often useful, for which case we have following corollary.

► **Corollary 8.** *For a matrix $A \in \mathbb{R}^{m \times n}$ with $\|A^j\| \leq L$ and $|a_i(j)| \leq M$ for all $i \in [n], j \in [m]$, let h be such that for every subset $S \subseteq [n]$ and every $i \in [m]$, $\sum_{j \in S} a_i(j)^2 \leq |S| \cdot h$. Then, $\text{disc}(A) \leq 5b_0 + 2M$, where $b_0 = \min(26\sqrt{hn \log(2m/n)}, 250L\sqrt{\log(2m)})$.*

Roadmap of the proof. The first main lemma below (Lemma 10) establishes that there is a large feasible subspace from which v_t as defined above can be chosen. Using this we prove Lemma 11, which bounds the change in potential. This will allow us to bound the discrepancy of each row and hence prove Theorem 1.

A key fact used for proving Lemma 10 is the following lemma in [7].

► **Lemma 9** ([7]). *Let $G, H \in \mathbb{R}^{m \times n}$ be matrices such that $|G_{ij}| \leq \alpha |H_{ij}|$ for all $i \in [m]$ and $j \in [n]$. Let $K = \text{diag}(H^\top H)$. Then for any $\beta \in (0, 1]$, there exists a subspace $W \subseteq \mathbb{R}^n$ such that $\dim(W) \geq (1 - \beta)n$, and $\forall w \in W$, $w^\top G^\top G w \leq \frac{\alpha^2}{\beta} \cdot w^\top K w$.*

We now arrive at the main Lemma.

► **Lemma 10** (Subspace Dimension). *For all $t \in T$, $\dim(\mathcal{Z}_t) \geq \lceil 2n_t/3 \rceil$.*

Setting the parameters. To show the two bounds in (4), we will set the parameters b_t, d_t (the change in b_t) and p in two ways:

$$\text{Case 1: } d_t = 4(p+1) \cdot h(n_t) \cdot \max_{i \in \mathcal{J}_t} s_i(t)^{-1} \text{ for all } t \in [T], \text{ and } p, b_0 \text{ arbitrary} \quad (13)$$

$$\text{Case 2: } p = 2 \log(2m), b_0 = 840(p+1) \cdot \max_{j \in \mathcal{J}_t} s_j(t)^{-1} \text{ and } d_t = 0 \text{ for all } t \in [T]. \quad (14)$$

Bounding the potential. The next lemma shows that in both these cases, the potential function remains bounded.

► **Lemma 11** (Bounded Potential). *In either of the cases given by (13) and (14), we have that $\Phi(t) \leq 4m(2/b_0)^p$, for all $t = 0, \dots, T$.*

The next lemma gives a bound on the minimum value of slack for any active row, given the bound on potential function.

► **Lemma 12.** *For any $t \in \{0, \dots, T\}$, if $\Phi(t) \leq 4m(2/b_0)^p$, then $\max_{i \in \mathcal{J}_t} s_i(t)^{-1} \leq \frac{2}{b_0} \left(\frac{48m}{n_t} \right)^{\frac{1}{p}}$.*

► **Lemma 13.** *For any $t \in [T]$, the choice of v_t satisfies*

$$\sum_{i \in \mathcal{J}_t} \frac{\langle 2\lambda e_{t,i} - a_i, v_t \rangle^2}{s_i(t)^{p+1}} \leq \sum_{i \in \mathcal{J}_t} \frac{8h(n_t)}{s_i(t)^{p+1}}. \quad (15)$$

These lemmas will allow us to prove the main theorem (see Appendix).

3 Applications

3.1 Set Coloring

We bound the discrepancy of a set system (U, \mathcal{S}) with $|U| = n$, $|\mathcal{S}| = m$, and $m \geq n$. As $\|A^j\|_2 \leq \sqrt{m}$, we have $L = \sqrt{m}$, and as $\sum_{j \in \mathcal{S}} a_i(j)^2 \leq |\mathcal{S}|$, we can set $h(t) = 1$ for all $t \in [n]$. Consider (4) and suppose $p \geq 1.1$ so that $p/(p-1) = O(1)$. Then

$$\beta = \int_{t=0}^{n-2} h(n-t) \cdot (n-t)^{-1/p} dt = O(n^{1-1/p}),$$

and the first bound in (4) gives $b_0 = O(pn^{1/2}(m/n)^{1/p})$. Setting $p = \log(2m/n)$ gives Spencer's $O(\sqrt{n \log(2m/n)})$ bound.

3.2 Vector Balancing

We now consider the discrepancy a matrix $A \in \mathbb{R}^{m \times n}$ with column ℓ_2 -norms at most 1.

Here $L = 1$ and the second term in (4) directly gives a $O(\sqrt{\log m})$ bound. This also implies an $O(\sqrt{\log n})$ bound as at most n^2 rows can have ℓ_1 -norm more than 1, and we can assume that $m \leq n^2$. In particular, for a row a_i with $\|a_i\|_2 < 1/n^{1/2}$, we have $|\langle a_i, x \rangle| \leq \|a_i\|_1 \leq \sqrt{n} \|a_i\|_2 < 1$ and it can be ignored. The sum of squares of elements in A is at most n the number of rows with $\|a_i\|_2 > 1/n^{1/2}$ is at most n^2 .

3.3 Sub-Gaussian Matrices and Random Matrices

We give the proofs for these application in the appendix.

4 Flexibility of the Method

An advantage of the potential function approach is its flexibility. We describe two illustrative applications. In Section A.2 we show how the bounds for matrices A and B obtained using the framework can be used to directly give bounds for $C = A + B$ by combining the potentials for A and B in a natural way.

In Section 4.1 we consider how the requirement on the function $h(\cdot)$ in Theorem 1 can be relaxed, and use it to bound the discrepancy of sparse hypergraphs (the Beck-Fiala setting) satisfying a certain pseudo-randomness condition.

4.1 Discrepancy of Sparse Pseudo-random Hypergraphs

In this section, we consider 0/1 matrices that satisfy a certain regularity property, namely, for most rows, the sum of their entries in any subset of columns is close to the sum of the full row scaled by the fraction of columns in the subset. This property is satisfied, e.g., by the matrices that correspond to sparse random hypergraphs. In particular, we show the following.

► **Theorem 6** (Pseudo-Random Bounded Degree Hypergraphs). *Let $A \in \{0,1\}^{m \times n}$ such that $\|A^j\|_1 \leq k$. Suppose there exists $\beta \leq k$ s.t. for any $S \subseteq [n]$ and any $c > 0$, the number of rows of A with*

$$\left| \sum_{j \in S} a_i(j) - \|a_i\|_1 \cdot (|S|/n) \right| \geq c\beta \quad (5)$$

is at most $c^{-2}|S|$. Then $\text{disc}(A) = O(\sqrt{k} + \beta)$.

Proof outline. At a high level the proof is similar to that of Theorem 4, using a weighted potential function. However, rather than just two potentials, we will have to consider a combination of $O(\log n)$ potentials, and it will take some care to make sure this doesn't create an overhead in the discrepancy. We note that the main algorithm remains: at each step choose a vector in a subspace defined by a set of constraints based on the current vector x_t .

We next discuss the details of the algorithm and the proof of Theorem 6. The full proof can be found in the arXiv version of this paper [8].

Partitioning rows according to ℓ_1 -norm. First, extend A such that for each original row a_i , there are two rows a_i and $-a_i$ in A . Since our goal is to prove discrepancy $O(\sqrt{k})$, we can ignore all rows with ℓ_1 -norm less than \sqrt{k} . Then $m \leq n\sqrt{k}$ because the number of rows with ℓ_1 -norm greater than \sqrt{k} is at most $2nk/\sqrt{k} = 2n\sqrt{k}$. Let $N = \lceil \log_2 n/k \rceil$ and $\mathcal{Q} = \{0\} \cup [N]$. Partition the rows of A into based on their initial ℓ_1 -norm into $|\mathcal{Q}| = N + 1$ classes:

- $\mathcal{A}_0 = \{i \in \mathcal{I} : \sqrt{k} \leq \|a_i\|_1 < 2k\}$.
- For each $i \in [N]$, let $\mathcal{A}_i = \{i \in \mathcal{I} : 2^i k \leq \|a_i\|_1 < 2^{i+1} k\}$.

The sum of ℓ_1 -norms of rows in A is at most $2nk$, therefore for any i , $2^i k |\mathcal{A}_i| \leq 2nk$ and $|\mathcal{A}_i| \leq 2^{1-i}n$.

We create $N + 1$ potential functions $\{\Phi_i(t)\}_{i=0}^N$, one associated with each row partition. The potential functions use the same p, b_0 parameters, and $\lambda = cb_0$ with $c = 1/42$, but have different rate of change of barrier functions $d_q(\cdot)$, based on q . We will run Algorithm 2 on each partition separately but use the same x_t and v_t at each step. In this case, we can select parameters to ensure that each potential function is decreasing in expectation (see Lemma 18). However, there might not exist a vector v_t that ensure that moving in v_t direction decreases all the potential functions simultaneously. To deal with this, we use a weighted combination of Φ_q as the potential function:

$$\Phi(t) = \frac{1}{k} \cdot \Phi_0(t) + \sum_{q \geq 1} 2^{2q} \cdot \Phi_q(t). \quad (16)$$

4.1.1 A suitable subspace

To identify the constrained subspace for the PotentialWalk (Algorithm 6), we use the following definitions. The set of *Active* rows is defined as $\mathcal{I}_t = \{i \in \mathcal{I} : \sum_{j \in \mathcal{V}_t} |a_i(j)| \leq 12k\}$. For each class q , let $h_q : \mathbb{R}^+ \rightarrow \mathbb{R}$ be a non-increasing function such that for every subset $S \subseteq n$, at most $n_t/16$ rows i from class \mathcal{A}_q violate the condition

$$\sum_{j \in S} |a_i(j)| \leq |S| \cdot h_q(|S|). \quad (17)$$

While following the general framework from Section 2.2, we make three crucial changes:

- Move orthogonal to rows with *large deviation*. At step t , the ℓ_1 norm of row a_i will be close to $(n_t/n) \cdot \|a_i\|_1$ for most rows. Let $a_{i,t}$ denote a vector in \mathbb{R}^n with j -th entry $\mathbf{1}_{j \in \mathcal{V}_t} a_i(j)$, i.e., $a_{i,t}$ is row a_i restricted to the alive coordinates at time t . Then the set of *large deviation* rows consists of rows that deviate significantly from this expected value

$$\mathcal{B}_t = \{i \in \mathcal{I} : \|\|a_{i,t}\|_1 - \|a_i\|_1 \cdot (n_t/n)\| \geq 4\beta\}. \quad (18)$$

For any $t \in [T]$, (5) implies that $\dim(\mathcal{B}_t) \leq \lfloor n_t/16 \rfloor$.

- Ignore *Dead* rows. As soon as the ℓ_1 -norm of some row becomes less than 8β , we drop it from the potential function. The set of *dead* rows at step t is defined as

$$\mathcal{D}_t = \{i \in \mathcal{I} : \|a_{i,t}\|_1 \leq 8\beta\}. \quad (19)$$

For a dead row, rather than keeping track of its discrepancy using a slack function, we uniformly bound the the additional discrepancy gained by a row after it becomes dead.

- *Block rows based on their initial size.* For $q \in \mathcal{Q}$, let \mathcal{C}_t^q be the subset of $\mathcal{A}_q \cap \mathcal{I}_t$ corresponding to the $\lfloor 2^{i-8} n_t^2 / n \rfloor$ smallest values of $\{s_i(t) : i \in \mathcal{A}_q \cap \mathcal{I}_t\}$, and let $\mathcal{J}_t^q = \mathcal{A}_i \setminus \{\mathcal{C}_t^q \cup \mathcal{D}_t\}$.

We are ready to state the algorithm for selecting v_t .

Algorithm 3 Algorithm for Selecting v_t .

- ```

1 Let $h_q(n_t) = 2^{q+2}/n$ and $w_q(t) = 2^{5-\frac{q}{4}} \left(\frac{n}{n_t}\right)^{1/4}$
2 for $t = 1, \dots, T$ do
3 Let $\mathcal{W}_t = \{w \in \mathbb{R}^n : w(i) = 0, \forall i \in \mathcal{V}_t\}$ // restrict to alive variables
4 Let $\mathcal{U}_t = \{w \in \mathcal{W}_t : \langle w, 2cb_0e_{t,i} - a_i \rangle = 0, \forall i \in \mathcal{C}_t \text{ and } \langle w, x_t \rangle = 0\}$
// restrict to large slack rows
5 Let $\mathcal{Y}_t = \{w \in \mathcal{W}_t : \langle w, a_i \rangle = 0, \forall i \in \mathcal{I} \setminus \mathcal{I}_t\}$ // move orthogonal to large norm
rows
6 Let $\mathcal{G}_t = \{w \in \mathcal{W}_t : \langle a_i, w \rangle = 0, \forall i \in \mathcal{B}_t\}$
// move orthogonal to large deviation rows
7 Let $\mathcal{Z}_t = \mathcal{U}_t \cap \mathcal{Y}_t \cap \mathcal{G}_t$ and let $W = \{w_1, \dots, w_k\}$ be an orthonormal basis for \mathcal{Z}_t
8 Let $v_t \in W$ such that for all $q \in \mathcal{Q}$,

```

$$\sum_{i \in \mathcal{J}_t^q} \langle 2cb_0 e_{t,i} - a_i, v_t \rangle^2 s_i(t)^{-p-1} \leq 8w_q(t) \cdot h_q(n_t) \sum_{i \in \mathcal{J}_t^q} s_i(t)^{-p-1}. \quad (20)$$

## 4.2 Proof Outline

The following lemma bounds the number of active classes at step  $t$ .

► **Lemma 14.** *At step  $t$ , the following two conditions hold: (i) The number of classes  $q$  for which  $\mathcal{A}_q \cap \{\mathcal{I}_t \setminus \{\mathcal{B}_t \cup \mathcal{D}_t\}\} \neq \emptyset$  is at most  $\log(16n/n_t)$  and (ii)  $h_q(t) = 2^{q+2}k/n$  satisfies (17) for all  $q \in \mathcal{Q}$ .*

So at any step  $t$ , the set of active rows is from the first  $\log_2(16n/n_t)$  classes of rows. It also helps us define two important parameters associated with a row class  $q$ . At step  $t$ , consider a  $q \in \mathcal{Q}$  with  $\mathcal{A}_q \cap \{\mathcal{I}_t \setminus \{\mathcal{B}_t \cup \mathcal{D}_t\}\} \neq \emptyset$ .

- Since  $n - \delta^2 t - 1 < n_t \leq 16 \cdot 2^{-q} n$ , for  $q \geq 1$ , let  $t_q := \max \{0, n\delta^{-2} (1 - 16 \cdot 2^{-q} - 1/n)\}$ . Similarly, let  $t_0 := n\delta^{-2} (1 - 16k^{-1/2} - 1/n)$ .
- On the other hand,  $q$  must satisfy  $2^q \leq \frac{16n}{n_t}$ . Let  $q_t := \arg \max_{i \geq 0} \{2^i \leq 16 \cdot (n/n_t)\}$ .

The next two lemmas are analogous to Lemma 10 and Lemma 13, respectively.

► **Lemma 15.** *For any  $t \in [T]$ , it holds that  $\dim(\mathcal{Z}_t) \geq \lceil n_t/2 \rceil$ .*

► **Lemma 16.** *For all  $t \in [T]$ , there exists  $v_t \in \mathcal{Z}_t$  such that  $\forall q \in \mathcal{Q}$ ,*

$$\sum_{i \in \mathcal{J}_t^q} \langle 2cb_0 e_{t,i} - a_i, v_t \rangle^2 s_i(t)^{-p-1} \leq 8w_q(t) \cdot h_q(n_t) \sum_{i \in \mathcal{J}_t^q} s_i(t)^{-p-1}. \quad (21)$$

Note that for any row  $i \in \mathcal{A}_q$ , at  $t \leq t_q$ ,  $\langle 2cb_0 e_{t,i} - a_i, v_t \rangle = 0$ . So, we can set  $d_t^q = 0$  for rows in class  $q$ . Lemma 11 and Lemma 16 imply that for all the potential functions to be decreasing, it suffices to have

$$d^q(t) = \begin{cases} 0 & \text{if } t \leq t_q \\ 4(p+1) \cdot w_q(t) \cdot h_q(n_t) \cdot \max_{i \in \mathcal{J}_t^q} s_i(t)^{-1} & \text{otherwise.} \end{cases} \quad (22)$$

The next lemma helps us bound the rate of change of  $b_q(t)$ , which eventually gives a bound on  $b_q(T)$  in Theorem 6.

► **Lemma 17.** *For any  $t \in \{0, \dots, T\}$ , if  $\Phi(t) \leq 8n \left(\frac{2}{b_0}\right)^p \left(\frac{16n}{n_t}\right)$ , then*

$$\max_{j \in \mathcal{J}_t^q} s_j(t)^{-1} \leq \begin{cases} k^{1/p} \cdot \frac{2^{1+15/p}}{b_0} \left(\frac{n}{n_t}\right)^{3/p} & \text{if } q = 0 \\ \frac{2^{1+(15-3q)/p}}{b_0} \left(\frac{n}{n_t}\right)^{3/p} & \text{if } q \geq 1. \end{cases} \quad (23)$$

► **Lemma 18.** *For  $p = 8$  and  $d_q$  given by (22), for all  $t \in [T]$ , we have  $\Phi(t) \leq \frac{2^7 n^2}{n_t} \cdot \left(\frac{2}{b_0}\right)^p$ .*

**Proof of Theorem 6.** If row  $i \in \mathcal{A}_q$  becomes dead after step  $t - 1$ , then

$$|\langle a_i, x_T \rangle| \leq |\langle a_i, x_t \rangle| + |\langle a_i^S, x_T - x_t \rangle| \leq b_t(q) + 2 \sum_{j \in \mathcal{V}_t} |a_i(j)| \leq b_T(q) + 16\beta.$$

Substituting the bound on  $\max_{i \in \mathcal{J}_t^q} s_i(t)^{-1}$  from (23), and using  $w_q(t) = 2^{5-q/4} \cdot (n/n_t)^{1/4}$  and  $h_q(t) = 2^{q+2}/n$ , equation (22) gives  $d_q(t) = 0$  for  $t < t_q$ , and

$$d_q(t) = \begin{cases} 9k \cdot \frac{2^{3q/8+14}}{nb_0} \left(\frac{n}{n-\delta^2 t-1}\right)^{5/8} & \text{if } q \geq 1 \text{ and } t \geq t_q \\ 9k^{\frac{9}{8}} \cdot \frac{2^{14}}{nb_0} \left(\frac{n}{n-\delta^2 t-1}\right)^{5/8} & \text{if } q = 0 \text{ and } t \geq t_0. \end{cases}$$

For any  $q \geq 1$ , summing up  $d_q(\cdot)$ ,

$$\begin{aligned} b_q(T) &= b_0 + \delta^2 \sum_{t=t_q}^{T-1} d_q(0) \leq \delta^2 \int_{t=t_q}^T \frac{9k \cdot 2^{3q/4+12+(15-3q)/8}}{nb_0} \left( \frac{n}{n-\delta^2 t-1} \right)^{5/8} dt \\ &\leq b_0 + \int_{t=\delta^2 t_q}^{n-2} \frac{9k \cdot 2^{3q/8+14}}{nb_0} \left( \frac{n}{n-t-1} \right)^{5/8} dt = b_0 + \frac{2^{20}k}{b_0}. \end{aligned}$$

For  $b_0 = 2^{10}\sqrt{k}$ ,  $b_q(T) \leq 2^{11}\sqrt{k}$  for all  $q \geq 1$ . Similar calculation for  $q = 0$  show that  $b_0 = 2^{10}\sqrt{k}$  and  $b_T(0) = 2^{11}\sqrt{k}$  suffice.

Let  $x \in \{-1, 1\}^n$  be obtained from  $x_T$  by the rounding  $x(j) = \text{sign}(x_T(j))$ . Since  $T = (n-2)/\delta^2$ ,  $\|x_T\|^2 = n-2$  with  $|x_T(j)| \leq 1$  for all  $j \in [n]$ . After rounding  $x_T$  to  $x$ ,  $\|x\|^2 = n$  and  $|\langle a_i, x \rangle| \leq |\langle a_i, x_T \rangle| + |\langle a_i, x - x_T \rangle| \leq 2b_T + 16\beta + \sum_j |x(j) - x_T(j)| \leq b_T + 16\beta + 2$ . ◀

**Random and Semi-random Sparse Hypergraphs.** This gives an alternate proof of the result [32] of Potukuchi that  $\text{disc}(\mathcal{H}) = O(\sqrt{k})$  for regular random  $k$ -regular hypergraph  $\mathcal{H}$ , on  $n$  vertices and  $m$  edges with  $m \geq n$  and  $k = o(m^{1/2})$ . In particular, Potukuchi showed that such hypergraphs satisfy condition (5) with high probability.

**Proof of Theorem 7.** By the subadditive property of stochastic discrepancy,  $\text{disc}(A + C) \leq O(\sqrt{k}) + O(\sqrt{t \log n})$ . However, this bound is not algorithmic because it requires running the algorithm separately on  $A$  and  $A_C - A$ . ◀

---

## References

- 1 Dylan J Altschuler and Jonathan Niles-Weed. The discrepancy of random rectangular matrices. *arXiv preprint*, 2021. [arXiv:2101.04036](#).
- 2 Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2014.
- 3 Wojciech Banaszczyk. Balancing vectors and gaussian measures of  $n$ -dimensional convex bodies. *Random Structures & Algorithms*, 12(4):351–360, 1998.
- 4 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 3–10. IEEE, 2010.
- 5 Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for Komlós conjecture matching Banaszczyk’s bound. *SIAM Journal on Computing*, 48(2):534–553, 2019.
- 6 Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The Gram-Schmidt walk: A cure for the Banaszczyk blues. *Theory Comput.*, 15:1–27, 2019.
- 7 Nikhil Bansal and Shashwat Garg. Algorithmic discrepancy beyond partial coloring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 914–926, 2017.
- 8 Nikhil Bansal, Aditi Laddha, and Santosh S Vempala. A unified approach to discrepancy minimization. *arXiv preprint*, 2022. [arXiv:2205.01023](#).
- 9 Nikhil Bansal and Raghu Meka. On the discrepancy of random low degree set systems. *Random Structures & Algorithms*, 57(3):695–705, 2020.
- 10 Nikhil Bansal and Joel Spencer. Deterministic discrepancy minimization. *Algorithmica*, 67(4):451–471, 2013.
- 11 Nikhil Bansal and Joel H. Spencer. On-line balancing of random inputs. *Random Struct. Algorithms*, 57(4):879–891, 2020.
- 12 Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
- 13 József Beck. Roth’s estimate of the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1(4):319–325, 1981.



- 14 József Beck. Discrepancy theory. *Handbook of combinatorics*, pages 1405–1446, 1995.
- 15 József Beck and Tibor Fiala. “Integer-making” theorems. *Discrete Applied Mathematics*, 3(1):1–8, 1981.
- 16 Karthekeyan Chandrasekaran and Santosh S Vempala. Integer feasibility of random polytopes: random integer programs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 449–458, 2014.
- 17 Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, 2001.
- 18 Ronen Eldan and Mohit Singh. Efficient algorithms for discrepancy minimization in convex sets. *Random Struct. Algorithms*, 53(2):289–307, 2018.
- 19 Esther Ezra and Shachar Lovett. On the beck-fiala conjecture for random set systems. *Random Structures & Algorithms*, 54(4):665–675, 2019.
- 20 Efim Davydovich Gluskin. Extremal properties of orthogonal parallelepipeds and their applications to the geometry of banach spaces. *Mathematics of the USSR-Sbornik*, 64(1):85, 1989.
- 21 Nicholas Harvey, Roy Schwartz, and Mohit Singh. Discrepancy without partial colorings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
- 22 Rebecca Hoberg and Thomas Rothvoss. A fourier-analytic approach for the discrepancy of random set systems. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2547–2556. SIAM, 2019.
- 23 Yin Tat Lee and Mohit Singh. Personal communication, 2016.
- 24 Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 250–269. IEEE, 2015.
- 25 Yin Tat Lee and Santosh S Vempala. Stochastic localization+ stieltjes barrier= tight bound for log-sobolev. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1122–1129, 2018.
- 26 Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss. Deterministic discrepancy minimization via the multiplicative weight update method. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 380–391. Springer, 2017.
- 27 Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM Journal on Computing*, 44(5):1573–1582, 2015.
- 28 Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 1999.
- 29 Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer, 2009.
- 30 Jiri Matousek. The determinant bound for discrepancy is almost tight. *Proceedings of the American Mathematical Society*, 141:451–60, 2013.
- 31 Jiri Matousek and Aleksandar Nikolov. Combinatorial discrepancy for boxes via the  $\gamma_2$  norm. In *31st International Symposium on Computational Geometry, SoCG*, volume 34, pages 1–15, 2015.
- 32 Aditya Potukuchi. A spectral bound on hypergraph discrepancy. In *47th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 93:1–93:14, 2020.
- 33 Thomas Rothvoss. Constructive discrepancy minimization for convex sets. *SIAM Journal on Computing*, 46(1):224–234, 2017.
- 34 Joel Spencer. Six standard deviations suffice. *Transactions of the American mathematical society*, 289(2):679–706, 1985.
- 35 Nikhil Srivastava, Roman Vershynin, et al. Covariance estimation for distributions with  $2+\epsilon$  moments. *The Annals of Probability*, 41(5):3081–3111, 2013.
- 36 Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

## A Appendix: Proof of main theorem

**Proof of Lemma 11.** We will prove this by induction. Clearly, this holds at  $t = 0$  as  $\Phi(0) \leq 2m(2/b_0)^p$ . For the inductive step, we will show that for any  $j = 0, \dots, T-1$ , if  $\Phi(j) \leq 4m(2/b_0)^p$  then

$$\Phi(j+1) \leq \Phi(j) + \frac{1}{Tb_0^p} + |\mathcal{I}_{j+1} \setminus \mathcal{I}_j| \cdot \left(\frac{2}{b_0}\right)^p. \quad (24)$$

Note that  $|\mathcal{I}_{j+1} \setminus \mathcal{I}_j|$  is the number of additional rows in  $\mathcal{I}^S$  that may become alive at step  $j$ . This gives the result by induction as summing (24) over  $j = 0, \dots, T-1$  will give

$$\Phi(t+1) \leq \Phi(0) + \sum_{j=0}^{T-1} \frac{1}{Tb_0^p} + \left(\frac{2}{b_0}\right)^p \sum_{j=0}^{T-1} |\mathcal{I}_{j+1} \setminus \mathcal{I}_j| \leq 2m \cdot \left(\frac{2}{b_0}\right)^p + \frac{1}{b_0^p} \leq 4m \cdot \left(\frac{2}{b_0}\right)^p. \quad (25)$$

We now focus on proving (24) for  $j = t$ .

By the induction hypothesis,  $\Phi(t) \leq 4m(2/b_0)^p$ . By Lemma 19, one of the signs for  $x_{t+1}$  gives  $\mathbb{E}(\Phi(t+1)) - \Phi(t) \leq f(t) + \frac{1}{Tnb_0^p} + |\mathcal{I}_{t+1} \setminus \mathcal{I}_t| \cdot \left(\frac{2}{b_0}\right)^p$ , where

$$f(t) = -p\delta^2 \sum_{i \in \mathcal{I}_t} \frac{d_t + \lambda \langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} + \frac{p(p+1)\delta^2}{2} \sum_{i \in \mathcal{I}_t} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+2}}.$$

So to prove (24), it suffices to show that  $f(t) \leq 0$ . We first consider the case when  $b_t, d_t$  and  $p$  are given by (13). As  $2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle = 0$  for all  $i \notin \mathcal{J}_t$ ,  $f(t)$  satisfies

$$f(t) \leq -p\delta^2 \sum_{i \in \mathcal{J}_t} \frac{d_t + \lambda \langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} + \frac{p(p+1)\delta^2}{2} \max_{j \in \mathcal{J}_t} s_j(t)^{-1} \cdot \sum_{i \in \mathcal{J}_t} \frac{(\langle 2\lambda e_{t,i} - a_i, v_t \rangle)^2}{s_i(t)^{p+1}}. \quad (26)$$

By a simple averaging argument described in Lemma 13, we also have that

$$\sum_{i \in \mathcal{I}_t} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+1}} \leq \sum_{i \in \mathcal{I}_t} \frac{8h(n_t)}{s_i(t)^{p+1}}. \quad (27)$$

Plugging (27) in (26) gives

$$f(t) \leq -p\delta^2 \sum_{i \in \mathcal{J}_t} \frac{d_t}{s_i(t)^{p+1}} + \frac{p(p+1)\delta^2}{2} \max_{j \in \mathcal{J}_t} s_j(t)^{-1} \cdot \sum_{i \in \mathcal{J}_t} \frac{8h(n_t)}{s_i(t)^{p+1}}. \quad (28)$$

Therefore, if  $d_t$  satisfies equation (13), then  $f(t) \leq 0$ .

We now consider the case in (14). As  $v_t \in \mathcal{G}_t$ , we have

$$\sum_{i \in \mathcal{J}_t} \frac{(2\lambda \langle e_{t,i}, v_t \rangle - \langle a_i, v_t \rangle)^2}{s_i(t)^{p+1}} \leq 40 \cdot \sum_{i \in \mathcal{J}_t} \frac{\langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}}. \quad (29)$$

Next, as  $d_t = 0$  and  $\lambda = b_0/42$ , (26) and (29) give

$$f(t) \leq \sum_{i \in \mathcal{J}_t} \frac{p\delta^2 \langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} \cdot \left( -\frac{b_0}{42} + 20(p+1) \cdot \max_{j \in \mathcal{J}_t} s_j(t)^{-1} \right).$$

So if  $b_0$  satisfies equation (14), then  $f(t) \leq 0$ . ◀

**Proof of Lemma 12.** By the definition of  $\mathcal{J}_t$ , for any  $i \in \mathcal{J}_t$ , there are at least  $\lfloor n_t/12 \rfloor + 1$  indices  $j$  in  $\mathcal{I}_t$  such that  $s_j(t) \leq s_i(t)$ . Therefore,

$$\max_{i \in \mathcal{J}_t} \frac{1}{s_i(t)} \leq \left( \frac{12\Phi(t)}{n_t} \right)^{\frac{1}{p}} \leq \frac{2}{b_0} \left( \frac{48m}{n_t} \right)^{\frac{1}{p}}, \quad (30)$$

where the last inequality follows by the assumption,  $\Phi(t) \leq 4m(2/b_0)^p$ .  $\blacktriangleleft$

**Proof of Lemma 13.** Using  $(a+b)^2 \leq 2(a^2+b^2)$ , and as  $|2\lambda e_{t,i}(j)| = |2\lambda a_i(j)^2 x_t(j)| \leq |a_i(j)|$  as  $|a_i(j)| \leq 1/2\lambda$  for any  $j$  and  $i \in \mathcal{I}^S$ , we have that for any  $w$ ,

$$\sum_{i \in \mathcal{J}_t} \frac{\langle 2\lambda e_{t,i} - a_i, w \rangle^2}{s_i(t)^{p+1}} \leq \sum_{i \in \mathcal{J}_t} \frac{2\langle a_i, w \rangle^2 + 2\langle 2\lambda e_{t,i}, w \rangle^2}{s_i(t)^{p+1}} \leq 4 \sum_{i \in \mathcal{J}_t} \frac{\langle a_i, w \rangle^2}{s_i(t)^{p+1}}.$$

Let  $W_t = \{w_1, \dots, w_k\}$  be an orthonormal basis for  $\mathcal{Z}_t$  and  $k = \dim(\mathcal{Z}_t)$ . As  $\mathcal{Z}_t \subseteq \mathcal{V}_t$ ,

$$\sum_{i \in \mathcal{J}_t} \frac{\sum_{j=1}^k \langle a_i, w_j \rangle^2}{s_i(t)^{p+1}} \leq \sum_{i \in \mathcal{J}_t} \frac{\sum_{j \in \mathcal{V}_t} a_i(j)^2}{s_i(t)^{p+1}} \leq n_t \sum_{i \in \mathcal{J}_t} \frac{h(n_t)}{s_i(t)^{p+1}}.$$

where the second inequality uses that  $\sum_{j \in \mathcal{V}_t} a_i(j)^2 \leq n_t \cdot h(n_t)$  by the definition of  $h$ .

As  $k \geq \lceil n_t/2 \rceil$ , this gives

$$\frac{1}{k} \sum_{j=1}^k \sum_{i \in \mathcal{J}_t} \frac{\langle 2\lambda e_{t,i} - a_i, w_j \rangle^2}{s_i(t)^{p+1}} \leq \frac{n_t}{k} \sum_{i \in \mathcal{J}_t} \frac{4h(n_t)}{s_i(t)^{p+1}} \leq \sum_{i \in \mathcal{J}_t} \frac{8h(n_t)}{s_i(t)^{p+1}}.$$

The result now follows as  $v_t$  in (12) minimizes  $\sum_{i \in \mathcal{J}_t} \langle 2\lambda e_{t,i} - a_i, w_j \rangle^2 s_i(t)^{-p-1}$  over all  $w_j \in W_t$ .  $\blacktriangleleft$

**Proof of Lemma 10.** To lower bound the dimension of  $\mathcal{Z}_t$  we lower bound the dimensions of  $\mathcal{U}_t, \mathcal{Y}_t$  and  $\mathcal{G}_t$ .

First, we have  $\dim(\mathcal{U}_t) \geq n_t - \dim(\mathcal{C}_t) - 1 \geq \lceil 11n_t/12 \rceil - 1$ . Second, at time  $t$ , as the sum of  $\ell_2$ -norm square of all columns is at most  $2n_t$ , we have that  $\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{V}_t} a_i(j)^2 \leq 2n_t$ . So the number of rows  $a_i$  with  $\sum_{j \in \mathcal{V}_t} a_i(j)^2 \geq 20$  is at most  $\lfloor n_t/10 \rfloor$  and  $\dim(\mathcal{Y}_t) \geq n_t - \lfloor n_t/10 \rfloor = \lceil 9n_t/10 \rceil$ .

We now bound  $\dim(\mathcal{G}_t)$  by applying Lemma 9. Let  $G$  denote the matrix with columns  $j$  corresponding to variables in  $\mathcal{V}_t$  and rows  $i$  restricted to  $i \in \mathcal{J}_t$  with  $(i, j)$  entry  $(2\lambda e_{t,i}(j) - a_i(j))s_i(t)^{-(p+1)/2}$ .

Let  $H$  be the matrix with entries  $a_i(j) \cdot s_i(t)^{-(p+1)/2}$  for  $i \in \mathcal{J}_t$  and  $j \in \mathcal{V}_t$ . As  $|a_{ij}| \leq 1/(2\lambda)$  for  $i \in \mathcal{I}_t$ , we have

$$|G_{ij}| = |2\lambda a_i(j)^2 x_t(j) - a_j(i)| \leq |2\lambda a_i(j)^2 x_t(j)| + |a_j(i)| \leq 2|a_j(i)| = 2|H_{ij}|.$$

Let  $K = \text{diag}(H^\top H)$ . Then, using Lemma 9 with  $\alpha = 2$  and  $\beta = 1/10$ , we get that there is a subspace  $\mathcal{G}_t$  with  $\dim(\mathcal{G}_t) \geq \lceil 9n_t/10 \rceil$  such that  $\mathcal{G}_t = \{w \in \mathcal{W}_t : w^\top G^\top G w \leq 40 \cdot w^\top K w\}$ , which by the definition of  $G$  and  $H$  is equivalent to that given by (11).

Putting together the bounds on the dimensions of these subspaces gives,

$$\dim(\mathcal{Z}_t) \geq \dim(\mathcal{U}_t \cap \mathcal{Y}_t \cap \mathcal{G}_t) \geq \lceil 11n_t/12 \rceil - 1 + \lceil 9n_t/10 \rceil + \lceil 9n_t/10 \rceil - 2n_t \geq \lceil 2n_t/3 \rceil. \blacktriangleleft$$

**Proof of Theorem 1.** Recall that we divide each row  $a$  of  $A$  as  $a = a^S + a^L$ . We will bound  $\langle a^L, x_T \rangle$  and  $\langle a^S, x_T \rangle$  separately.

Let  $t_1$  denote the earliest when the squared norm of  $a^L$  (restricted to the alive variables) is at most 20, and let  $n_1$  be number of non-zeros in  $a^L$  restricted to the set  $\mathcal{V}_{t_1}$ . As  $|a^L(j)| \geq 1/(2\lambda)$  for each  $j$ , the number of non-zero variables  $n_1$  in  $a^L$  at time  $t_1$  is at most  $80\lambda^2$ , as  $n_1/(4\lambda^2) \leq \sum_{j \in \mathcal{V}_{t_1}} a^L(j)^2 \leq 20$ . Moreover, as  $a^L$  incurs zero discrepancy until  $t_1$ , the overall discrepancy satisfies

$$|\langle a^L, x_T \rangle| = |\langle a^L, x_{t_1} \rangle| + |\langle a^L, x_T - x_{t_1} \rangle| \leq \sqrt{n_1} \cdot \left( \sum_{j \in \mathcal{V}_{t_1}} a^L(j)^2 \right)^{1/2} \leq 80\lambda \leq 3b_0. \quad (31)$$

Henceforth, we focus on the rows  $a^S$ . We first show that the slacks are always positive. Let  $\gamma = b_0/4(4m)^{\frac{1}{p}}$ . By Lemma 11, for all  $t \in [T]$ ,  $\Phi(t) \leq 4m(2/b_0)^p < \gamma^{-p}$ . This implies that  $|s_i(t)| \geq \gamma$  for all  $i \in \mathcal{I}_t^S$  and  $t \in [T]$ . In one step of the algorithm,

$$\begin{aligned} |s_i(t) - s_i(t-1)| &\leq \delta^2 d_{t-1} + |\langle a_i, x_t \rangle - \langle a_i, x_{t-1} \rangle| \\ &\leq \delta^2 d_{t-1} + |\delta \langle a_i, v_{t-1} \rangle| \leq 20n\delta \leq 2\gamma. \end{aligned}$$

So, if  $s_i(t-1) \geq \gamma$  and  $\Phi(t) < \gamma^{-p}$ , then  $s_i(t) \geq 0$ , i.e., the slack  $s_i(t)$  cannot go from being greater than  $\gamma$  to less than  $-\gamma$  in a single step. So, for every  $i \in \mathcal{I}^S$  and  $t \in [T]$ ,  $s_i(t) \geq \gamma$  and  $\langle a_i, x_T \rangle \leq b_T$ . Together with (31) this gives,  $|\langle a, x_T \rangle| \leq |\langle a^S, x_T \rangle| + |\langle a^L, x_T \rangle| \leq b_T + 3b_0$ .

Let  $x \in \{-1, 1\}^n$  be obtained from  $x_T$  by the rounding  $x(j) = \text{sign}(x_T(j))$ . As  $T = (n-2)/\delta^2$ ,  $\|x_T\|^2 = n-2$  with  $|x_T(j)| \leq 1$  for all  $j \in [n]$ . After rounding  $x_T$  to  $x$ , we have  $\|x\|^2 = n$ . For any row  $a$  of  $A$ , the discrepancy is bounded by

$$|\langle a, x \rangle| = |\langle a, x_T \rangle| + |\langle a, x - x_T \rangle| \leq |\langle a, x_T \rangle| + M \sum_{j=1}^n |x(j) - x_T(j)| \leq b_T + 3b_0 + 2M.$$

We now consider the two cases for  $b_0$ ,  $d_t$ ,  $p$ . If the second case given by (14), then by (30),  $b_0 \leq 1680(p+1) \cdot (48m/n_t)^{1/p}/b_0$ . As  $n_t \geq 1$  for all  $t \in [T]$  and  $p = \log(2m)$ , we have  $(48m/n_t)^{1/p} \leq 10e$ , and setting  $b_0 = 250\sqrt{\log(2m)}$  suffices. Since  $d_t = 0$ ,  $b_T = b_0$  and  $\|Ax\|_\infty \leq 4b_0 + 2M$ .

In the first case given by (13), then by (30), we have  $d_t = 8(p+1)(48m)^{\frac{1}{p}} \cdot \frac{h(n_t)}{b_0 n_t^{1/p}}$  for all  $t \in [T]$ . Summing  $d_t$  over  $t$  gives

$$b_T - b_0 = \delta^2 \sum_{t=0}^{T-1} d_t = 8(p+1)(48m)^{\frac{1}{p}} \delta^2 \cdot \sum_{t=0}^{T-1} h(n_t)/(b_0 n_t^{1/p}).$$

As  $n_t > n - \delta^2 t - 1 \geq$  and  $h$  is non-increasing,  $\delta^2 \cdot \sum_{t=0}^{T-1} h(n_t) n_t^{-1/p} \leq \beta$ , so that  $b_T \leq b_0 + 8(p+1)(48m)^{1/p} \beta / b_0$ . Optimizing  $b_0 = (8(p+1)(48m)^{1/p} \beta)^{1/2}$  gives that  $b_T = 2b_0$  and thus  $\|Ax\|_\infty \leq b_T + 3b_0 + 2M \leq 5b_0 + 2M$ , giving the desired result.  $\blacktriangleleft$

**Proof of Corollary 8.** For a constant  $h$ , we have  $\beta = \int_0^{n-2} (n-t)^{-1/p} h dt \leq n^{1-1/p} h / (1-1/p)$ . Choosing  $p = \log(2m/n)$  to optimize the first term in (4) gives the result.  $\blacktriangleleft$

## A.1 Sub-Gaussian Matrices and Random Matrices

Let  $X$  be a random variable with  $\mathbb{E}(X) = 0$ .  $X$  is called Sub-Gaussian with variance  $\sigma^2$  if its moment generating function satisfies  $\mathbb{E}(e^{sX}) \leq e^{\sigma^2 s^2/2}$  for all  $s \in \mathbb{R}$ . For a Sub-Gaussian random variable,  $\mathbb{E}(X^2) \leq 4\sigma^2$ .

**Proof of Theorem 2.** As  $a_i(j)$  is a Sub-Gaussian with variance  $\sigma^2$ ,  $a_i(j)^2 - \mathbb{E}(a_i(j)^2)$  is a mean zero and sub-exponential random variable with parameter  $16\sigma^2$  [36].

For any  $S \subseteq [n]$  with  $|S| = s$ , Bernstein's inequality for sub-exponential random variables [36] (Theorem 2.8.1) gives that,

$$\Pr\left(\sum_{j \in S} a_i(j)^2 - \mathbb{E}(a_i(j)^2) \geq st\right) \leq \exp(-\min(s^2 t^2 / 16\sigma^4, st / 16\sigma^2)). \quad (32)$$

Setting  $t = 96\sigma^2 (\log(ne/s) + (\log m)/s)$  and as  $\mathbb{E}(a_i(j)^2) \leq 4\sigma^2$ , and taking a union bound over all the rows and all possible subsets of  $s$  columns, we get that,

$$\sum_{j \in S} a_i^2(j) \leq 100\sigma^2 |S| (\log(ne/|S|) + \log m / |S|). \quad (33)$$

for every  $S \subseteq [n]$ ,  $i \in [m]$ , with probability at least  $1 - 1/2m^2$ .

Similarly, as  $a_i(j)$  is sub-Gaussian with mean 0 and variance  $\sigma^2$ , with probability at least  $1 - 1/2m^2$ , we have  $|a_i(j)| \leq 3\sigma\sqrt{\log(mn)}$  for all  $i \in [m], j \in [n]$ , and thus the  $\ell_2$ -norm of a column is at most  $L = 3\sqrt{m}\sigma\sqrt{\log(mn)}$  and  $M = 3\sigma\sqrt{\log mn}$ . By (33), we can set

$$h(t) = 100\sigma^2 \left( \log\left(\frac{ne}{t}\right) + \frac{\log m}{t} \right).$$

A direct computation gives  $\beta = \int_0^{n-2} h(n-t)(n-t)^{-1/p} dt = O(\sigma^2(n^{1-1/p} + p \log m))$ . Using Theorem 1 with  $p = 2\lceil \log(2m/n) \rceil$ , gives  $b_0 = O(\sigma(p(m/n)^{1/p}(n + n^{1/p}p \log m))^{1/2}) = O(\sigma n^{1/2} \log(2m/n))$ .

Thus, with high probability  $\|Ax\|_\infty \leq (5b_0 + 2M) = O(\sigma\sqrt{n \log(2m/n)})$ .  $\blacktriangleleft$

**Proof of Theorem 3.** Consider a random vector  $X$  chosen uniformly at random from the unit ball,  $\{x \in \mathbb{R}^m : \|x\|_2 \leq 1\}$ . Then every coordinate of  $X$  is sub-Gaussian with variance  $\sigma^2 = C/\sqrt{m}$ , where  $C$  is a constant [36] (Theorem 3.4.6, Ex 3.4.7). The result now follows from Theorem 5.  $\blacktriangleleft$

## A.2 Subadditive Stochastic Discrepancy

**Proof of Theorem 4.** Let  $\Phi_1(t)$ ,  $\Phi_2(t)$  be the potential functions corresponding to  $A$  and  $B$ , respectively. Let the parameters for Algorithm 2 on  $A$  be  $b_0^1, p_1, d_t^1, h_1(\cdot)$  and for  $B$  be  $b_0^2, p_2, d_t^2, h_2(\cdot)$ .

Note that it might not be possible to select an update  $v_t$  at time  $t$ , that ensures that both  $\Phi_1(t+1) \leq \Phi_1(t)$  and  $\Phi_2(t+1) \leq \Phi_2(t)$  hold, but we can find a  $v_t$  for which a weighted sum of  $\Phi_1(t)$  and  $\Phi_2(t)$  decreases at every step.

Consider the potential function  $\Phi(t) = (b_0^1/2)^{p_1} \Phi_1(t) + (b_0^2/2)^{p_2} \Phi_2(t)$ . We apply the same algorithmic framework. For  $t = 1, \dots, T$ , select  $v_t$  such that  $\mathbb{E}(\Phi(t+1)) \leq \Phi(t)$ , and select the sign of  $\varepsilon$  for which  $\Phi(t+1) \leq \Phi(t)$ , and set  $x_{t+1} = x_t + \varepsilon \delta v_t$ . To this end, it suffices to find a  $v_t$  such that  $\mathbb{E}(\Phi_1(t+1)) \leq \Phi_1(t)$  and  $\mathbb{E}(\Phi_2(t+1)) \leq \Phi_2(t)$ .

Let  $\mathcal{Z}_t^1$  and  $\mathcal{Z}_t^2$  be the feasible subspaces at step  $t$  for  $A$  and  $B$  respectively from Algorithm 2. We will search for  $v_t$  in  $\mathcal{Z}_t = \mathcal{Z}_t^1 \cap \mathcal{Z}_t^2$ . By Lemma 10,  $\dim(\mathcal{Z}_t^1), \dim(\mathcal{Z}_t^2) \geq \lceil 2n_t/3 \rceil$ . Therefore,  $\dim(\mathcal{Z}_t) = \dim(\mathcal{Z}_t^1 \cap \mathcal{Z}_t^2) \geq \lceil 2n_t/3 \rceil + \lceil 2n_t/3 \rceil - n_t \geq n_t/3$ .

Using Lemma 13 on  $A$  and  $B$ , along with Markov's inequality implies that there exists a vector  $w \in \mathcal{Z}_t$  such that

$$\sum_{i \in \mathcal{I}_t^1} \frac{\langle 2cb_0^1 e_{t,i} - a_i, w \rangle^2}{s_i(t)^{p_1+1}} \leq \sum_{i \in \mathcal{I}_t^1} \frac{25h_1(n_t)}{s_i(t)^{p_1+1}} \quad \text{and} \quad \sum_{i \in \mathcal{I}_t^2} \frac{\langle 2cb_0^2 e_{t,i} - a_i, w \rangle^2}{s_i(t)^{p_2+1}} \leq \sum_{i \in \mathcal{I}_t^2} \frac{25h_2(n_t)}{s_i(t)^{p_2+1}}.$$

(34)

Comparing (34) with (15), the functions  $h_1(\cdot)$  and  $h_2(\cdot)$  only increase by a constant factor when compared to running Algorithm 2 on  $A$  and  $B$  independently. So it suffices to multiply  $d_t^1$  and  $d_t^2$  by 4 to ensure that by Lemma 11,

$$\mathbb{E}[\Phi_1(t)] - \Phi_1(t-1) \leq \frac{1}{Tn(b_0^1)^{p_1}} \quad \text{and} \quad \mathbb{E}[\Phi_2(t)] - \Phi_2(t-1) \leq \frac{1}{Tn(b_0^2)^{p_2}}. \quad (35)$$

Plugging (35) in the definition of  $\Phi(t)$ , we get  $\mathbb{E}[\Phi(t)] - \Phi(t-1) \leq 2/(Tn)$ . So one of the two choices of  $x_t$  gives  $\Phi(t) - \Phi(t-1) \leq 2/(Tn)$ . Summing over  $t$ ,

$$\Phi(t) \leq \Phi(0) + \frac{2}{n} \leq \left(\frac{b_0^1}{2}\right)^{p_1} \Phi_1(0) + \left(\frac{b_0^2}{2}\right)^{p_2} \Phi_2(0) + \frac{2}{n}.$$

By Lemma 19,  $\Phi_1(0) \leq 2m \cdot (2/b_0^1)^{p_1}$  and  $\Phi_2(0) \leq 2m \cdot (2/b_0^2)^{p_2}$ , thus  $\Phi(t) \leq \Phi(0) + 2/n \leq 5m$ . For a row  $i \in \mathcal{J}_t^\ell$  for  $\ell \in \{1, 2\}$ , we have  $(\lfloor n_t/12 \rfloor + 1) \cdot (b_0^\ell/2)^{p_\ell} \cdot s_i(t)^{-p_\ell} \leq \Phi(t) \leq 5m$ , which implies that for any  $t$ , and  $\ell \in \{1, 2\}$ ,

$$\max_{i \in \mathcal{J}_t^\ell} s_i(t)^{-1} \leq \frac{2}{b_0^\ell} \left(\frac{60m}{n_t}\right)^{\frac{1}{p_\ell}}. \quad (36)$$

Upon comparing (36) with (30), notice that  $\max_{k \in \mathcal{J}_t^1} s_k(t)^{-1}$  and  $\max_{k \in \mathcal{J}_t^2} s_k(t)^{-1}$  are only a constant factor larger when compared to running Algorithm 2 on  $A$  and  $B$  separately, and hence the discrepancies for both  $A$  and  $B$  are only a constant factor larger. ◀

## B Appendix: Bounding the step size

► **Lemma 19.** For  $A \in \mathbb{R}^{m \times n}$ ,

- $\Phi(0) + \sum_t |\mathcal{I}_{t+1} \setminus \mathcal{I}_t| \cdot \left(\frac{2}{b_0}\right)^p \leq 2m \cdot \left(\frac{2}{b_0}\right)^p.$
- For all  $t \in \{0, 1, \dots, T-1\}$ , if  $\Phi(t) \leq 2^7 m^2 \left(\frac{2}{b_0}\right)^p$  and  $d_t = O(pn \cdot \max_{i \in \mathcal{J}_t} s_i(t)^{-1})$ , then for step size  $\delta^2 \leq (Cn^2 m^6 p^4)^{-1}$ ,

$$\mathbb{E}(\Phi(t+1)) - \Phi(t) \leq f(t) + \frac{1}{Tnb_0^p} + |\mathcal{I}_{t+1} \setminus \mathcal{I}_t| \cdot \left(\frac{2}{b_0}\right)^p, \text{ where}$$

$$f(t) = -p\delta^2 \sum_{i \in \mathcal{I}_t} \frac{d_t + cb_0 \langle a_i^{(2)}, v_t^{(2)} \rangle}{s_i(t)^{p+1}} + \frac{p(p+1)\delta^2}{2} \sum_{i \in \mathcal{I}_t} \frac{\langle 2cb_0 e_{t,i} - a_i, v_t \rangle^2}{s_i(t)^{p+2}}.$$

# Fourier Growth of Regular Branching Programs

Chin Ho Lee ✉

Harvard University, Cambridge, MA, USA

Edward Pyne ✉

Harvard University, Cambridge, MA, USA

Salil Vadhan ✉

Harvard University, Cambridge, MA, USA

---

## Abstract

We analyze the *Fourier growth*, i.e. the  $L_1$  Fourier weight at level  $k$  (denoted  $L_{1,k}$ ), of read-once regular branching programs. We prove that every read-once regular branching program  $B$  of width  $w \in [1, \infty]$  with  $s$  accepting states on  $n$ -bit inputs must have its  $L_{1,k}$  bounded by

$$\min \left\{ \Pr[B(U_n) = 1] (w-1)^k, s \cdot O((n \log n)/k)^{\frac{k-1}{2}} \right\}.$$

For any constant  $k$ , our result is tight up to constant factors for the AND function on  $w-1$  bits, and is tight up to polylogarithmic factors for unbounded width programs. In particular, for  $k=1$  we have  $L_{1,1}(B) \leq s$ , with no dependence on the width  $w$  of the program.

Our result gives new bounds on the coin problem and new pseudorandom generators (PRGs). Furthermore, we obtain an explicit generator for unordered permutation branching programs of unbounded width with a constant factor stretch, where no PRG was previously known.

Applying a composition theorem of Blasiok, Ivanov, Jin, Lee, Servedio and Viola (RANDOM 2021), we extend our results to “generalized group products,” a generalization of modular sums and product tests.

**2012 ACM Subject Classification** Theory of computation → Pseudorandomness and derandomization

**Keywords and phrases** pseudorandomness, fourier analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.2

**Category** RANDOM

**Related Version** Preprint: <https://eccc.weizmann.ac.il/report/2022/034/>

**Funding** *Chin Ho Lee*: Supported by NSF grant CCF-1763299 and a Simons Investigator grant to S. Vadhan.

*Salil Vadhan*: Supported by NSF grant CCF-1763299 and a Simons Investigator grant.

**Acknowledgements** We thank the anonymous reviewers for their helpful comments.

## 1 Introduction

Every Boolean function  $f: \{-1, 1\}^n \rightarrow \{0, 1\}$  can be identified by its unique multilinear extension

$$f(x) := \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i,$$

where the coefficients

$$\hat{f}(S) := \mathbf{E}_{x \sim \{-1, 1\}^n} \left[ f(x) \prod_{i \in S} x_i \right]$$



© Chin Ho Lee, Edward Pyne, and Salil Vadhan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 2; pp. 2:1–2:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



are called the *Fourier coefficients* of  $f$ . Over the past few decades, the analysis of these coefficients of Boolean functions has become an indispensable tool in theoretical computer science and mathematics. We refer the readers to the excellent textbook by O’Donnell [46] for a broad introduction.

Given the wide applicability of this tool, researchers have proposed and analyzed different quantitative measures of Fourier coefficients of Boolean functions. In this work we focus on the  $L_1$  Fourier norm at level  $k$ :

► **Definition 1** ( $L_1$  Fourier norm at level  $k$ ). *The  $L_1$  Fourier norm of a function  $\{-1, 1\}^n \rightarrow \{0, 1\}$  at level  $k$  is*

$$L_{1,k}(f) := \sum_{S \subseteq [n]: |S|=k} |\hat{f}(S)|.$$

For a function class  $\mathcal{F}$ , we use  $L_{1,k}(\mathcal{F})$  to denote  $\max_{f \in \mathcal{F}} L_{1,k}(f)$ .

The notion of *Fourier growth* is a convenient way of capturing the growth of  $L_{1,k}$  with respect to levels  $k$ .

► **Definition 2** (Fourier growth). *A function class  $\mathcal{F} \subseteq \{f: \{-1, 1\}^n \rightarrow \{0, 1\}\}$  has Fourier growth  $L_1(a, b)$  if  $L_{1,k}(\mathcal{F}) \leq a \cdot b^k$  for every  $k$ .*

By the Cauchy–Schwarz inequality, every Boolean function has its  $L_{1,k}$  bounded by  $\binom{n}{k}^{1/2}$ , and thus has Fourier growth  $L_1(1, \sqrt{n})$ .

Fourier growth was first studied by Mansour to obtain sample-efficient algorithms for learning DNFs [42]. It was later formally introduced by Reingold, Steinke and Vadhan in [51], where they constructed explicit unconditional pseudorandom generators for permutation branching programs. Subsequently, this notion has led to many exciting developments in learning theory [36, 25] and pseudorandomness [19, 16, 26, 18, 15]. In recent years researchers have also discovered new applications to other areas such as separating quantum and classical computation [50, 59, 5, 54, 27], and proving correlation bounds with the Majority function (and its variants) [17, 15, 61].

Thus given a function class, it has now become a natural question to analyze its Fourier growth. Indeed, in the past decade it has been shown that several well-studied classes of functions exhibit bounded Fourier growth. These include (parity) decision trees [47, 7, 59, 54, 28], constant-depth circuits [42, 58], subclasses of low-degree  $\mathbb{F}_2$ -polynomials [16, 28, 15], low-degree real polynomials [36, 25], functions with bounded sensitivity [32], product tests [37], and read-once branching programs [51, 57, 19].

Motivated by derandomization of space-bounded algorithms, in this work we continue the line of research on the Fourier growth of *read-once branching programs*.

► **Definition 3** (Read-once branching programs). *An (unordered) read-once branching program  $B$  of length  $n$  and width  $w$  computes a function  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$ . On input  $x \in \{-1, 1\}^n$ , the program  $B$  fixes a permutation  $\pi: [n] \rightarrow [n]$  and computes as follows. It starts at a fixed start state  $v_1 \in [w]$ . Then for  $t = 1, \dots, n$ , it reads the next input bit  $x_{\pi(t)}$  and updates its state according to a transition function  $B_t: [w] \times \{-1, 1\} \rightarrow [w]$  by taking  $v_{t+1} := B_t(v_t, x_{\pi(t)})$ . Note that the transition function  $B_t$  can differ at each time step. The program has a fixed set of accept states  $V_{\text{acc}} \subseteq [w]$ , and  $B(x) = \mathbb{1}(v_{n+1} \in V_{\text{acc}})$ .*

For a branching program  $B$  of length  $n$  and width  $w$ , we will view it as a directed layered graph with  $n + 1$  layers of vertices denoted by  $V_1, \dots, V_{n+1}$ , each consists of  $w$  vertices. For every two consecutive layers  $V_t$  and  $V_{t+1}$ , every vertex  $u \in V_t$  has two outgoing edges labeled by  $b \in \{-1, 1\}$ , where the  $b$ -edge goes to the vertex  $B_t(u, b)$  in  $V_{t+1}$ .



As we will not consider non-read-once branching programs in this work, henceforth we will often omit the word “read-once” and use “branching programs” to refer to read-once branching programs for simplicity. Furthermore, as the Fourier growth of a function is unaffected by reordering the input bits, for the purpose of establishing  $L_{1,k}$  bounds we can restrict our attention to the case where  $\pi$  is the identity permutation.

A well-studied subclass of branching programs is the class of *regular branching programs*. This model has received a lot of attention in the literature [52, 21, 56, 13, 51, 9], in part due to the fact that pseudorandomness against this restricted subclass sometimes suffices for pseudorandomness against general branching programs, and hence the derandomization of space-bounded computation [52, 9].

► **Definition 4** (Read-once regular branching programs). *A read-once regular branching program is a read-once branching program where for every time step  $t$  and state  $v \in [w]$ , there are exactly 2 pairs  $(u, b) \in [w] \times \{-1, 1\}$  such that  $B_t(u, b) = v$ .*

Note that the underlying graph of a regular branching program forms a regular directed layered graph. A more restricted class that has also been well-studied is the class of *permutation branching programs*, where in addition to being a regular graph, the  $(-1)$ -edges and  $1$ -edges between every two adjacent layers in the graph give rise to two permutations on  $[w]$ .

► **Definition 5** (Read-once permutation branching programs). *A read-once permutation branching program is a read-once regular branching program where for every time step  $t$  and pair of states  $(u, u')$ , if  $B_t(u, b) = B_t(u', b')$  then either  $u = u'$  or  $b \neq b'$ .*

A recent line of works constructed explicit pseudorandom objects for regular and permutation branching programs of *unbounded* width with a bounded number of accept states<sup>1</sup> [34, 48, 49, 9], a model for which prior to these works even non-explicit constructions were not known to exist. Motivated by these results, we investigate the Fourier growth of these same models.

## 1.1 Our results

We obtain near-optimal  $L_{1,k}$  bounds for regular branching programs of *any* width, improving the bounds in [51] and obtaining the first non-trivial bounds for unbounded width programs.

► **Theorem 6.** *Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a regular branching program of width  $w \in [1, \infty]$  with  $s$  accept states in its final layer. Then*

$$L_{1,k}(B) \leq \min \left\{ \underbrace{\Pr[B(U_n) = 1]}_1 \cdot (w-1)^k, \underbrace{s \cdot O((n \log n)/k)^{\frac{k-1}{2}}}_2 \right\}.$$

Note that the two bounds are incomparable: the first bound is independent of the input length  $n$ , and the second bound is independent of the width  $w$ . The first bound is tight for the  $\text{AND}_{w-1}$  function on  $w-1$  bits, which can be computed by a width- $w$  permutation branching program, since

$$L_{1,k}(\text{AND}_{w-1}) = 2^{-(w-1)} \cdot \binom{w-1}{k} = \Pr[\text{AND}_{w-1}(U_{w-1}) = 1] \cdot \binom{w-1}{k}.$$

<sup>1</sup> Note that unbounded width permutation programs with an unbounded number of accept states can compute arbitrary Boolean functions.

For  $k = 1$ , our second bound can be sharpened to  $s \cdot \Pr[B(U_n) = 0]$  (see Theorem 21). We complement Theorem 6 by a lower bound showing that our second upper bound is in fact tight up to a factor of  $\Theta_k(1) \cdot (\log n)^{\frac{k-1}{2}}$  for  $k \geq 2$ , even for the restricted subclass of permutation branching programs.

► **Proposition 7.** *For all positive integers  $k, n$ , and  $s$  where  $s \leq \sqrt{kn}$ , there exists a permutation branching program  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  of width  $\Theta(\sqrt{kn})$  with  $s$  accept states such that  $L_{1,k}(B) \geq \frac{s}{\sqrt{kn}} \cdot \Omega(n/k)^{k/2} = \Omega_k(1) \cdot s \cdot n^{\frac{k-1}{2}}$ .*

We now make some remarks on Theorem 6. Previously, Reingold, Steinke and Vadhan proved an upper bound of  $(2w^2)^k$  [51]. Hence, our first upper bound improves their bound on two fronts. Our first improvement is a quadratic sharpening on the dependence on the width  $w$ . Our second improvement is the additional acceptance probability factor in our bounds, which, as we will discuss in the next section, has further implications.  $L_{1,k}$  bounds with a dependence on the acceptance probability have proved to be useful, both in extending the bounds to higher levels  $k' > k$  [19] and extending the bounds to other classes of tests [37, 8]. Indeed, we obtain both our  $L_{1,k}$  bounds for  $k > 1$  by applying the reduction in [19] to bounds at a lower level, and this reduction requires obtaining an  $L_{1,k}$  bound that scales linearly with respect to the acceptance probability of the function. We note that functions admitting  $L_{1,k}$  bounds that scale linearly with acceptance probability include arbitrary Boolean functions [46, 37], constant-width read-once branching programs [19],  $\mathbb{F}_2$ -polynomials [28, 8], and product tests with outputs  $\{-1, 1\}$  [37, 8]. Therefore, Theorem 6 adds the class of regular branching programs to this list.

Our second upper bound gives the first non-trivial  $L_{1,k}$  bounds for regular branching programs of unbounded width. Recall that every bounded function has its  $L_{1,k}$  bounded by  $\sqrt{\binom{n}{k}}$ ; so this upper bound is interesting only when  $s = o(\sqrt{n/(k(\log n)^{k-1})})$ .

Proposition 7 follows from the observation that symmetric  $\mathbb{F}_2$ -polynomials of degree  $w$  can be computed by a permutation branching program of width at most  $2w$  [6], where  $L_{1,k}$  lower bounds on the former class were recently established in [8]. For the same reason, Theorem 6 recovers the  $L_{1,k}$  bounds for symmetric  $\mathbb{F}_2$ -polynomials in [8, Theorem 8] with a different proof.

## 1.2 Applications

We describe several consequences of Theorem 6.

### 1.2.1 Coin problem

Let  $X_\delta = (X_1, \dots, X_n)$  be the distribution over  $\{-1, 1\}^n$ , where the  $X_i$ 's are independent and each  $X_i$  has expectation  $\delta$ . The  $\delta$ -coin problem studies the maximum advantage for a function class  $\mathcal{F}$  to distinguish between the distributions  $X_\delta$  and  $X_0 = U_n$ . This basic problem has been studied extensively for various restricted classes of tests, and has a wide range of applications in computational complexity, including circuit complexity [4, 60, 53, 40, 29], pseudorandom generators [14], quantum computing [1, 2], streaming algorithms [11], and multiparty computation [20]. In particular, there has been a rich line of work on the coin problems for branching programs [14, 55, 38, 11, 12].

It is known that bounds on the Fourier growth of  $\mathcal{F}$  imply bounds on the distinguishing advantage for the coin problem of functions in  $\mathcal{F}$  (see [37, Fact 9]). Thus we obtain the following corollary of Theorem 6.

► **Corollary 8.** *There exists a constant  $\alpha > 0$  such that the following holds. Let  $B: \{-1, 1\} \rightarrow \{0, 1\}$  be a regular branching program of width  $w \in [1, \infty]$  with  $s$  accept states. For every  $\delta \leq \alpha \max\{1/w, 1/\sqrt{n \log n}\}$ , we have*

$$|\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(X_0)]| \leq \delta s + \delta^2 \cdot O\left(\min\{w^2, s\sqrt{n \log n}\}\right).$$

Moreover, Avishay Tal showed (see [3, Lemma 9]) that if a class  $\mathcal{F}$  is closed under restrictions, then  $L_{1,1}$  bounds on  $\mathcal{F}$  already implies bounds on the coin problem for  $\mathcal{F}$ . Since the class of permutation branching programs is closed under restrictions, we obtain the following stronger coin problem bounds for that class:

► **Corollary 9.** *Let  $B: \{-1, 1\} \rightarrow \{0, 1\}$  be a permutation branching program with  $s$  accept states. Then  $|\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(X_0)]| \leq \frac{\delta}{1-\delta} \cdot s$ .*

▷ **Claim 10.** For every  $\delta > 0$  and positive integer  $s \leq 32/\delta$ , there exists a permutation branching program  $B$  of length  $32/\delta^2$  and width  $128/\delta$  with  $s$  accept states such that  $\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(X_0)] \geq \frac{s\delta}{1000}$ .

Corollaries 8 and 9 can be interpreted as follows. Regular (and permutation) programs with a single accept state cannot distinguish (sufficiently small) biased coins from uniform much better than simply outputting their first input bit.

Previously Braverman, Rao, Raz, and Yehudayoff [13] obtained a coin problem bound of  $\delta \cdot s \cdot (w - 1)$  for width- $w$  regular branching programs with  $s$  accept states. Corollaries 8 and 9 improve this to roughly  $\delta \cdot s$  when  $\delta$  is very small (Corollary 8) or when we restrict to permutation branching programs (Corollary 9). Claim 10 shows that the upper bound in Corollary 9 is tight up to constant factors.

## 1.2.2 Pseudorandom generators

Theorem 6 also implies new pseudorandom generators for permutation branching programs.

► **Definition 11** (Pseudorandom generators). *A function  $G: \{0, 1\}^s \rightarrow \{-1, 1\}^n$  is a pseudorandom generator (PRG) for a function class  $\mathcal{F}$  with seed length  $s$  and error  $\epsilon$ , if for every  $f \in \mathcal{F}$ ,*

$$|\mathbf{E}[f(U_n)] - \mathbf{E}[f(G(U_s))]| \leq \epsilon.$$

$G$  is explicit if it can be computed in polynomial time.

Recall that we consider unordered branching programs, where a program can read its inputs in arbitrary order before its execution. Starting from the work of Bogdanov, Papakonstantinou, and Wan [10], there has been extensive research on constructing pseudorandom generators for unordered branching programs [10, 35, 51, 57, 33, 39, 19, 43, 26, 22, 37, 23, 24], in search for new ideas for improving Nisan’s PRG for ordered branching programs [45], which remains the best PRG for derandomizing space-bounded computation to date. This line of research recently led to the first improvement over Nisan’s PRG for the special case of width-3 (ordered) branching programs [43].

Applying our  $L_{1,k}$  bounds to the “polarizing random walk” framework of [16, 18, 15], we obtain the following pseudorandom generator.

► **Corollary 12.** *There is an explicit pseudorandom generator for width- $w$  permutation branching programs with seed length  $w^2 \cdot O(\log(n/\epsilon))(\log(1/\epsilon) + \log \log n)$  and error  $\epsilon$ .*

Corollary 12 gives a slight improvement on the PRG given by [16], reducing the dependence on width from  $w^4$  to  $w^2$ , stemming directly from the  $L_{1,k}(B) \leq (w-1)^k$  bound in Theorem 6, which improves the  $L_{1,k}(B) \leq (2w^2)^k$  bound of [51]. (Corollary 12 is for permutation branching programs rather than regular branching programs, because the polarizing random walk framework requires that the class is closed under restriction). By the reduction of [9], this also implies a *hitting set generator* (HSG) for permutation branching programs of *unbounded* width with seed length  $O(1/\epsilon^2) \cdot \log(n/\epsilon)(\log(1/\epsilon) + \log \log n)$ , quadratically improving the dependence on  $\epsilon$ . (An  $\epsilon$ -HSG for a class  $\mathcal{F}$  is a function  $G: \{0,1\}^s \rightarrow \{-1,1\}^n$  where for all  $f \in \mathcal{F}$  with  $\Pr[f(U_n) = 1] > \epsilon$  there is an  $x \in \{0,1\}^s$  such that  $f(G(x)) = 1$ .)

From Corollary 9, we also obtain the first nontrivial pseudorandom generator that fools unordered permutation branching programs of unbounded width with constant factor stretch and constant error.<sup>2</sup> For simplicity we state our result for constant error, and do not optimize constants.

Let  $H(x) := x \log(\frac{1}{x}) + (1-x) \log(\frac{1}{1-x})$  denote the binary entropy function.

► **Theorem 13.** *Given any constant  $\delta \in (0, 1/2)$  independent of  $n$ , there is an explicit PRG for unordered permutation branching programs with a single accept state with seed length  $H(1/2 + 0.499\delta) \cdot n + o(n)$  and error  $\frac{\delta}{1-\delta} + \frac{\delta}{100}$ .*

This is proven by noting that with the specified seed length, we can approximately sample  $n$  independent  $\delta$ -biased coins, which are pseudorandom by Corollary 9. We are not aware of any PRGs prior to our result.

As mentioned above, there exist explicit hitting-set generators (HSGs) with better seed length for this class [9]. For the easier case of *ordered* permutation programs, Hoza, Pyne, and Vadhan [34] constructed an explicit PRG with significantly better seed length, namely  $\tilde{O}(\log n \cdot \log(1/\epsilon))$ .

We note that our results do not give any PRGs for regular programs, because all of the methods for obtaining PRGs from Fourier growth bounds require the class to be closed under restrictions. In particular, even in the ordered setting, it remains unknown whether a nontrivial PRG for unbounded width regular programs exists.

### 1.2.3 Generalized group products

As mentioned in the previous section,  $L_{1,k}$  bounds with the acceptance probability factor (as in Theorem 6) are useful for obtaining  $L_{1,k}$  bounds for wider function classes. To make this precise, we recall the definition of *disjoint composition* of two function classes.

► **Definition 14** (Disjoint composition). *Let  $\mathcal{F}$  be a class of functions from  $\{-1,1\}^m$  to  $\{-1,1\}$  and let  $\mathcal{G}$  be a class of functions from  $\{-1,1\}^\ell$  to  $\{-1,1\}$ . Define the class  $\mathcal{F} \circ \mathcal{G}$  of disjoint composition of  $\mathcal{F}$  and  $\mathcal{G}$  to be the class of all functions from  $\{-1,1\}^{m\ell}$  to  $\{-1,1\}$  of the form*

$$h(x^1, \dots, x^m) = f(g_1(x^1), \dots, g_m(x^m)),$$

where  $g_1, \dots, g_m \in \mathcal{G}$  are defined on  $m$  disjoint sets of variables and  $f \in \mathcal{F}$ .

Błasiok, Ivanov, Jin, Lee, Servedio and Viola [8] proved a composition theorem showing that if both  $\mathcal{F}$  and  $\mathcal{G}$  are closed under negation of their outputs, and  $\mathcal{F}$  is closed under restrictions, then  $L_{1,k}$  bounds with the acceptance probability factor for  $\mathcal{F}$  and  $\mathcal{G}$  imply  $L_{1,k}$

<sup>2</sup> The co-HSG of [9] can be interpreted as an explicit PRG for permutation programs with error  $1 - 1/(n+1)$ .

bounds on the disjoint composition of  $\mathcal{F}$  and  $\mathcal{G}$ . Specifically, if for every  $1 \leq k \leq K$ , we have  $L_{1,k}(f) \leq \Pr[f(U_m) = 1] \cdot b_{\text{outer}}^k$  for every  $f \in \mathcal{F}$  and  $L_{1,k}(g) \leq \Pr[g(U_\ell) = 1] \cdot b_{\text{inner}}^k$  for every  $g \in \mathcal{G}$ , then for every function  $h \in \mathcal{F} \circ \mathcal{G}$ , we have that

$$L_{1,K}(h) \leq \Pr[h(U_{m\ell}) = 1] \cdot (b_{\text{inner}} b_{\text{outer}})^K.$$

Therefore, we also obtain new  $L_{1,k}$  bounds for the disjoint composition of permutation branching programs and other classes of functions that admit the acceptance probability factor in their  $L_{1,k}$  bounds (see Section 1 for a list). As a concrete example of such composition, we introduce the class of *generalized group products*.

► **Definition 15** (Generalized group products). *A function  $f: \{-1, 1\}^n \rightarrow \{0, 1\}$  is a  $(m, \ell, G)$ -group product if there exist  $m$  disjoint subsets  $I_1, \dots, I_m \subseteq [n]$  of size at most  $\ell$  such that*

$$f(x) = \mathbb{1} \left( \prod_{i=1}^m g_i^{f_i(x_{I_i})} \subseteq S \right),$$

for some subset  $S \subseteq G$ , group elements  $g_i \in G$ , and functions  $f_i: \{-1, 1\}^{I_i} \rightarrow \{0, 1\}$ . Here  $x_{I_i}$  are the  $|I_i|$  bits of  $x$  indexed by  $I_i$ .

Note that generalized group products are unordered by definition. They are a generalization of several function classes that have received some attention in the past, including *modular sums* [41, 44, 30] (when  $G$  is the cyclic group and  $\ell = 1$ ), product tests with outputs  $\{-1, 1\}$  [33, 38, 39, 37] (when  $G = \{-1, 1\}$ ), and unordered *combinatorial shapes* [31, 30] (when  $G = \mathbb{Z}_{m+1}$ ).

An  $(m, \ell, G)$ -group product can be written as the disjoint composition of a width- $|G|$  permutation branching program and arbitrary Boolean functions on  $\ell$  bits. Since both of these classes admit  $L_{1,k}$  bounds with the acceptance probability factor, using the composition theorem of [8] we obtain Fourier growth bounds for generalized group products.

► **Corollary 16.** *Let  $f: \{-1, 1\}^n \rightarrow \{0, 1\}$  be an  $(m, \ell, G)$ -group product. Then  $L_{1,k}(f) \leq \Pr[f(U_n) = 1] \cdot O(\ell \cdot |G|)^k$ .*

Corollary 16 extends the Fourier growth bounds for product tests studied in [37] (where  $G = \{-1, 1\}$ ). Plugging our bounds into the polarizing random walk framework, we also obtain new pseudorandom generators for generalized group products.

► **Corollary 17.** *There is an explicit pseudorandom generator for  $(m, \ell, G)$ -group products with seed length  $O(\ell \cdot |G|)^2 \cdot \log(n/\epsilon) \cdot (\log(1/\epsilon) + \log \log n)$  and error  $\epsilon$ .*

Note that an  $(m, 1, G)$ -group product can be computed by a permutation branching program of width  $|G|$ , and a  $(m, \ell, G)$ -group product can be computed by a general branching program of width  $w = 2^\ell \cdot |G|$ . When  $\ell \geq 2$ , we are not aware of any PRG that fools  $(m, \ell, G)$ -group products better than unordered general branching programs. For the latter class, the current best PRGs are given by Forbes and Kelley [26] which, with the above choice of  $w$ , have seed lengths  $O(\ell + \log(|G|) + \log(n/\epsilon)) \log^2 n$  and  $\tilde{O}(2^\ell + |G|) \log(n/\epsilon) \log n$ . For comparison, for any error  $\epsilon = O(1)$ , our PRG for generalized group products has seed length  $(\ell \cdot |G|)^2 \cdot \tilde{O}(\log n)$ , which is nearly optimal when  $\ell \cdot |G| = O(1)$ , whereas the Forbes–Kelley PRGs have seed lengths  $\Omega(\log^2 n)$ .

Finally, we note that when  $G = \{-1, 1\}$ , there exists a PRG [37, 23] with seed length  $\tilde{O}(\ell + \log(m/\epsilon)) + \text{poly}(\log \log(n/\epsilon))$ , which is nearly optimal.

### 1.3 Techniques

Our main contribution is a simple inductive proof for bounding the first level  $L_{1,1}$  of a regular branching program in terms of the number of its accept states and *rejection* probability. Specifically, for a regular branching program  $B$  of  $s$  accept states, we prove that

$$L_{1,1}(B) \leq s \cdot \Pr[B(U_n) = 0]. \quad (1)$$

We prove Equation (1) by induction on  $n$ , the length of the program. We give some intuition for where Equation (1) came from. Let  $S$  be the set of accept states in the final layer. By regularity, the set of states  $T$  in the previous layer that lead to  $S$  must be at least the size of  $S$ . If they have the same size then the current layer is redundant. So we must have a nonempty set  $T_1$  of vertices that have only one outgoing edge leading to  $S$ . Since these vertices also have one edge leading to the complement of  $S$ , they all contribute to the probability that the program rejects. This suggests bounding  $L_{1,1}$  in terms of  $|S|$  and the rejection probability. In the proof we use regularity of the program to relate  $|S|$  to  $|T|$  and  $|T_1|$ .

Our first  $L_{1,k}$  bound  $L_{1,k}(B) \leq \Pr[B(U_n) = 1] \cdot (w-1)^k$  then follows from the same inductive argument in [19], where the authors proved  $L_{1,k}$  bounds for general constant-width branching programs. We note that this inductive argument relies on bounding the  $L_{1,1}$  of the *local monotone* of a branching program [14], which does not preserve the permutation property. Therefore, even for proving Fourier growth bounds of permutation programs, to apply this argument it is crucial to establish Equation (1) for the wider class of regular programs. Proving our second bound  $L_{1,k}(B) \leq s \cdot O((n \log n)/k)^{\frac{k-1}{2}}$  is slightly more involved. Our proof combines the inductive idea in [19] with the “level- $k$  inequalities” of Lee [37] (Lemma 22), which give  $L_{1,k}$  bounds for an arbitrary Boolean function in terms of its acceptance probability, and the approximator from Bogdanov, Hoza, Prakriya, and Pyne [9] (Lemma 23).

Given a regular branching program  $B$  of unbounded width, as in [9] we first construct a regular program  $B'$  that approximates  $B$  by rejecting all the states in  $B$  that can be reached with probability at most  $q := \tilde{O}(1/\sqrt{n})$ . In [9], they observed that the probability that the program  $B$  accepts via *any* of these “sudden reject” states is at most  $q$ . So the error function  $B - B'$  has small acceptance probability, and by the level- $k$  inequalities it has small  $L_{1,k}$ . So it suffices to bound the  $L_{1,k}$  of the approximator  $B'$ . We use the fact that  $B'$  has at most  $1/q$  non-sudden-reject states in each layer, and so the total number of non-reject states in  $B'$  is bounded by  $n/q = \text{poly}(n)$ . This allows us to apply an inductive argument to reduce bounding  $L_{1,k}(B')$  to bounding (roughly) the product of  $L_{1,k-1}(B')$  and  $L_{1,1}(B')$ . For  $L_{1,k-1}(B')$  we again use the level- $k$  inequalities, and for  $L_{1,1}(B)$  we use the bound in Equation (1). Note that while the states in  $B'$  all have reaching probability at least  $q$  in the original program  $B$ , some of them may have reaching probability much smaller than  $q$  in the approximator  $B'$ . To deal with this, we take a similar approach in [19] to handle states with small reaching probabilities separately.

### Organization

We begin by introducing some notation in the next paragraph. In Section 2, we prove our  $L_{1,k}$  bounds of regular branching programs (Theorem 6 and Proposition 7) and generalized group products (Corollary 16). In Section 3, we prove our coin problem bounds (Corollaries 8 and 9 and Claim 10), and construct our pseudorandom generators for permutation programs (Corollary 12 and Theorem 13) and generalized group products (Corollary 17).

## Notation

When we view a branching program as a graph, we will overload notation and consider the transition function as a map  $B_t: V_t \times \{-1, 1\} \rightarrow V_{t+1}$  in addition to thinking of it as a map  $B_t: [w] \times \{-1, 1\} \rightarrow [w]$ . Similarly, we will often think of the start state  $v_1$  as being an element of  $V_1$  instead of an element of  $[w]$ , and  $V_{\text{acc}} \subseteq V_{n+1}$  instead of  $V_{\text{acc}} \subseteq [w]$ , etc.

For a vertex  $v$  in some layer  $V_t$ , we use  $B_{\rightarrow v}$  to denote the sub-branching program of length  $t - 1$  but with  $v$  being the only accept vertex. We also use  $B_{v \rightarrow}$  to denote the sub-branching program of length  $n + 1 - t$  that starts at  $v$  and ends in  $V_n$  with accept vertices  $V_{\text{acc}}$ .

For ease of notation we use  $\mu(f)$  to denote the expectation of  $f$  under uniform inputs.

## 2 $L_{1,k}$ bounds of regular branching programs

In this section we prove our  $L_{1,k}$  bounds for regular branching programs (Theorem 6) and generalized group products (Corollary 16). We start with bounding the first level  $L_{1,1}$  of regular branching programs.

► **Lemma 18.** *Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a regular branching program of width  $w \in [1, \infty]$  with  $s$  accept states. Then*

$$L_{1,1}(B) \leq \min\{s \cdot \Pr[B(U_n) = 0], \Pr[B(U_n) = 1] \cdot (w - 1)\}.$$

**Proof.** We prove the first bound by induction on  $n$ . For  $n = 0$  the bound is vacuous. Now assume it holds for  $n - 1$  and consider a regular program  $B(x_1, \dots, x_n)$  with a set  $S$  of  $s$  accept states. Define the following 3 subsets  $T$ ,  $T_+$  and  $T_-$  of states in layer  $n - 1$ , where  $T$  is the set of states with both of its edges leading to  $S$ ,  $T_+$  is the set of states with only 1-edges leading to  $S$ , and likewise for  $T_-$  and  $(-1)$ -edges. Observe that we can write  $B$  as

$$B(x_1, \dots, x_n) = g(x_1, \dots, x_{n-1}) + \frac{1 + x_n}{2} g_+(x_1, \dots, x_{n-1}) + \frac{1 - x_n}{2} g_-(x_1, \dots, x_{n-1}),$$

where  $g, g_+, g_-$  are functions computable by regular branching programs of length  $n - 1$  with  $T, T_+$  and  $T_-$  as the sets of accept vertices, respectively. Note that  $s = |T| + \frac{|T_-| + |T_+|}{2}$ . Define  $g_1 := g_- + g_+$  and  $T_1 := T_+ \cup T_-$ . Now observe that for  $i \in [n - 1]$

$$|\widehat{B}(\{i\})| = \left| \widehat{g}(\{i\}) + \frac{1}{2} (\widehat{g_+}(\{i\}) + \widehat{g_-}(\{i\})) \right| \leq \frac{1}{2} |\widehat{g}(\{i\})| + \frac{1}{2} |\widehat{g}(\{i\}) + \widehat{g_1}(\{i\})|$$

and

$$|\widehat{B}(\{n\})| = \frac{1}{2} |\mu(g_+) - \mu(g_-)| \leq \frac{1}{2} (\mu(g_+) + \mu(g_-)) = \frac{1}{2} \mu(g_1)$$

$$\mu(B) = \mu(g) + \frac{\mu(g_1)}{2}.$$

Finally, as  $T$  and  $T_1$  are disjoint, the function  $g + g_1$  is Boolean and is computable by a regular program of length  $n - 1$  with  $|T| + |T_1|$  accept states, and  $\mu(g + g_1) = \mu(g) + \mu(g_1)$ . So applying our induction assumption on  $g$  and  $g + g_1$ , we have

$$\begin{aligned}
 2L_{1,1}(B) &= 2 \sum_{i=1}^n |\widehat{B}(\{i\})| \\
 &\leq \sum_{i=1}^{n-1} |\widehat{g}(\{i\})| + \sum_{i=1}^{n-1} |\widehat{(g+g_1)}(\{i\})| + \mu(g_1) \\
 &= L_{1,1}(g) + L_{1,1}(g+g_1) + \mu(g_1) \\
 &\leq |T| \cdot (1 - \mu(g)) + (|T| + |T_1|) \cdot (1 - \mu(g+g_1)) + \mu(g_1) \\
 &= (2|T| + |T_1|) - (|T| \cdot \mu(g) + (|T| + |T_1|) \cdot \mu(g+g_1) - \mu(g_1)) \\
 &= 2s - ((2|T| + |T_1|) \cdot \mu(g) + (|T| + |T_1| - 1) \cdot \mu(g_1)) \\
 &\leq 2s - ((2|T| + |T_1|) \cdot \mu(g) + (|T| + \frac{|T_1|}{2}) \cdot \mu(g_1)) \\
 &= 2s - 2s \left( \mu(g) + \frac{\mu(g_1)}{2} \right) \\
 &= 2s \cdot (1 - \mu(B)),
 \end{aligned}$$

where the last inequality uses that  $\frac{|T_1|}{2} \cdot \mu(g_1) \geq \mu(g_1)$ , since  $T_1$  is either empty or has size at least 2.

To prove the second bound, suppose  $B$  is a regular program of width  $w < \infty$  with a set  $S$  of accept states. For every state  $v \in S$ , the function  $1 - B_{\rightarrow v}$  is computable by a regular branching program with  $w - 1$  accept states. Since  $L_{1,1}(B_{\rightarrow v}) = L_{1,1}(1 - B_{\rightarrow v})$ , it follows from the first bound we just proved that  $L_{1,1}(B_{\rightarrow v}) \leq \mathbf{Pr}[B_{\rightarrow v}(U_n) = 1] \cdot (w - 1)$ . Summing over all the accept states  $v \in S$  gives the second bound.  $\blacktriangleleft$

To obtain  $L_{1,k}$  bounds at higher levels, we will apply the inductive argument in [51, 19]. We first recall the local monotoneization of a branching program introduced in [14, 19]. For a branching program  $B$ , we define the *local monotoneization*  $B'$  of  $B$  by the following process. For every layer  $t$ , state  $u \in V_t$ , and input  $b \in \{-1, 1\}$ , let  $v_b := B_t(u, b)$  and define

$$B'_t(u, b) = \begin{cases} B_t(u, -b) & \text{if } \mu(B_{v_1 \rightarrow}) < \mu(B_{v_{-1} \rightarrow}) \\ B_t(u, b) & \text{otherwise.} \end{cases}$$

In words, we swap the two outgoing edge-labels of  $u$  whenever  $\mu(B_{v_1 \rightarrow}) < \mu(B_{v_{-1} \rightarrow})$ . As the underlying graph of  $B'$  remains the same as  $B$ , if  $B$  is regular then  $B'$  is also regular (with the same set of accept states). Also  $\mu(B_{v \rightarrow}) = \mu(B'_{v \rightarrow})$  for every state  $v$ . By construction we have  $|\widehat{B}(\{i\})| = |\widehat{B'}(\{i\})|$  for every  $i \in [n]$ .

The following claim reduces bounding  $L_{1,k}$  of a branching program to bounding its  $L_{1,k-1}$ .

$\triangleright$  **Claim 19 ([19]).** Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a branching program, and  $B'$  be its local monotoneization. Then  $L_{1,k+1}(B) \leq \sum_{i=1}^n \sum_{v \in V_i} (L_{1,k}(B_{\rightarrow v}) \cdot \widehat{B'_{v \rightarrow}}(\{i\}))$ .



Proof. We have

$$\begin{aligned}
L_{1,k+1}(B) &= \sum_{i=1}^n \sum_{\substack{S \subseteq \{1, \dots, i-1\}: \\ |S|=k}} \left| \widehat{B}(S \cup \{i\}) \right| \\
&= \sum_{i=1}^n \sum_{\substack{S \subseteq \{1, \dots, i-1\}: \\ |S|=k}} \left| \sum_{v \in V_i} \widehat{B_{\rightarrow v}}(S) \widehat{B_{v \rightarrow}}(\{i\}) \right| \\
&\leq \sum_{i=1}^n \sum_{\substack{S \subseteq \{1, \dots, i-1\}: \\ |S|=k}} \sum_{v \in V_i} \left( \left| \widehat{B_{\rightarrow v}}(S) \right| \cdot \left| \widehat{B_{v \rightarrow}}(\{i\}) \right| \right) \\
&= \sum_{i=1}^n \sum_{v \in V_i} \left( \sum_{\substack{S \subseteq \{1, \dots, i-1\}: \\ |S|=k}} \left| \widehat{B_{\rightarrow v}}(S) \right| \right) \left| \widehat{B_{v \rightarrow}}(\{i\}) \right| \\
&= \sum_{i=1}^n \sum_{v \in V_i} \left( L_{1,k}(B_{\rightarrow v}) \cdot \widehat{B'_{v \rightarrow}}(\{i\}) \right). \quad \triangleleft
\end{aligned}$$

► **Theorem 20.** Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a regular branching program of width  $w$ . Then

$$L_{1,k}(B) \leq \Pr[B(U_n) = 1] \cdot (w-1)^k.$$

**Proof.** Let  $B'$  be the local monotonization of  $B$ . By Claim 19,

$$\begin{aligned}
L_{1,k+1}(B) &\leq \sum_{i=1}^n \sum_{v \in V_i} \left( L_{1,k}(B_{\rightarrow v}) \cdot \widehat{B'_{v \rightarrow}}(\{i\}) \right) \\
&\leq (w-1)^k \sum_{i=1}^n \sum_{v \in V_i} \Pr[B_{\rightarrow v}(U_i) = 1] \cdot \widehat{B'_{v \rightarrow}}(\{i\}) \\
&= (w-1)^k \sum_{i=1}^n \sum_{v \in V_i} \Pr[B'_{\rightarrow v}(U_i) = 1] \cdot \widehat{B'_{v \rightarrow}}(\{i\}) \\
&= (w-1)^k \sum_{i=1}^n \widehat{B'}(\{i\}) \\
&\leq \Pr[B(U_n) = 1] \cdot (w-1)^{k+1}. \quad \blacktriangleleft
\end{aligned}$$

► **Theorem 21.** Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be any regular branching program with  $s$  accepting states. Then

$$L_{1,k}(B) \leq s \Pr[B(U_n) = 0] \cdot O\left(\frac{n}{k} \left(1 + \frac{1}{k} \log\left(\frac{n}{\Pr[B(U_n) = 0]}\right)\right)\right)^{\frac{k-1}{2}}.$$

We will use the following “ $L_1$  level- $k$  inequalities,” which follows from applying Cauchy–Schwarz to Lemma 10 in [37], and the observation that every non-constant Boolean function  $f$  has  $\mu(f) \geq 2^{-n}$ .

► **Lemma 22.** For every Boolean function  $f: \{0, 1\}^n \rightarrow \{0, 1\}$ , we have

$$L_{1,k}(f) \leq \sqrt{\binom{n}{k}} \cdot \mu(f) \cdot O\left(\log\left(\frac{2}{\mu(f)^{1/k}}\right)\right)^{k/2} \leq \Pr[f(U_n) = 1] \cdot O(n)^k.$$

We also need the following lemma in [9], which follows from applying a union bound over all the  $s$  accept vertices to Claim 3.1 in [9].

► **Lemma 23** (Claim 3.1 in [9]). *Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a regular branching program with  $s$  accept vertices. Let  $V^\epsilon := \{v : \mu(B_{\rightarrow v}) \leq \epsilon\}$  be the set of states in  $B$  that have at most  $\epsilon$  probability of being reached over uniform inputs. Then for every state  $v$ ,*

$$\Pr_{x \sim U_n} [B_{\rightarrow v}(x) = 1 \wedge B_{\rightarrow u}(x_1, \dots, x_t) = 1 \text{ for some } t \in [n] \text{ and } u \in V^\epsilon] \leq s \cdot \epsilon.$$

**Proof of Theorem 21.** Let  $\bar{\mu} := 1 - \mu(B)$ , and define

$$q := \frac{\bar{\mu}}{s} \left( \frac{k}{n \log(ns/\bar{\mu})} \right)^{1/2}.$$

Let  $V^q$  be the set of states  $v$  in  $B$  with  $\mu(B_{\rightarrow v}) \leq q$ . As in [9], we construct another regular program  $B'$  that approximates  $B$  as follows. For each state  $u \in V^q$ , we “sudden reject”  $u$  by rewiring its outgoing edges to an “unused” state. Specifically, we construct  $B'$  by modifying  $B$  as follows. We iterate each  $u \in V^q$  and do the following: Suppose  $u \in V_t \cap V^q$  for some layer  $t$ . Let  $u' \in V_t$  be a new dummy state that is initially always wired to itself in other layers and such has  $\mu(B_{\rightarrow u'}) = \mu(B_{u' \rightarrow}) = 0$ . We swap the outgoing  $b$ -edges of  $u$  and  $u'$  for both  $b \in \{-1, 1\}$ . Observe that for every state  $u$  in  $B'$  that is reached with positive probability we have  $\mu(B_{\rightarrow u}) \geq q$  and so in each layer of  $B'$  there are at most  $1/q$  many non-sudden-reject states with  $\mu(B'_{\rightarrow u}) > 0$ .

We now bound above  $L_{1,k+1}(B)$  by  $L_{1,k+1}(B - B') + L_{1,k+1}(B')$ . By Lemma 23,

$$\begin{aligned} \Pr[(B - B')(U_n) = 1] \\ = \Pr_{x \sim U_n} [B(x) = 1 \wedge B_{\rightarrow u}(x_1, \dots, x_t) = 1 \text{ for some } t \text{ and } u \in V^q] \leq s \cdot q. \end{aligned}$$

As  $B - B'$  has small acceptance probability, it follows from Lemma 22 that

$$\begin{aligned} L_{1,k+1}(B - B') \\ \leq O(1)^k \sqrt{\binom{n}{k+1}} \cdot s \cdot q \cdot \left( \log \frac{2}{q^{1/(k+1)}} \right)^{\frac{k+1}{2}} \\ \leq O(1)^k \cdot \left( \frac{n}{k} \right)^{\frac{k+1}{2}} \cdot \bar{\mu} \cdot \left( \frac{k}{n \log(ns/\bar{\mu})} \right)^{1/2} \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right)^{\frac{k+1}{2}} \\ \leq O(1)^k \cdot \bar{\mu} \cdot \left( \frac{n}{k} \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right) \right)^{k/2} \left( \frac{n}{k} \cdot \frac{k}{n \log(ns/\bar{\mu})} \cdot \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right) \right)^{1/2} \\ \leq O(1)^k \cdot \bar{\mu} \cdot \left( \frac{n}{k} \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right) \right)^{k/2}, \end{aligned}$$

It remains to bound  $L_{1,k+1}(B')$ . Let  $B''$  be the local monotonicization of  $B'$ . By Claim 19 and Lemma 22,

$$\begin{aligned} L_{1,k+1}(B') &\leq \sum_{i=1}^n \sum_{v \in V'_i} \left( L_{1,k}(B'_{\rightarrow v}) \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \\ &\leq O(1)^k \cdot \left( \frac{n}{k} \right)^{k/2} \cdot \sum_{i=1}^n \sum_{v \in V'_i} \left( \mu(B'_{\rightarrow v}) \cdot \log \left( \frac{2}{\mu(B'_{\rightarrow v})^{1/k}} \right)^{k/2} \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right). \end{aligned}$$

We separate the double sum above into two parts, depending on whether the states  $v$  can be reached with probability at least  $q\bar{\mu}/n$ .

We first consider those with reaching probability less than  $q\bar{\mu}/n$ . As the function  $x \mapsto x \log(2/x^{1/k})^{k/2}$  is increasing for  $x \in [0, 1]$ , we have

$$\begin{aligned} & \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ \mu(B_{\rightarrow v}) < \frac{q\bar{\mu}}{n}}} \left( \mu(B_{\rightarrow v}) \cdot \log \left( \frac{2}{\mu(B_{\rightarrow v})^{1/k}} \right)^{k/2} \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \\ & \leq O(1)^k \cdot \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right)^{k/2} \cdot \frac{q\bar{\mu}}{n} \cdot \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ 0 < \mu(B'_{\rightarrow v}) < \frac{q\bar{\mu}}{n}}} \widehat{B''_{v \rightarrow}}(\{i\}) \\ & \leq O(1)^k \cdot \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right)^{k/2} \bar{\mu}, \end{aligned}$$

where the last inequality is because  $|\widehat{B''_{v \rightarrow}}(\{i\})| \leq 1$  and we are summing over at most  $n \cdot 1/q$  many vertices.

For those states that are reached with probability at least  $q\bar{\mu}/n$ , we apply Lemma 22 and our  $L_{1,1}$  bound in Lemma 18. We have

$$\begin{aligned} & \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ \mu(B'_{\rightarrow v}) \geq \frac{q\bar{\mu}}{n}}} \left( \mu(B'_{\rightarrow v}) \cdot \log \left( \frac{2}{\mu(B'_{\rightarrow v})^{1/k}} \right)^{k/2} \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \\ & \leq O \left( 1 + \frac{\log(n/\bar{\mu})}{k} \right)^{k/2} \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ \mu(B'_{\rightarrow v}) \geq \frac{q\bar{\mu}}{n}}} \left( \mu(B'_{\rightarrow v}) \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right), \end{aligned}$$

where by Lemma 18 we get

$$\begin{aligned} & \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ \mu(B'_{\rightarrow v}) \geq \frac{q\bar{\mu}}{n}}} \left( \mu(B'_{\rightarrow v}) \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \\ & = \sum_{i=1}^n \sum_{\substack{v \in V'_i: \\ \mu(B''_{\rightarrow v}) \geq \frac{q\bar{\mu}}{n}}} \left( \mu(B''_{\rightarrow v}) \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \quad (\mu(B''_{\rightarrow v}) = \mu(B'_{\rightarrow v})) \\ & \leq \sum_{i=1}^n \sum_{v \in V'_i} \left( \mu(B''_{\rightarrow v}) \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \quad (\widehat{B''_{v \rightarrow}}(\{i\}) \geq 0) \\ & \leq \sum_{i=1}^n \widehat{B''}(\{i\}) \leq s \cdot \mathbf{Pr}[B''(U_n) = 0] \leq 2s\bar{\mu}, \end{aligned}$$

where we use  $\mathbf{Pr}[B''(U_n) = 0] = \mathbf{Pr}[B'(U_n) = 0] \leq \mathbf{Pr}[B(U_n) = 0] + sq \leq 2\bar{\mu}$  in the last inequality. Hence,

$$\begin{aligned} L_{1,k+1}(B') & \leq O(1)^k \cdot \left( \frac{n}{k} \right)^{k/2} \cdot \sum_{i=1}^n \sum_{v \in V'_i} \left( \mu(B'_{\rightarrow v}) \cdot \log \left( \frac{2}{\mu(B'_{\rightarrow v})^{1/k}} \right)^{k/2} \cdot \widehat{B''_{v \rightarrow}}(\{i\}) \right) \\ & \leq O \left( \frac{n}{k} \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right) \right)^{k/2} s\bar{\mu}. \end{aligned}$$

## 2:14 Fourier Growth of Regular Branching Programs

Therefore, we have

$$\begin{aligned} L_{1,k+1}(B) &\leq L_{1,k+1}(B - B') + L_{1,k+1}(B') \\ &\leq O(1)^k \cdot s\bar{\mu} \cdot \left( \frac{n}{k} \left( 1 + \frac{\log(ns/\bar{\mu})}{k} \right) \right)^{k/2} \\ &\leq O(1)^k \cdot s\bar{\mu} \cdot \left( \frac{n}{k} \left( 1 + \frac{\log(n/\bar{\mu})}{k} \right) \right)^{k/2}, \end{aligned}$$

where the last inequality is because if  $s \geq \sqrt{n}$ , then the conclusion directly follows from Lemma 22; so we can assume  $s \leq \sqrt{n}$ . ◀

Theorem 6 now follows from Theorems 20 and 21.

**Proof of Theorem 6.** The first bound  $L_{1,k}(B) \leq \Pr[B(U_n) = 1] \cdot (w - 1)^k$  directly follows from Theorem 20. We now show that Theorem 21 implies the second bound  $L_{1,k}(B) \leq s \cdot O((n \log n)/k)^{\frac{k-1}{2}}$ . Let  $\bar{\mu} := \Pr[B(U_n) = 0]$ . As the function  $x \mapsto x \log(2/x^{1/k})^{\frac{k-1}{2}}$  is increasing for  $x \in [0, 1]$ , we have

$$\begin{aligned} \bar{\mu} \left( 1 + \frac{\log(n/\bar{\mu})}{k} \right)^{\frac{k-1}{2}} &= \bar{\mu} \left( \frac{\log n}{k} + \log \left( \frac{2}{\bar{\mu}^{1/k}} \right) \right)^{\frac{k-1}{2}} \\ &\leq 2 \max \left\{ \bar{\mu} \left( \frac{\log n}{k} \right)^{\frac{k-1}{2}}, \bar{\mu} \log \left( \frac{2}{\bar{\mu}^{1/k}} \right)^{\frac{k-1}{2}} \right\} \leq 2(\log n)^{\frac{k-1}{2}}. \end{aligned}$$

Hence, by Theorem 20,

$$L_{1,k}(B) \leq s \cdot \bar{\mu} \cdot O \left( \frac{n}{k} \left( 1 + \frac{\log(n/\bar{\mu})}{k} \right) \right)^{\frac{k-1}{2}} \leq s \cdot O \left( \frac{n \log n}{k} \right)^{\frac{k-1}{2}}. \quad \blacktriangleleft$$

We now prove Proposition 7. This is a direct consequence of a result of Błasiok, Ivanov, Jin, Lee, Servedio and Viola:

► **Theorem 24** (Theorem 24 of [8]). *For all positive integers  $n$  and  $k$  where  $k \leq n$ , there is a symmetric  $\mathbb{F}_2$ -polynomial  $p(x_1, \dots, x_n)$  of degree a power of two in  $[\sqrt{kn}, 8\sqrt{kn}]$  such that*

$$M_k(p) := \left| \sum_{|S|=k} \widehat{p}(S) \right| \geq (e^{-k}/2) \binom{n}{k}^{1/2}.$$

Their result is stated for  $L_{1,k}(p)$ , but the proof holds without modification for  $M_k(p)$ .

**Proof of Proposition 7.** Given  $n$  and  $k$ , let  $p(x_1, \dots, x_n)$  be the  $\mathbb{F}_2$ -symmetric polynomial in Theorem 24. As observed in [8], as a consequence of a result of Bhatnagar, Gopalan, and Lipton [6],  $p$  can be computed by a permutation branching program  $B$  of width  $16\sqrt{kn}$ . As  $\sum_{|S|=k} \widehat{B}(S) = \sum_{v \in V_{\text{acc}}} \sum_{|S|=k} \widehat{B \rightarrow v}(S)$ , the conclusion follows by an averaging argument. ◀

We end this section by proving the  $L_{1,k}$  bounds for generalized group products. To do so, we recall the formal statement of the composition theorem of [8].

► **Theorem 25** (Theorem 31 in [8]). *Suppose  $\mathcal{F}$  and  $\mathcal{G}$  are closed under negation of their outputs. Let  $g_1, \dots, g_m \in \mathcal{G}$  and let  $f \in \mathcal{F}$ , where  $\mathcal{F}$  is closed under restrictions. Suppose that for every  $1 \leq k \leq K$ , we have*

1.  $L_{1,k}(f) \leq \Pr[f(U_m) = 1] \cdot a_{\text{outer}} \cdot b_{\text{outer}}^k$  for every  $f \in \mathcal{F}$ , and
  2.  $L_{1,k}(g) \leq \Pr[g(U_\ell) = 1] \cdot a_{\text{inner}} \cdot b_{\text{inner}}^k$  for every  $g \in \mathcal{G}$ .
- Then for every function  $h \in \mathcal{F} \circ \mathcal{G}$ , we have that

$$L_{1,K}(h) \leq \Pr[h(U_{m\ell}) = 1] \cdot a_{\text{outer}} \cdot (a_{\text{inner}} b_{\text{inner}} b_{\text{outer}})^K.$$

**Proof of Corollary 16.** An  $(m, \ell, G)$ -product can be computed by the disjoint composition of a width- $|G|$  permutation branching program and arbitrary Boolean functions on  $\ell$  bits, where both classes are closed under negation of their outputs and restrictions. Note that applying the map  $f \mapsto 2f - 1$  to a  $\{0, 1\}$ -valued function  $f$  only affects its  $L_{1,k}$  by at most a factor of 2. So we can apply Theorem 25 to Theorem 6 and Lemma 22.  $\blacktriangleleft$

### 3 Coin theorems and pseudorandom generators

In this section, we prove our coin problem bounds for regular and permutation branching programs (Corollaries 8 and 9 and Claim 10), and construct PRGs for permutation branching programs (Corollary 9 and Theorem 13) and generalized group products (Corollary 17).

We start with Corollary 8.

**Proof of Corollary 8.** Let  $B$  be a regular branching program. We identify  $B$  with its multilinear extension. By linearity of expectation and Theorem 6, we have

$$\begin{aligned} |\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(X_0)]| &= |B(\vec{\delta}) - B(\vec{0})| \\ &\leq \sum_{k=1}^n \delta^k \sum_{|S|=k} |\hat{B}(S)| \\ &\leq \delta L_{1,1}(B) + \sum_{k=2}^n \delta^k L_{1,k}(B) \\ &\leq \delta s + \sum_{k=2}^n \delta^k \min\{w^k, s \cdot O(\sqrt{n \log n})^{k-1}\} \\ &\leq \delta s + \delta^2 \cdot O\left(\min\{w^2, s\sqrt{n \log n}\}\right), \end{aligned}$$

where the last inequality is because when  $\delta \leq \alpha \max\{1/w, 1/\sqrt{n \log n}\}$ , then at least one of the summations  $\sum_k (\delta w)^k$  and  $\sum_k O(\delta \sqrt{n \log n})^k$  is a geometric sum with ratio at most  $1/2$ , and thus is bounded by twice of its first term.  $\blacktriangleleft$

Corollary 9 follows from applying a result of Avishay Tal establishing that  $L_{1,1}$  bounds imply coin problem bounds for classes that are closed under restrictions to Theorem 6.

► **Lemma 26** (Lemma 3.2 in [3]). *Let  $\mathcal{F}$  be a function class that is closed under restrictions. Then for every  $f \in \mathcal{F}$ ,*

$$|\mathbf{E}[f(X_\delta)] - \mathbf{E}[f(X_0)]| \leq \ln\left(\frac{1}{1-\delta}\right) L_{1,1}(\mathcal{F}) \leq \frac{\delta}{1-\delta} L_{1,1}(\mathcal{F}).$$

We now prove Claim 10. The idea is similar to proof idea behind Proposition 7. Here we give a self-contained argument. We approximate the Majority function on some  $n = \Theta(1/\delta^2)$  bits by computing it correctly on inputs of Hamming weights between  $n/2 + \Theta(\sqrt{n})$  and  $n/2 - \Theta(\sqrt{n})$ . This can be implemented by counting their Hamming weights modulo  $\Theta(\sqrt{n})$  and hence can be done using a permutation program of width  $\Theta(\sqrt{n})$ .

**Proof of Claim 10.** Let  $n := 32/\delta^2$  and  $m := 64/\delta$ . Consider the function  $f: \{-(m-1), \dots, m\} \rightarrow \{0, 1\}$  defined by  $f(\ell) := 1$  if and only if  $\ell \geq m/4$ . We first construct the permutation program  $B$ , which on inputs  $x$  where  $\sum_i x_i \in \ell + 2m\mathbb{Z}$  for some  $\ell \in \{-(m-1), \dots, m\}$ , outputs  $B(x) := f(\ell)$ . By counting modulo  $2m$ , this can be computed with width  $2m$  and at least  $m/2$  accept states. By the Chernoff bound,

$$\Pr[B(X_\delta) = 0] \leq \Pr\left[\left|\sum_{i=1}^n (X_\delta)_i\right| \geq m\right] + \Pr\left[\sum_{i=1}^n (X_\delta)_i < m/4\right] \leq 1/20.$$

Similarly,  $\Pr[B(X_0) = 1] \leq 1/20$ . Therefore,  $\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(X_0)] \geq 9/10$ .

We now modify  $B$  by choosing  $s$  of its at most  $m$  many accept states uniformly at random, then letting  $B$  accept only at these  $s$  states and reject the rest of them. It follows by an averaging argument that there exists a choice of  $s$  accepting states such that the modified program  $B'$  satisfies

$$\mathbf{E}[B'(X_\delta)] - \mathbf{E}[B'(X_0)] \geq (s/m) \cdot (9/10) \geq s\delta/1000. \quad \blacktriangleleft$$

We now construct PRGs for bounded width permutation branching programs and generalized group products. We will use the following result that constructs PRGs from Fourier growth bounds using the “polarizing random walk framework.”

► **Theorem 27** (Theorem 1.3 in [16]). *Let  $\mathcal{F}$  be a function class on  $n$  bits that is closed under restrictions. Suppose  $L_{1,k}(\mathcal{F}) \leq b^k$  for some  $b \geq 1$ . Then there exists an explicit pseudorandom generator for  $\mathcal{F}$  with seed length  $b^2 \cdot O(\log(n/\epsilon))(\log(1/\epsilon) + \log \log n)$  and error  $\epsilon$ .*

Corollaries 12 and 17 then follow from applying Theorem 27 to Theorem 6 and Corollary 16, respectively.

We prove Theorem 13 by approximately sampling  $\delta$ -biased coins. To do this efficiently, we follow the approach in [33], and defer the proof to Appendix A.

---

## References

- 1 Scott Aaronson. BQP and the polynomial hierarchy. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 141–150. ACM, New York, 2010.
- 2 Scott Aaronson and Andrew Drucker. Advice coins for classical and quantum computation. In *Automata, languages and programming. Part I*, volume 6755 of *Lecture Notes in Comput. Sci.*, pages 61–72. Springer, Heidelberg, 2011. doi:10.1007/978-3-642-22006-7\_6.
- 3 Rohit Agrawal. Coin theorems and the Fourier expansion. *Chic. J. Theoret. Comput. Sci.*, pages Art. 4, 15, 2020. doi:10.4086/cjtcs.
- 4 M. Ajtai.  $\Sigma_1^1$ -formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.
- 5 Nikhil Bansal and Makrand Sinha.  $k$ -forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, New York, NY, USA, 2021. doi:10.1145/3406325.3451040.
- 6 Nayantara Bhatnagar, Parikshit Gopalan, and Richard J. Lipton. Symmetric polynomials over  $\mathbb{Z}_m$  and simultaneous communication protocols. *J. Comput. Syst. Sci.*, 72(2):252–285, 2006. doi:10.1016/j.jcss.2005.06.007.
- 7 Eric Blais, Li-Yang Tan, and Andrew Wan. An inequality for the fourier spectrum of parity decision trees, 2015. arXiv:1506.01055.

- 8 Jaroslaw Blasiok, Peter Ivanov, Yaonan Jin, Chin Ho Lee, Rocco A. Servedio, and Emanuele Viola. Fourier growth of structured  $F_2$ -polynomials and applications. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 53:1–53:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.53.
- 9 Andrej Bogdanov, William Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. *Electron. Colloquium Comput. Complex.*, page 143, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/143>.
- 10 Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011*, pages 240–246. IEEE Computer Soc., Los Alamitos, CA, 2011. doi:10.1109/FOCS.2011.57.
- 11 Mark Braverman, Sumegha Garg, and David P. Woodruff. The coin problem with applications to data streams. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 318–329. IEEE Computer Soc., Los Alamitos, CA, [2020] ©2020.
- 12 Mark Braverman, Sumegha Garg, and Or Zamir. Tight space complexity of the coin problem. *Electron. Colloquium Comput. Complex.*, page 83, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/083>.
- 13 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. doi:10.1137/120875673.
- 14 Joshua Brody and Elad Verbin. The coin problem, and pseudorandomness for branching programs. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010*, pages 30–39. IEEE Computer Soc., Los Alamitos, CA, 2010.
- 15 Eshan Chattopadhyay, Jason Gaitonde, Chin Ho Lee, Shachar Lovett, and Abhishek Shetty. Fractional Pseudorandom Generators from Any Fourier Level. In Valentine Kabanets, editor, *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:24, 2021. doi:10.4230/LIPIcs.CCC.2021.10.
- 16 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. *Theory Comput.*, 15:Paper No. 10, 26, 2019. doi:10.4086/toc.2019.v015a010.
- 17 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, Shachar Lovett, and David Zuckerman. XOR Lemmas for Resilient Functions against Polynomials. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 234–246, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384242.
- 18 Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom Generators from the Second Fourier Level and Applications to AC0 with Parity Gates. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2019.22.
- 19 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *STOC’18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 363–375. ACM, New York, 2018. doi:10.1145/3188745.3188800.
- 20 Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae (extended abstract). In *Advances in cryptology—CRYPTO 2013. Part II*, volume 8043 of *Lecture Notes in Comput. Sci.*, pages 185–202. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40084-1\_11.



- 21 Anindya De. Pseudorandomness for permutation and regular branching programs. In *26th Annual IEEE Conference on Computational Complexity*, pages 221–231. IEEE Computer Soc., Los Alamitos, CA, 2011.
- 22 Dean Doron, Pooya Hatami, and William M. Hoza. Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas. In *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:34. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.16.
- 23 Dean Doron, Pooya Hatami, and William M. Hoza. Log-Seed Pseudorandom Generators via Iterated Restrictions. In *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:36. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.6.
- 24 Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan. Pseudorandom Generators for Read-Once Monotone Branching Programs. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:21, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.58.
- 25 Alexandros Eskenazis and Paata Ivanisvili. Learning low-degree functions from a logarithmic number of random queries. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 203–207, 2022.
- 26 Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *59th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2018*, pages 946–955. IEEE Computer Soc., Los Alamitos, CA, 2018. doi:10.1109/FOCS.2018.00093.
- 27 Uma Girish, Ran Raz, and Wei Zhan. Lower Bounds for XOR of Forrelations. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:14, 2021. URL: <https://drops.dagstuhl.de/opus/volltexte/2021/14745>.
- 28 Uma Girish, Avishay Tal, and Kewen Wu. Fourier Growth of Parity Decision Trees. In *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:36. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.39.
- 29 Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal.  $AC^0[p]$  Lower Bounds Against MCSP via the Coin Problem. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.66.
- 30 Parikshit Gopalan, Daniel M. Kane, and Raghu Meka. Pseudorandomness via the discrete Fourier transform. *SIAM J. Comput.*, 47(6):2451–2487, 2018. doi:10.1137/16M1062132.
- 31 Parikshit Gopalan, Raghu Meka, Omer Reingold, and David Zuckerman. Pseudorandom generators for combinatorial shapes. *SIAM J. Comput.*, 42(3):1051–1076, 2013. doi:10.1137/110854990.
- 32 Parikshit Gopalan, Rocco A. Servedio, Avishay Tal, and Avi Wigderson. Degree and sensitivity: tails of two distributions. *Electron. Colloquium Comput. Complex.*, 23:69, 2016. URL: <http://eccc.hpi-web.de/report/2016/069>.
- 33 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018. doi:10.1137/17M1129088.
- 34 William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom Generators for Unbounded-Width Permutation Branching Programs. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.7.



- 35 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):Art. 11, 16, 2019. doi:10.1145/3230630.
- 36 Siddharth Iyer, Anup Rao, Victor Reis, Thomas Rothvoss, and Amir Yehudayoff. Tight bounds on the fourier growth of bounded functions on the hypercube. *CoRR*, abs/2107.06309, 2021. arXiv:2107.06309.
- 37 Chin Ho Lee. Fourier Bounds and Pseudorandom Generators for Product Tests. In *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.7.
- 38 Chin Ho Lee and Emanuele Viola. The coin problem for product tests. *ACM Trans. Comput. Theory*, 10(3):Art. 14, 10, 2018. doi:10.1145/3201787.
- 39 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: pseudorandom generators for read-once polynomials. *Theory Comput.*, 16:Paper No. 7, 50, 2020. doi:10.4086/toc.2020.v016a007.
- 40 Nutan Limaye, Karteeek Sreenivasaiiah, Srikanth Srinivasan, Utkarsh Tripathi, and S. Venkitesh. A fixed-depth size-hierarchy theorem for  $AC^0[\oplus]$  via the coin problem. *SIAM J. Comput.*, 50(4):1461–1499, 2021. doi:10.1137/19M1276467.
- 41 Shachar Lovett, Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom bit generators that fool modular sums. In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 615–630. Springer, Berlin, 2009. doi:10.1007/978-3-642-03685-9\_46.
- 42 Yishay Mansour. An  $O(n^{\log \log n})$  learning algorithm for DNF under the uniform distribution. *J. Comput. System Sci.*, 50(3, part 3):543–550, 1995. Fifth Annual Workshop on Computational Learning Theory (COLT) (Pittsburgh, PA, 1992). doi:10.1006/jcss.1995.1043.
- 43 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 626–637. ACM, New York, 2019. doi:10.1145/3313276.3316319.
- 44 Raghu Meka and David Zuckerman. Small-bias spaces for group products. In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 658–672. Springer, Berlin, 2009. doi:10.1007/978-3-642-03685-9\_49.
- 45 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.
- 46 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. doi:10.1017/CB09781139814782.
- 47 Ryan O'Donnell and Rocco A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007. doi:10.1137/060669309.
- 48 Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract). In *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.33.
- 49 Edward Pyne and Salil Vadhan. Deterministic approximation of random walks via queries in graphs of unbounded size. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 57–67. SIAM, 2022.
- 50 Ran Raz and Avishay Tal. Oracle separation of BQP and PH. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 13–23. ACM, New York, 2019. doi:10.1145/3313276.3316315.
- 51 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, randomization, and combinatorial optimization*, volume 8096 of *Lecture Notes in Comput. Sci.*, pages 655–670. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40328-6\_45.

- 52 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 457–466. ACM, New York, 2006. doi:10.1145/1132516.1132583.
- 53 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.
- 54 Alexander A. Sherstov, Andrey A. Storozhenko, and Pei Wu. *An Optimal Separation of Randomized and Quantum Query Complexity*, pages 1289–1302. Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3406325.3451019.
- 55 John Steinberger. The distinguishability of product distributions by read-once branching programs. In *2013 IEEE Conference on Computational Complexity—CCC 2013*, pages 248–254. IEEE Computer Soc., Los Alamitos, CA, 2013. doi:10.1109/CCC.2013.33.
- 56 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. *Electron. Colloquium Comput. Complex.*, 2012. URL: <http://eccc.hpi-web.de/report/2012/083>.
- 57 Thomas Steinke, Salil Vadhan, and Andrew Wan. Pseudorandomness and Fourier-growth bounds for width-3 branching programs. *Theory Comput.*, 13:Paper No. 12, 50, 2017. doi:10.4086/toc.2017.v013a012.
- 58 Avishay Tal. Tight Bounds on the Fourier Spectrum of AC0. In *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.15.
- 59 Avishay Tal. Towards optimal separations between quantum and randomized query complexities. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 228–239, 2020. doi:10.1109/FOCS46700.2020.00030.
- 60 L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.
- 61 Emanuele Viola. Fourier Conjectures, Correlation Bounds, and Majority. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111:1–111:15, 2021. doi:10.4230/LIPIcs.ICALP.2021.111.

## A Proof of Theorem 13

Recall that  $H(x) = x \log(\frac{1}{x}) + (1-x) \log(\frac{1}{1-x})$  denotes the binary entropy function. For two distributions  $X$  and  $Y$ , we use  $\|X - Y\|_1$  to denote their total variation distance.

► **Lemma 28.** *Given  $\delta > 0$ , there is some  $s = H(1/2 + 0.499\delta)n + o(n)$  and a polynomial-time computable function  $f: \{0, 1\}^s \rightarrow \{-1, 1\}^n$  such that  $\|X_\delta - f(U_s)\|_1 \leq \delta/100$ .*

As its proof is only a slight modification of the one in [33], we defer it to the end of this section. To construct our PRG, it suffices to sample a distribution close to  $X_\delta$  using Lemma 28.

**Proof of Theorem 13.** Let  $f: \{0, 1\}^s \rightarrow \{-1, 1\}^n$  be the function obtained from Lemma 28 with the given  $\delta$ , where

$$s \leq H(1/2 + 0.499\delta)n + o(n) + O(\log(1/\delta)) = (H(1/2 + 0.499\delta) + o(1))n.$$

Let  $B: \{-1, 1\}^n \rightarrow \{0, 1\}$  be a permutation branching program with a single accept state. Then

$$\begin{aligned}
& |\mathbf{E}[B(U_n)] - \mathbf{E}[B(f(U_s))]| \\
& \leq |\mathbf{E}[B(U_n)] - \mathbf{E}[B(X_\delta)]| + |\mathbf{E}[B(X_\delta)] - \mathbf{E}[B(f(U_s))]| \\
& \leq |\mathbf{E}[B(U_n)] - \mathbf{E}[B(X_\delta)]| + \delta/100 \quad (\text{Lemma 28}) \\
& \leq \frac{\delta}{1-\delta} + \frac{\delta}{100} \quad (\text{Corollary 9}),
\end{aligned}$$

proving the theorem.  $\blacktriangleleft$

It remains to prove Lemma 28. We will use a lemma in [33] enabling us to approximately sample distributions.

► **Lemma 29** (Lemma 36 in [33]). *Let  $D$  be a distribution on  $[m]$ . Suppose that given  $i \in [m]$  we can compute in time polynomial in  $O(\log m)$  the cumulative distribution  $\Pr[D \leq i]$ . Then there is a  $\text{polylog}(mt)$ -time computable function  $f$  such that given any  $t \geq 1$ ,  $f$  uses  $s = \lceil \log(mt) \rceil$  bits to sample an element from the support of  $D$  such that  $\|f(U_s) - D\|_1 \leq 1/t$ .*

We will also bound above the binomial coefficients in terms of the entropy function.

► **Remark 30.** For every  $\rho > 0$  we have

$$\log \binom{n}{\lceil n(1/2 + \rho) \rceil} \leq (1 + o(1))n \cdot H(1/2 + \rho).$$

We now prove the lemma, by giving an appropriate sampling procedure:

**Proof of Lemma 28.** Let  $X'_\delta$  as the distribution over  $\{0, 1\}^n$ , where the coordinates are independent and each coordinate is 1 with probability  $1/2 + \delta/2$  and 0 otherwise. Our sampling procedure below will sample a distribution  $D$  over  $\{0, 1\}^n$  that is close to  $X'_\delta$  (over  $\{0, 1\}^n$ ), then apply  $x_i \mapsto 2x_i - 1$  to each coordinate  $x_i$  of  $D$  to sample the target distribution over  $\{-1, 1\}^n$ .

Consider the following procedure for sampling a string  $x$  from  $X'_\delta$ . First sample the Hamming weight  $i$  of  $x$  according to  $\text{Binomial}(n, 1/2 + \delta/2)$ , where each weight  $i \in \{0, \dots, n\}$  is chosen with probability  $\binom{n}{i}(1/2 + \delta/2)^i(1/2 - \delta/2)^{n-i}$ . Then given  $i \in \{0, \dots, n\}$ , sample  $x$  uniformly from the set of strings with weight exactly  $i$ . By performing both steps in an approximate manner, we obtain  $f$ .

To do this, we apply Lemma 29 to sample the weight  $i$  from a distribution  $D$  (over  $\{0, \dots, n\}$ ) that is within  $\delta/300$  in total variation distance to  $\text{Binomial}(n, 1/2 + \delta/2)$ , which costs  $O(\log n + \log(1/\delta))$  bits. Given  $i \sim D$ , if  $i < \lceil n(1/2 + 0.499\delta) \rceil$  then we return the all 0s string; otherwise, we apply Lemma 29 to sample from the set of strings of Hamming weight  $i \geq \lceil n(1/2 + 0.499\delta) \rceil$ .

As  $D$  is  $(\delta/300)$ -close to  $|X'_\delta|$ , for every sufficiently large  $n$ , we have

$$\Pr[D < \lceil n(1/2 + 0.499\delta) \rceil] \leq \Pr[|X'_\delta| < \lceil n(1/2 + 0.499\delta) \rceil] + \delta/300 < \delta/150.$$

Here, we use Remark 30 to bound the log of the description size of the universe, i.e. the number of strings of some Hamming weight  $i \geq \lceil n(1/2 + 0.499\delta) \rceil$ , by

$$\log \binom{n}{\lceil n(1/2 + 0.499\delta) \rceil} \leq (1 + o(1))H(1/2 + 0.499\delta)n = H(1/2 + 0.499\delta)n + o(n).$$

Furthermore, Haramaty, Lee, and Viola show (in the proof of [33, Lemma 35]) that we can sample from the distribution of strings of length  $n$  with Hamming weight  $i$  in time  $\text{poly}(n)$ . Thus, the total number of random bits required to sample a distribution within  $\delta/100$  in total variation distance to  $X_\delta$  is at most  $s = H(1/2 + 0.499\delta) \cdot n + o(n) + o(n) + O(\log(1/\delta))$ , and  $f$  can be computed in polynomial time as desired.  $\blacktriangleleft$



# Double Balanced Sets in High Dimensional Expanders

Tali Kaufman ✉

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

David Mass ✉

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

---

## Abstract

Recent works have shown that expansion of pseudorandom sets is of great importance. However, all current works on pseudorandom sets are limited only to product (or approximate product) spaces, where Fourier Analysis methods could be applied. In this work we ask the natural question whether pseudorandom sets are relevant in domains where Fourier Analysis methods cannot be applied, e.g., one-sided local spectral expanders.

We take the first step in the path of answering this question. We put forward a new definition for pseudorandom sets, which we call “double balanced sets”. We demonstrate the strength of our new definition by showing that small double balanced sets in one-sided local spectral expanders have very strong expansion properties, such as unique-neighbor-like expansion. We further show that cohomologies in cosystolic expanders are double balanced, and use the newly derived strong expansion properties of double balanced sets in order to obtain an exponential improvement over the current state of the art lower bound on their minimal distance.

**2012 ACM Subject Classification** Theory of computation → Computational complexity and cryptography

**Keywords and phrases** High dimensional expanders, Double balanced sets, Pseudorandom functions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.3

**Category** RANDOM

**Funding** *Tali Kaufman*: Supported by ERC and BSF.

*David Mass*: Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

## 1 Introduction

The study of pseudorandom (or “global”) functions has led to many recent advancements. It has been shown that they possess an effective hypercontractive inequality in many domains such as the  $p$ -biased cube [15], the slice [16], the Grassmann graph [17] and two-sided local spectral expanders [6]. The common observation in all of these works is that while hypercontractivity does not hold for any general function, it holds for a certain subclass of pseudorandom functions. This phenomenon has been the key to many breakthroughs, most famously the resolution of Khot’s 2-to-2 Games Conjecture [17].

While this study of pseudorandom functions has been very fruitful in many domains, currently it is still limited only to domains where Fourier Analysis methods could be applied. These domains are product (or approximate product) spaces, so each function has an orthogonal (or an approximate orthogonal) decomposition. While these domains are enough for a lot of applications, there are many applications that require other domains. Some examples are the recent works on efficient sampling algorithms (e.g., [5, 4, 2, 3] and more). The domains in these works are one-sided local spectral expanders, which inherently do not possess an orthogonal decomposition.



© Tali Kaufman and David Mass;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 3; pp. 3:1–3:17



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this work we make the first step in the study of pseudorandom functions in other domains where Fourier Analysis methods cannot be applied. We put forward an alternative definition for pseudorandom functions, which we call “double balanced sets”. We demonstrate the strength of our new definition by showing that small double balanced sets in one-sided local spectral expanders have very strong expansion properties. We further show that cohomologies in cosystolic expanders are double balanced, and then by the strong expansion properties of double balanced sets, we achieve an exponential improvement over the state of the art lower bound on their minimal distance.

### 1.1 Double balanced sets

In order to present our definition of double balanced sets, we need to set some notations first. A  $d$ -dimensional simplicial complex  $X$  is a  $(d + 1)$ -hypergraph which is closed under inclusions, i.e., if  $\sigma \in X$  then every  $\tau \subseteq \sigma$  is also in  $X$ . A  $k$ -face is a hyperedge of size  $k + 1$  and the set of  $k$ -faces in the complex is denoted by  $X(k)$ . For any face  $\sigma \in X$ , the link of  $\sigma$ , denoted by  $X_\sigma$ , is the subcomplex that is obtained by all the faces that contain  $\sigma$  and then removing  $\sigma$  from all of them.

Let  $f \subseteq X(k)$  be a subset of  $k$ -faces in  $X$ . For any face  $\sigma \in X(\ell)$ ,  $\ell < k$ , we denote by  $f_\sigma \subseteq X_\sigma(k - \ell - 1)$  the *localization* of  $f$  to the link of  $\sigma$ , where a face  $\tau \in X_\sigma(k - \ell - 1)$  is in  $f_\sigma$  if and only if  $\tau \cup \sigma \in f$ . We also denote by  $f^\sigma$  the *restriction* of  $f$  to the link of  $\sigma$ , where  $f^\sigma = f \cap X_\sigma(k)$ . Note that both  $f_\sigma$  and  $f^\sigma$  “live” in the link of  $\sigma$ , but  $f_\sigma$  is a subset of  $(k - \ell - 1)$ -faces whereas  $f^\sigma$  is a subset of  $k$ -faces.

For simplicity, we assume in the introduction that the complex has a uniform probability distribution in every dimension. In the body of the paper we will take into account general probability distributions.

► **Definition 1** (Double balanced sets). *We say that  $f \subseteq X(k)$  is  $\alpha$ -double balanced in dimension  $\ell$ ,  $\ell < k$ , if for every  $\ell$ -face  $\sigma \in X(\ell)$  it holds that*

$$\frac{|f_\sigma|}{|X_\sigma(k - \ell - 1)|} \leq \alpha \mathbb{E}_{v \in \sigma} \left[ \frac{|(f_\sigma \setminus v)^v|}{|X_\sigma(k - \ell)|} \right]. \quad (1)$$

*We say that  $f$  is  $\alpha$ -double balanced if it is  $\alpha$ -double balanced in dimension  $\ell$  for every  $\ell < k$ .*

In order to get some intuition, let us focus on low dimensions first. Let  $X$  be a 3-dimensional complex and  $f \subseteq X(2)$  (i.e., a set of triangles in a complex with pyramids).

- For every vertex  $v \in X(0)$ , the left-hand side of (1) translates to the fraction of triangles in  $f$  that contain  $v$  out of all the triangles that contain  $v$ , and the right-hand side of (1) translates to  $\alpha$  times the fraction of triangles in  $f$  that together with  $v$  form a pyramid out of all the pyramids that contain  $v$ .
- For every edge  $\{u, v\} \in X(1)$ , the left-hand side of (1) translates to the fraction of triangles in  $f$  that contain  $\{u, v\}$  out of all the triangles that contain  $\{u, v\}$ , and the right-hand side of (1) translates to  $\alpha$  times the average fraction of triangles in  $f$  that contain  $u$  or  $v$  and together with  $v$  or  $u$ , respectively, form a pyramid out of all the pyramids that contain  $\{u, v\}$ .

In general, the left-hand side of (1) translates to the fraction of  $k$ -faces in  $f$  that contain  $\sigma$ , and the right-hand side translates to the average fraction of  $k$ -faces in  $f$  that contain  $|\sigma| - 1$  vertices from  $\sigma$  and together with  $\sigma$  forms a  $(k + 1)$ -face.

Let us explain briefly the motivation behind this definition. From a spectral point of view, it is known that high dimensional random walks with intersections do not mix rapidly, whereas random walks without intersections (also known as swap walks [1] or complement

walks [7]) have an optimal mixing rate<sup>1</sup>. Previous works on pseudorandom sets (e.g., [6]) benefit from the optimal mixing rate of non-intersecting random walks, but for that the complex has to be of a very high dimension, i.e., in order to gain anything on a pseudorandom set of dimension  $k$ , the complex has to be of dimension at least  $2k$  (so we can move between  $k$ -faces without intersections). Our definition of double balanced sets benefits from the optimal mixing rate of non-intersecting random walks *even when  $d = k + 1$* . The reason is that the right-hand side of (1), when viewed in the link of  $\sigma \setminus v$  for some vertex  $v \in \sigma$ , is concerned with faces that do not contain  $v$ , i.e., it is related to a non-intersecting random walk inside the link of  $\sigma \setminus v$ .

From a topological point of view, our definition of double balanced sets relates faces of two consecutive dimensions (i.e.,  $(k - \ell - 1)$ -faces in the left-hand side of (1) and  $(k - \ell)$ -faces in the right-hand side), similar to usual topological operators (e.g., the boundary and coboundary operators). In this sense, our definition has the potential to benefit also from the topological properties of the complex. Indeed, we show that cohomologies in high dimensional expanders are double balanced by utilizing the topological expansion of the complex.

To summarize the above discussion, our definition of double balanced sets has the potential to imitate a situation where the complex has many dimensions above (like in previous works) while having only one dimension above. It benefits both from spectral and topological properties of the complex, whereas previous works could only use spectral properties. We believe that utilizing the topological properties of the complex, as well as spectral properties, would lead to many breakthroughs in the future.

## 1.2 Relation to the common definition

We would like to formalize the intuitive similarity of our new definition (of double balanced sets) to the common definition (of pseudorandom sets).

The common definition of pseudorandom sets, as given in [6]<sup>2</sup>, says that a set of  $k$ -faces  $f$  is  $\varepsilon$ -pseudorandom in dimension  $\ell$ ,  $\ell < k$ , if for every  $\ell$ -face  $\sigma \in X(\ell)$  it holds that

$$\frac{|f_\sigma|}{|X_\sigma(k - \ell - 1)|} \leq \varepsilon. \quad (2)$$

As demonstrated in the following lemma, our definition of double balanced sets implies almost pseudorandomness.

► **Lemma 2.** *Let  $X$  be a good enough one-sided local spectral expander<sup>3</sup>. For any  $\alpha$ -double balanced set of  $k$ -faces  $f \in X(k)$  and any dimension  $\ell < k$ , if*

$$\frac{|f|}{|X(k)|} \leq \frac{\varepsilon}{(\ell + 1)\alpha^\ell}$$

*then*

$$\Pr_{\sigma \in X(\ell)} \left[ \frac{|f_\sigma|}{|X_\sigma(k - \ell - 1)|} \leq \varepsilon \right] \geq 1 - \varepsilon \frac{|f|}{|X(k)|}.$$

<sup>1</sup> By random walks with intersections we mean that we move from an  $i$ -face  $\sigma$  to a  $j$ -face  $\tau$  through a  $k$ -face that contain both  $\sigma$  and  $\tau$ , where the intersection  $\sigma \cap \tau$  may be non-empty, whereas random walks without intersections require that  $\sigma \cap \tau$  would be empty.

<sup>2</sup> The actual definition is considered with general functions from  $X(k)$  to  $\mathbb{R}$ . For simplicity we consider only functions from  $X(k)$  to  $\{0, 1\}$ , i.e., functions that correspond to subsets of  $k$ -faces.

<sup>3</sup> The definition of one-sided local spectral expansion will be introduced later in the paper.

### 3:4 Double Balanced Sets in High Dimensional Expanders

In words, for a sufficiently small set, if the set is  $\alpha$ -double balanced then it is also almost pseudorandom, i.e., all  $\ell$ -faces besides of a negligible fraction of them satisfy the pseudorandomness property.

#### 1.3 Inheritance property

An interesting property that applies to double balanced sets is that it is inherited by lower dimensions. We show that a set of  $k$ -faces which is double balanced in dimension  $\ell$  is also double balanced in all dimensions below  $\ell$ . This result is obtained by applying the following lemma step by step.

► **Lemma 3** (Double balance inheritance). *If  $f \subseteq X(k)$  is  $\alpha$ -double balanced in dimension  $\ell$ , then  $f$  is  $\alpha'$ -double balanced in dimension  $\ell - 1$ , where*

$$\alpha' = \frac{\alpha\ell}{\ell + 1 - \alpha}.$$

It is worth to note that when  $f$  is perfectly double balanced, i.e., when  $\alpha = 1$ , then lemma 3 implies that  $f$  is also perfectly double balanced in all dimensions below  $\ell$ . In other words, perfect double balance is inherited by lower dimensions *without any loss*.

#### 1.4 $\delta_1$ -expansion of small double balanced sets

In recent years, a few different notions of high dimensional expansion have been studied. One such notion is  $\delta_1$ -expansion, which can be viewed as a generalization of unique-neighbor expansion in graphs. It is a strong expansion notion that is usually very hard to get. For a set of  $k$ -faces  $f \subseteq X(k)$ ,  $\delta_1(f)$  is defined as the set of  $(k + 1)$ -faces that contain exactly one  $k$ -face from  $f$ . We say that  $f$  is  $\delta_1$ -expanding if

$$\frac{|\delta_1(f)|}{|X(k + 1)|} \geq \varepsilon \frac{|f|}{|X(k)|}. \quad (3)$$

In [14] it has been shown that  $\delta_1$ -expansion for small sets implies group-independent cosystolic expansion, i.e., cosystolic expansion over any group.

In order to demonstrate the strength of our definition of double balanced sets, we show that small double balanced sets are  $\delta_1$ -expanding. On one hand, we show that when a double balanced set  $f$  is sufficiently small, it has a *nearly perfect*  $\delta_1$ -expansion, i.e.,  $\varepsilon$  in equation (3) is very close to  $k + 2$ . On the other hand, for larger double balanced sets (which are still small, but not that small), we show that they have some  $\delta_1$ -expansion, i.e.,  $\varepsilon > 0$  in equation (3). We prove the following two theorems.

► **Theorem 4** (Nearly optimal  $\delta_1$ -expansion for sufficiently small double balanced sets). *Let  $X$  be a good enough one-sided local spectral expander. For any  $\alpha$ -double balanced set of  $k$ -faces  $f \subseteq X(k)$  and  $\varepsilon > 0$ , if*

$$\frac{|f|}{|X(k)|} \leq \frac{\varepsilon}{(k + 1)^2 \alpha^k}$$

*then*

$$\frac{|\delta_1(f)|}{|X(k + 1)|} \geq (1 - 3\varepsilon)(k + 2) \frac{|f|}{|X(k)|}.$$



► **Theorem 5** (Some  $\delta_1$ -expansion for small double balanced sets). *Let  $X$  be a good enough one-sided local spectral expander. For any  $\alpha$ -double balanced set of  $k$ -faces  $f \subseteq X(k)$  and  $\varepsilon > 0$ , if*

$$\frac{|f|}{|X(k)|} \leq \frac{1 - \varepsilon}{(k + 1)\alpha^k}$$

*then*

$$\frac{|\delta_1(f)|}{|X(k + 1)|} > 0.$$

Both of theorems 4 and 5 demonstrate the strength of our definition of double balanced sets. The key idea that since  $f$  is a small set, its double balance property implies that it has to be small in every link as well, which in turn implies  $\delta_1$ -expansion. The novelty over previous works (e.g., [13, 9, 14]) is to benefit from the optimal mixing rate of non-intersecting random walks. As explained in section 1.1, our definition of double balanced sets is related in a sense to non-intersecting random walks and hence benefits from an optimal mixing rate. This is in contrast to previous works, which essentially used only intersecting random walks, and hence could obtain worse bounds and only for much smaller sets.

## 1.5 Application to minimal distance of cohomologies

Cohomologies stand in the center of recent studies in Mathematics, and they have already found some applications in Theoretical Computer Science as well. Complexes with large cohomologies have played a key role in the construction of efficiently decodable quantum LDPC codes with a large distance [10]. It is known by now to construct quantum LDPC codes with a larger distance [20, 18], however these are not known to be efficiently decodable. Complexes with large cohomologies were also the main block in the first construction of explicit 3XOR instances that are hard for the Sum-of-Squares Hierarchy [8]. Other constructions which are hard for more levels of the the Sum-of-Squares Hierarchy [12] are known by now. Nonetheless, the construction of [8] is still the best known construction from *simplicial* complexes and it has been the first step in this line of works.

In order to define cohomologies, let us identify a set of  $k$ -faces in  $X$  with an  $\mathbb{F}_2$ -valued function  $f : X(k) \rightarrow \mathbb{F}_2$  and denote by  $C^k(X)$  the space of all  $\mathbb{F}_2$ -valued functions on  $X(k)$ . The coboundary operator  $\delta^k : C^k(X) \rightarrow C^{k+1}(X)$  is defined by

$$\delta^k f(\sigma) = \sum_{u \in \sigma} f(\sigma \setminus \{u\}) \bmod 2.$$

The image of  $\delta^{k-1}$  is called the  $k$ -coboundaries and is denoted by

$$B^k(X) = \{\delta^{k-1} f \mid f \in C^{k-1}(X)\}.$$

The kernel of  $\delta^k$  is called the  $k$ -cocycles and is denoted by

$$Z^k(X) = \{f \in C^k(X) \mid \delta^k f = \mathbf{0}\}.$$

It is not hard to check that  $B^k(X) \subseteq Z^k(X) \subseteq C^k(X)$ . The  $k$ -cohomology of  $X$  is the quotient space  $H^k(X) = Z^k(X)/B^k(X)$ .

Previous works could only obtain complexes with some constant lower bound on the size of their cohomologies [13, 9, 14]. We show that for high dimensional expanders (in a topological sense), all of their cohomology elements are double balanced. We then utilize the  $\delta_1$ -expansion of double balanced sets in order to obtain a lower bound on their size, achieving an *exponential* improvement upon the current state of the art.

► **Theorem 6** (Cohomologies are double balanced). *For a complex whose links are topological expanders, every  $k$ -cohomology element is  $((k+1)/\beta)$ -double balanced, where  $\beta$  is the expansion constant in the links of the complex.*

► **Theorem 7** (Lower bound on cohomology elements). *For a good enough one-sided local spectral expander whose links are topological expanders, every  $k$ -cohomology element must be of density at least  $\beta^k/(k+1)!$ , where  $\beta$  is the expansion constant in the links of the complex.*

► **Remark.** The current state of the art lower bound on the size of cohomologies prior to this work is  $\approx (\beta^k/k!)^{2^k}$  [14, Lemma 3.10].

## 1.6 Organization

In section 2 we provide the required preliminaries. In section 3 we introduce the formal definition of double balanced sets and prove its inheritance property. In section 4 we show that small double balanced sets in one-sided local spectral expanders have the strong  $\delta_1$ -expansion property, and also explain how to prove lemma 2. In section 5 we show that cohomologies in a complex with topological expanding links are double balanced, obtaining an exponential improvement upon the current state of the art lower bound on their minimal distance.

## 2 Preliminaries

### 2.1 Simplicial complexes

Recall that a  $d$ -dimensional simplicial complex  $X$  is a downwards closed  $(d+1)$ -hypergraph. A  $k$ -face of  $X$  is a hyperedge of size  $k+1$ , and the set of  $k$ -faces of  $X$  is denoted by  $X(k)$ . An assignment of values from  $\mathbb{F}_2$  to the  $k$ -faces,  $k \leq d$ , is called a  $k$ -cochain, and the space of all  $k$ -cochains over  $\mathbb{F}_2$  is denoted by  $C^k(X)$ .

Any assignment to the  $k$ -faces  $f \in C^k(X)$  induces an assignment to the  $(k+1)$ -faces by the coboundary operator  $\delta$ . For any  $(k+1)$ -face  $\sigma = \{v_0, \dots, v_{k+1}\}$ ,  $\delta(f)(\sigma)$  is defined by

$$\delta(f)(\sigma) = \sum_{i=0}^{k+1} f(\sigma \setminus \{v_i\}) \pmod{2}.$$

The kernel of the coboundary operator is called the  $k$ -cocycles and denoted by

$$Z^k(X) = \{f \in C^k(X) \mid \delta(f) = \mathbf{0}\}.$$

The image of  $\delta$  is called the  $k$ -coboundaries and denote by

$$B^k(X) = \{\delta(f) \mid f \in C^{k-1}(X)\}.$$

One can check that  $\delta(\delta(f)) = \mathbf{0}$  always holds, hence  $B^k(X) \subseteq Z^k(X) \subseteq C^k(X)$ . The quotient space  $Z^k(X)/B^k(X)$  is called the  $k$ -cohomologies and denoted by  $H^k(X)$ .

For a  $d$ -dimensional simplicial complex  $X$ , let  $P_d : X(d) \rightarrow \mathbb{R}_{\geq 0}$  be a probability distribution over the  $d$ -faces of the complex. For simplicity, we will assume in this work that  $P_d$  is the uniform distribution. This probability distribution over the  $d$ -faces induces a probability distribution  $P_k$  for every dimension  $k < d$  by selecting a  $d$ -face  $\sigma_d$  according to  $P_d$  and then selecting a  $k$ -face  $\sigma_k \subset \sigma_d$  uniformly at random.

The weight of any  $k$ -cochain  $f \in C^k(X)$  is defined by

$$\|f\| = \Pr_{\sigma_k \sim P_k} [f(\sigma_k) \neq 0],$$

i.e., the (weighted) fraction of non-zero elements in  $f$ . The distance between two  $k$ -cochains  $f, g \in C^k(X)$  is defined as  $\text{dist}(f, g) = \|f - g\|$ .

We also add a useful definition of a mutual weight of two cochains. For  $\ell < k$  and two cochains  $f \in C^k(X)$ ,  $g \in C^\ell(X)$  we define their mutual weight by

$$\|(f, g)\| = \Pr_{\sigma_k \sim P_k, \sigma_\ell \subset \sigma_k} [f(\sigma_k) \neq 0 \wedge g(\sigma_\ell) \neq 0],$$

where  $\sigma_k$  is chosen according to the distribution  $P_k$  and  $\sigma_\ell$  is an  $\ell$ -face chosen uniformly from  $\sigma_k$  (i.e.,  $\sigma_\ell$  is chosen according to  $P_\ell$  conditioned on  $\sigma_k$  being chosen).

## 2.2 Cosystolic and coboundary expansion

Coboundary expansion has been introduced by Linial and Meshulam [19] and independently by Gromov [11]. It is a generalization of edge expansion of graphs to higher dimensions.

► **Definition 8** (Coboundary expansion). *A  $d$ -dimensional simplicial complex  $X$  is said to be an  $\varepsilon$ -coboundary expander if for every  $k < d$  and  $f \in C^k(X) \setminus B^k(X)$  it holds that*

$$\frac{\|\delta(f)\|}{\text{dist}(f, B^k(X))} \geq \varepsilon,$$

where  $\text{dist}(f, B^k(X)) = \min\{\text{dist}(f, g) \mid g \in B^k(X)\}$ .

Cosystolic expansion is similar to coboundary expansion, with the main difference that it can have non-trivial cohomologies as long as they are large.

► **Definition 9** (Cosystolic expansion). *A  $d$ -dimensional simplicial complex  $X$  is said to be an  $(\varepsilon, \mu)$ -cosystolic expander if for every  $k < d$ :*

1. *For any  $f \in C^k(X) \setminus Z^k(X)$  it holds that*

$$\frac{\|\delta(f)\|}{\text{dist}(f, Z^k(X))} \geq \varepsilon,$$

where  $\text{dist}(f, Z^k(X)) = \min\{\text{dist}(f, g) \mid g \in Z^k(X)\}$ .

2. *For any  $f \in Z^k(X) \setminus B^k(X)$  it holds that  $\|f\| \geq \mu$ .*

## 2.3 Links, localization and restriction

For every face  $\sigma \in X$ , its local view, also called its *link*, is a  $(d - |\sigma| - 1)$ -dimensional simplicial complex defined by  $X_\sigma = \{\tau \setminus \sigma \mid \sigma \subseteq \tau \in X\}$ . The probability distribution over the top faces of  $X_\sigma$  is induced from the probability distribution of  $X$ , where for any top face  $\tau \in X_\sigma(d - |\sigma| - 1)$ , its probability is the probability to choose  $\sigma \cup \tau$  in  $X$  conditioned on choosing  $\sigma$ . Since we assume in this work that the probability distribution over the top faces of  $X$  is the uniform distribution, it follows that the probability distribution over the top faces of  $X_\sigma$  is the uniform distribution.

For any  $k$ -cochain  $f \in C^k(X)$  and an  $\ell$ -face  $\sigma \in X(\ell)$ , the *localization* of  $f$  to the link of  $\sigma$  is a  $(k - \ell - 1)$ -cochain in the link of  $\sigma$ ,  $f_\sigma \in C^{k-\ell-1}(X_\sigma)$  defined by

$$f_\sigma(\tau) = f(\sigma \cup \tau).$$

The *restriction* of  $f$  to the link of  $\sigma$  is a  $k$ -cochain in the link of  $\sigma$ ,  $f^\sigma \in C^k(X_\sigma)$  defined by

$$f^\sigma(\tau) = f(\tau).$$

## 2.4 Local spectral expansion

Another notion of high dimensional expansion, called *local spectral expansion* is concerned with the spectral properties of the links of the complex.

► **Definition 10** (Two-sided local spectral expansion). *A  $d$ -dimensional simplicial complex  $X$  is called a  $\lambda$ -two-sided local spectral expander,  $\lambda > 0$ , if for every  $k \leq d - 2$  and  $\sigma \in X(k)$ , the underlying graph<sup>4</sup> of  $X_\sigma$  is a  $\lambda$ -two-sided spectral expander, i.e., its spectrum is bounded from above by  $\lambda$  and from below by  $-\lambda$ .*

► **Definition 11** (One-sided local spectral expansion). *A  $d$ -dimensional simplicial complex  $X$  is called a  $\lambda$ -one-sided local spectral expander,  $\lambda > 0$ , if for every  $k \leq d - 2$  and  $\sigma \in X(k)$ , the underlying graph<sup>4</sup> of  $X_\sigma$  is a  $\lambda$ -one-sided spectral expander, i.e., its spectrum is bounded from above by  $\lambda$ .*

## 2.5 Minimal and locally minimal cochains

One of the technical notions we use in this work is the notion of a minimal cochain. We say that a  $k$ -cochain  $f \in C^k(X)$  is *minimal* if its weight cannot be reduced by adding a coboundary to it, i.e., for every  $g \in B^k(X)$  it holds that  $\|f\| \leq \|f - g\|$ . Recall that the distance of  $f$  from the coboundaries is defined by  $\text{dist}(f, B^k(X)) = \min\{\|f - g\| \mid g \in B^k(X)\}$ . Since  $0 \in B^k(X)$ , it follows that for every  $f \in C^k(X)$ ,  $\|f\| \geq \text{dist}(f, B^k(X))$ . Thus,  $f$  is said to be *minimal* if and only if  $\|f\| = \text{dist}(f, B^k(X))$ .

We also define the notion of a locally minimal cochain, where we say that  $f \in C^k(X)$  is *locally minimal* if for every vertex  $v$ , the localization of  $f$  to the link of  $v$  is minimal in the link, i.e.,  $f_v$  is minimal in  $X_v$  for every  $v \in X(0)$ . It is not hard to check that any minimal cochain is also locally minimal.

## 3 Double balanced sets

We start by providing the formal definition of a double balanced cochain. Recall that for any  $k$ -cochain  $f \in C^k(X)$  and a vertex  $u \in X(0)$ , we denote by  $f^u$  the restriction of  $f$  to the  $k$ -faces in the link of  $u$ , i.e.,  $f^u \in C^k(X_u)$ .

► **Definition 12** (Double balanced cochains). *Let  $X$  be a  $d$ -dimensional simplicial complex. A  $k$ -cochain  $f \in C^k(X)$  is said to be  $\alpha$ -double balanced in dimension  $\ell$ , where  $\alpha \geq 1$  and  $0 \leq \ell \leq k - 1$ , if for every  $\ell$ -face  $\sigma \in X(\ell)$  it holds that*

$$\|f_\sigma\| \leq \alpha \mathbb{E}_{u \in \sigma} \|(f_{\sigma \setminus u})^u\|.$$

*$f$  is said to be  $\alpha$ -double balanced if  $f$  is  $\alpha$ -double balanced in dimension  $\ell$  for every  $\ell < k$ .*

### 3.1 Balance inheritance

An interesting property that applies to double balanced cochains is that it is inherited by lower dimensions. We show that a cochain of  $k$ -faces which is double balanced in dimension  $\ell$  is also double balanced in all dimensions below  $\ell$ . We prove lemma 3 from the introduction, which we restate here for convenience.

---

<sup>4</sup> The graph whose vertices are  $X_\sigma(0)$  and its edges are  $X_\sigma(1)$ .

► **Lemma 13** (Double balance inheritance). *Let  $f \in C^k(X)$  be an  $\alpha$ -double balanced cochain in dimension  $\ell$ . Then  $f$  is  $\alpha'$ -double balanced in dimension  $\ell - 1$ , where*

$$\alpha' = \frac{\alpha\ell}{\ell + 1 - \alpha}.$$

**Proof.** Let  $\tau \in X(\ell - 1)$ .

$$\begin{aligned} \|f_\tau\| &= \mathbb{E}_{u \in X_\tau(0)} [\|f_{\tau u}\|] \\ &\leq \mathbb{E}_{u \in X_\tau(0)} \left[ \alpha \mathbb{E}_{v \in \tau u} [\|(f_{\tau u \setminus v})^v\|] \right] \\ &= \mathbb{E}_{u \in X_\tau(0)} \left[ \frac{\alpha}{\ell + 1} \|(f_\tau)^u\| + \frac{\alpha\ell}{\ell + 1} \mathbb{E}_{v \in \tau} [\|(f_{\tau u \setminus v})^v\|] \right] \\ &= \frac{\alpha}{\ell + 1} \mathbb{E}_{u \in X_\tau(0)} [\|(f_\tau)^u\|] + \frac{\alpha\ell}{\ell + 1} \mathbb{E}_{v \in \tau} \left[ \mathbb{E}_{u \in X_\tau(0)} [\|(f_{\tau u \setminus v})^v\|] \right] \\ &= \frac{\alpha}{\ell + 1} \|f_\tau\| + \frac{\alpha\ell}{\ell + 1} \mathbb{E}_{v \in \tau} \|(f_{\tau \setminus v})^v\|, \end{aligned}$$

where the inequality follows since  $f$  is  $\alpha$ -double balanced in dimension  $\ell$  and all the other steps follow from laws of probability. This implies that

$$\|f_\tau\| \leq \frac{\alpha\ell}{\ell + 1 - \alpha} \mathbb{E}_{v \in \tau} \|(f_{\tau \setminus v})^v\|. \quad \blacktriangleleft$$

It is worth to note that when  $f$  is perfectly double balanced, i.e., when  $\alpha = 1$ , then lemma 13 implies that  $f$  is also perfectly double balanced in all dimensions below  $\ell$ . In other words, perfect double balance is inherited by lower dimensions *without any loss*.

► **Corollary 14.** *Let  $f \in C^k(X)$  be a 1-double balanced cochain in dimension  $\ell$ . Then  $f$  is also 1-double balanced in all dimensions below  $\ell$ .*

#### 4 $\delta_1$ -expansion for small double balanced sets

In this section we show that small double balanced sets are  $\delta_1$ -expanding. On one hand, we show that when a double balanced set  $f$  is sufficiently small, it has a *nearly optimal*  $\delta_1$ -expansion. On the other hand, for larger double balanced sets (which are still small, but not that small), we show that they have some  $\delta_1$ -expansion, i.e.,  $\|\delta_1(f)\| > 0$ . We prove theorems 4 and 5 from the introduction, which we restate here in a formal way.

► **Theorem 15** (Nearly optimal  $\delta_1$ -expansion for sufficiently small double balanced sets). *For every  $d \geq 2$ ,  $\alpha \geq 1$  and  $0 < \varepsilon < 1$  there exists  $\lambda = \lambda(d, \alpha, \varepsilon)$  such that the following holds: Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander. For any  $k$ -cochain  $f \in C^k(X)$ ,  $1 \leq k < d$ , such that  $f$  is  $\alpha$ -double balanced and  $\|f\| \leq \frac{\varepsilon}{(k+1)^2 \alpha^k}$  it holds that*

$$\|\delta_1(f)\| \geq (k+2)(1-3\varepsilon) \|f\|.$$

► **Theorem 16** (Some  $\delta_1$ -expansion for small double balanced sets). *For every  $d \geq 2$ ,  $\alpha \geq 1$  and  $0 < \varepsilon < 1$  there exists  $\lambda = \lambda(d, \alpha, \varepsilon)$  such that the following holds: Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander. For any  $k$ -cochain  $f \in C^k(X)$ ,  $1 \leq k < d$ , such that  $f$  is  $\alpha$ -double balanced and  $\|f\| \leq \frac{1-\varepsilon}{(k+1)\alpha^k}$  it holds that*

$$\|\delta_1(f)\| > 0.$$

We split the proof of these theorems to two parts. In the first part we show that if almost all of the  $(k-1)$ -faces of a cochain are not dense then its  $\delta_1$  is optimal. In the second part, we show that for sufficiently small double balanced cochains, almost all of their  $(k-1)$ -faces are indeed not dense.

#### 4.1 Part I – Bound $\delta_1(f)$ by the dense $(k-1)$ -faces

Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander and  $0 < \eta < 1$  a density constant.

For any  $k$ -cochain  $f \in C^k(X)$  we define the set of dense  $(k-1)$ -faces by

$$\text{DENSE}_{k-1} = \{\sigma \in X(k-1) \mid \|f_\sigma\| > \eta\}.$$

We show in this section that  $\|\delta_1(f)\|$  can be bounded by the fraction of dense  $(k-1)$ -faces.

► **Proposition 17.** *Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander and  $0 < \eta < 1$  a density constant. For any  $k$ -cochain  $f \in C^k(X)$ ,  $1 \leq k < d$ ,*

$$\|\delta_1(f)\| \geq (k+2) \|f\| \left( 1 - (k+1) \left( \lambda + \eta + \frac{\|\text{DENSE}_{k-1}\|}{\|f\|} \right) \right).$$

The proof of this proposition will follow from the following two lemmas. The first lemma holds for any simplicial complex.

► **Lemma 18.** *Let  $X$  be a  $d$ -dimensional simplicial complex. For any  $k$ -cochain  $f \in C^k(X)$ ,  $1 \leq k < d$ ,*

$$\|\delta_1(f)\| \geq (k+2) \left( \frac{1}{2} \sum_{\sigma \in X(k-1)} \|(\delta_1(f_\sigma), \sigma)\| - k \sum_{\sigma \in X(k-1)} \|(\delta_2(f_\sigma), \sigma)\| \right)$$

**Proof.** Denote by  $\delta_i(f)$  the set of  $(k+1)$ -faces that contain exactly  $i$   $k$ -faces from  $f$ . Summing  $\delta_1(f_\sigma)$  in the links of all  $\sigma \in X(k-1)$  equals

$$\sum_{\sigma \in X(k-1)} \|(\delta_1(f_\sigma), \sigma)\| = \sum_{i=1}^{k+1} \frac{i(k+2-i)}{\binom{k+2}{2}} \|\delta_i(f)\|. \quad (4)$$

Summing  $\delta_2(f_\sigma)$  in the links of all  $\sigma \in X(k-1)$  equals

$$\sum_{\sigma \in X(k-1)} \|(\delta_2(f_\sigma), \sigma)\| = \sum_{i=2}^{k+2} \frac{\binom{i}{2}}{\binom{k+2}{2}} \|\delta_i(f)\|. \quad (5)$$

Multiplying (5) by  $2k$  yields

$$2k \sum_{\sigma \in X(k-1)} \|(\delta_2(f_\sigma), \sigma)\| = \sum_{i=2}^{k+2} \frac{i(i-1)k}{\binom{k+2}{2}} \|\delta_i(f)\| \geq \sum_{i=2}^{k+2} \frac{i(k+2-i)}{\binom{k+2}{2}} \|\delta_i(f)\|. \quad (6)$$

Subtracting (6) from (4) yields

$$\sum_{\sigma \in X(k-1)} \|(\delta_1(f_\sigma), \sigma)\| - 2k \sum_{\sigma \in X(k-1)} \|(\delta_2(f_\sigma), \sigma)\| \leq \frac{2}{k+2} \|\delta_1(f)\|.$$

Multiplying both sides by  $(k+2)/2$  finishes the proof. ◀

The following lemma holds for any  $\lambda$ -one-sided local spectral expander.

► **Lemma 19.** *Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander and  $0 < \eta < 1$  a density constant. For any  $k$ -cochain  $f \in C^k(X)$ ,  $1 \leq k < d$ ,*

1.  $\sum_{\sigma \in X^{(k-1)}} \|(\delta_1(f_\sigma), \sigma)\| \geq 2(1 - \lambda - \eta) \|f, \text{SPARSE}_{k-1}\|,$
  2.  $\sum_{\sigma \in X^{(k-1)}} \|(\delta_2(f_\sigma), \sigma)\| \leq \|f, \text{DENSE}_{k-1}\| + (\lambda + \eta) \|f, \text{SPARSE}_{k-1}\|,$
- where  $\text{SPARSE}_{k-1} = X^{(k-1)} \setminus \text{DENSE}_{k-1}$ .

**Proof.** Since  $X$  is a one-sided local spectral expander,  $f_\sigma$  is a subset of vertices in  $X_\sigma$  so both inequalities follow immediately from the well known Cheeger inequality. ◀

We can now prove Proposition 17.

**Proof of Proposition 17.** Since

$$\|f\| = \|f, \text{DENSE}_{k-1}\| + \|f, \text{SPARSE}_{k-1}\|,$$

lemma 19(1) yields

$$\sum_{\sigma \in X^{(k-1)}} \|(\delta_1(f_\sigma), \sigma)\| \geq 2(1 - \lambda - \eta) \|f\| - 2 \|f, \text{DENSE}_{k-1}\|, \quad (7)$$

and lemma 19(2) yields

$$\sum_{\sigma \in X^{(k-1)}} \|(\delta_2(f_\sigma), \sigma)\| \leq (\lambda + \eta) \|f\| + \|f, \text{DENSE}_{k-1}\|. \quad (8)$$

Substituting (7) and (8) in lemma 18 finishes the proof. ◀

## 4.2 Part II – Bound the fraction of dense $(k - 1)$ -faces

We show in this section that for every double balanced and small cochain in a good enough one-sided local spectral expander, the fraction of dense  $(k - 1)$ -faces is very small.

We first extend the definition of dense faces to every dimension  $-1 \leq i \leq k - 1$ . Given a density constant  $0 < \eta < 1$  and  $\varepsilon > 0$ , we set  $\eta_{k-1} = \eta$  and for every  $0 \leq i \leq k - 1$  we define

$$\eta_{i-1} = \frac{\eta_i}{\alpha} - \frac{\varepsilon}{(k+1)^2 \alpha^{k-i}}.$$

We then define the dense faces in dimension  $i$  to be

$$\text{DENSE}_i = \{\sigma \in X(i) \mid \|f_\sigma\| > \eta_i\}.$$

Our goal in this subsection is to prove the following proposition.

► **Proposition 20.** *Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander,  $1 \leq k < d$  any dimension,  $\alpha \geq 1$  a balance constant,  $0 < \eta < 1$  a density constant and  $\varepsilon > 0$ . For any  $k$ -cochain  $f \in C^k(X)$  such that  $f$  is  $\alpha$ -double balanced and  $\|f\| \leq \eta_{-1}$  it holds that*

$$\|\text{DENSE}_{k-1}\| \leq 3k! \left( \frac{(k+1)^3 \alpha^k \lambda}{\varepsilon} \right)^2 \|f\|$$

We start by showing that in a  $\lambda$ -one-sided local spectral expander, the restriction of a cochain to almost every vertex is seen with the right proportion.

### 3:12 Double Balanced Sets in High Dimensional Expanders

► **Lemma 21.** *Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander. For any  $k$ -cochain  $f \in C^k(X)$ ,  $0 \leq k < d$ , and  $\varepsilon > 0$  it holds that*

$$\Pr_{u \in X(0)} [\|f^u\| > \|f\| + \varepsilon] \leq \left( \frac{(k+1)\lambda}{\varepsilon} \right)^2 \|f\|.$$

**Proof.** Define the following graph  $G = (V, E)$ , where  $V = X(k)$ , i.e., all  $k$ -faces of  $X$ , and  $E = \{\{\sigma_1, \sigma_2\} \mid \exists u \in X(0) \text{ s.t. } \sigma_1 \cup u, \sigma_2 \cup u \in X(k+1)\}$ , i.e., there is an edge between  $\sigma_1$  and  $\sigma_2$  if and only if there exists some vertex in  $X$  that completes both  $\sigma_1$  and  $\sigma_2$  to a  $(k+1)$ -face.

We define a probability distribution on  $G$  that corresponds to the probability distribution of  $X$  as follows:

- The probability of a vertex  $\sigma \in V$  equals to the probability of the corresponding  $k$ -face  $\sigma \in X(k)$ .
- The probability of an edge  $\{\sigma_1, \sigma_2\} \in E$  equals  $\mathbb{E}_{u \in X(0)} \Pr[\sigma_1 \cup u \mid u] \cdot \Pr[\sigma_2 \cup u \mid u]$ , where all the probabilities are taken according to the complex  $X$ .

Since  $X$  is a  $\lambda$ -one-sided local spectral expander, by [7, Claim 4.9]  $G$  is a  $((k+1)\lambda)^2$ -spectral expander, because its adjacency operator is a two steps walk of the 0, 2-complement walk of [7].

Now, define  $\mu : X(0) \rightarrow \mathbb{R}$  by  $\mu(u) = \|f^u\| = \Pr[\sigma \in f \mid \sigma \cup u \in X(k+1)]$ . The following holds by laws of probability:

$$\mathbb{E}_{u \in X(0)} [\mu(u)] = \mathbb{E}_{u \in X(0)} \Pr[\sigma \in f \mid \sigma \cup u \in X(k+1)] = \Pr[\sigma \in f] = \|f\|. \quad (9)$$

$$\begin{aligned} \mathbb{E}_{u \in X(0)} [\mu(u)^2] &= \mathbb{E}_{u \in X(0)} \Pr[\sigma_1 \in f \mid \sigma_1 \cup u \in X(k+1)] \cdot \Pr[\sigma_2 \in f \mid \sigma_2 \cup u \in X(k+1)] \\ &= \Pr_{\{\sigma_1, \sigma_2\} \in E} [\sigma_1 \in f \wedge \sigma_2 \in f] = \|E(f)\|, \end{aligned} \quad (10)$$

where  $E(f)$  is the set of edges  $\{\sigma_1, \sigma_2\}$  in  $G$  such that both  $\sigma_1$  and  $\sigma_2$  are in  $f$ . Since  $G$  is a  $((k+1)\lambda)^2$ -spectral expander, it follows that  $\|E(f)\| \leq \|f\|^2 + ((k+1)\lambda)^2 \|f\|$ . Substituting in (10) and combining (9) yields

$$\text{Var}_{u \in X(0)} [\mu(u)] = \mathbb{E}_{u \in X(0)} [\mu(u)^2] - \mathbb{E}_{u \in X(0)} [\mu(u)]^2 \leq ((k+1)\lambda)^2 \|f\|.$$

Now, by Chebyshev's inequality

$$\Pr [\|f^u\| > \|f\| + \varepsilon] = \Pr [\mu(u) > \mathbb{E}[\mu] + \varepsilon] \leq \frac{\text{Var}[\mu]}{\varepsilon^2} \leq \left( \frac{(k+1)\lambda}{\varepsilon} \right)^2 \|f\|.$$

This completes the proof. ◀

In the next lemma we show that for every dimension  $i$ , if  $f$  is double balanced in dimension  $i$  then the fraction of dense  $i$ -faces is not much more than the fraction of dense  $(i-1)$ -faces.

► **Lemma 22.** *Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander and  $f \in C^k(X)$ ,  $0 \leq k < d$ . For every  $0 \leq i < k$ , if  $f$  is  $\alpha$ -double balanced in dimension  $i$  then*

$$\|DENSE_i\| \leq (i+1) \|DENSE_{i-1}\| + (i+1) \left( \frac{(k+1-i)(k+1)^2 \alpha^{k-i} \lambda}{\varepsilon} \right)^2 \|f\|.$$



**Proof.** Note that for every  $\sigma \in \text{DENSE}_i$  there must exist a vertex  $u \in \sigma$  such that

$$\|(f_{\sigma \setminus u})^u\| > \frac{\eta_i}{\alpha}, \quad (11)$$

since otherwise

$$\|f_\sigma\| \leq \frac{\alpha}{i+1} \sum_{u \in \sigma} \|(f_{\sigma \setminus u})^u\| \leq \eta_i$$

and  $\sigma \notin \text{DENSE}_i$ .

For every  $\sigma \in \text{DENSE}_i$ , fix one  $(i-1)$ -face  $\tau(\sigma) = \sigma \setminus u$  that satisfies (11). By laws of probability

$$\begin{aligned} \|\text{DENSE}_i\| &= \Pr[\sigma_i \in \text{DENSE}_i] = (i+1) \Pr[\sigma_i \in \text{DENSE}_i \wedge \sigma_{i-1} = \tau(\sigma_i)] \leq \\ &(i+1) \|\text{DENSE}_{i-1}\| + (i+1) \Pr[\sigma_i \in \text{DENSE}_i \wedge \sigma_{i-1} = \tau(\sigma_i) \mid \tau(\sigma_i) \notin \text{DENSE}_{i-1}], \end{aligned} \quad (12)$$

where the inequality holds by splitting to the two cases whether  $\tau(\sigma_i) \in \text{DENSE}_{i-1}$ .

We focus now on the right summand of (12) which is the case where  $\tau(\sigma_i) \notin \text{DENSE}_{i-1}$ . Recall that  $\tau(\sigma_i)$  satisfies (11). Thus, we can bound the probability of this event by the probability to choose a sparse  $(i-1)$ -face and then a vertex such that (11) holds, i.e.,

$$\Pr[\sigma_i \in \text{DENSE}_i \wedge \sigma_{i-1} = \tau(\sigma_i) \mid \tau(\sigma_i) \notin \text{DENSE}_{i-1}] \leq \mathbb{E}_{\tau \in \text{SPARSE}_{i-1}} \Pr_{u \in X_\tau(0)} \left[ \|(f_\tau)^u\| > \frac{\eta_i}{\alpha} \right]. \quad (13)$$

Since  $\tau \in \text{SPARSE}_{i-1}$ , it holds that  $\|f_\tau\| \leq \eta_{i-1}$ . Thus,

$$\begin{aligned} \mathbb{E}_{\tau \in \text{SPARSE}_{i-1}} \Pr_{u \in X_\tau(0)} \left[ \|(f_\tau)^u\| > \frac{\eta_i}{\alpha} \right] &\leq \\ &\mathbb{E}_{\tau \in \text{SPARSE}_{i-1}} \Pr_{u \in X_\tau(0)} \left[ \|(f_\tau)^u\| > \|f_\tau\| + \frac{\varepsilon}{(k+1)^2 \alpha^{k-i}} \right], \end{aligned} \quad (14)$$

where the inequality holds since

$$\|f_\tau\| + \frac{\varepsilon}{(k+1)^2 \alpha^{k-i}} \leq \eta_{i-1} + \frac{\varepsilon}{(k+1)^2 \alpha^{k-i}} = \frac{\eta_i}{\alpha}.$$

Combining (12), (13) and (14) yields

$$\begin{aligned} \|\text{DENSE}_i\| &\leq \\ &(i+1) \|\text{DENSE}_{i-1}\| + (i+1) \mathbb{E}_{\tau \in \text{SPARSE}_{i-1}} \Pr_{u \in X_\tau(0)} \left[ \|(f_\tau)^u\| > \|f_\tau\| + \frac{\varepsilon}{(k+1)^2 \alpha^{k-i}} \right] \leq \\ &(i+1) \|\text{DENSE}_{i-1}\| + (i+1) \mathbb{E}_{\tau \in \text{SPARSE}_{i-1}} \left[ \left( \frac{(k+1-i)(k+1)^2 \alpha^{k-i} \lambda}{\varepsilon} \right)^2 \|f_\tau\| \right] \leq \\ &(i+1) \|\text{DENSE}_{i-1}\| + (i+1) \left( \frac{(k+1-i)(k+1)^2 \alpha^{k-i} \lambda}{\varepsilon} \right)^2 \|f\|, \end{aligned}$$

where the second inequality follows by lemma 21. This completes the proof.  $\blacktriangleleft$

We can now prove Proposition 20.

**Proof of Proposition 20.** We apply lemma 22 for  $i = k - 1, k - 2, \dots, 0$  step by step.

$$\begin{aligned}
\|\text{DENSE}_{k-1}\| &\leq k \|\text{DENSE}_{k-2}\| + k \left( \frac{2(k+1)^2 \alpha \lambda}{\varepsilon} \right)^2 \|f\| \leq \\
&k(k-1) \|\text{DENSE}_{k-3}\| + \left( k(k-1) \left( \frac{3(k+1)^2 \alpha^2 \lambda}{\varepsilon} \right)^2 + k \left( \frac{2(k+1)^2 \alpha \lambda}{\varepsilon} \right)^2 \right) \|f\| \\
&\leq \dots \leq k! \|\text{DENSE}_{-1}\| + \left( k! \left( \frac{(k+1)^3 \alpha^k \lambda}{\varepsilon} \right)^2 + \dots + k \left( \frac{2(k+1)^2 \alpha \lambda}{\varepsilon} \right)^2 \right) \|f\| \\
&= \sum_{i=0}^{k-1} \frac{k!}{i!} \left( \frac{(k+1-i)(k+1)^2 \alpha^{k-i} \lambda}{\varepsilon} \right)^2 \|f\| \\
&\leq k! \left( \frac{(k+1)^2 \alpha^k \lambda}{\varepsilon} \right)^2 \sum_{i=0}^{k-1} \frac{(k+1-i)^2}{i!} \|f\| \\
&\leq 3k! \left( \frac{(k+1)^3 \alpha^k \lambda}{\varepsilon} \right)^2 \|f\|
\end{aligned}$$

where the equality holds since  $\|f_\emptyset\| = \|f\| \leq \eta_{-1}$ , i.e., the empty set is not dense, and hence  $\|\text{DENSE}_{-1}\| = 0$ . The rest of the inequalities are just calculations. This completes the proof.  $\blacktriangleleft$

### 4.3 Proof of Theorems 15 and 16

**Proof of Theorem 15.** Let  $\lambda \leq \frac{\varepsilon}{d^3 \alpha^{d-1}} \sqrt{\frac{\varepsilon}{3d!}}$  and  $\eta = \frac{\varepsilon}{(k+1)}$ . By simple calculation

$$\eta_{-1} = \frac{\varepsilon}{(k+1)^2 \alpha^k}.$$

Thus, since  $\|f\| \leq \eta_{-1}$ , Proposition 20 implies that

$$\|\text{DENSE}_{k-1}\| \leq 3k! \left( \frac{(k+1)^3 \alpha^k \lambda}{\varepsilon} \right)^2 \leq \frac{\varepsilon}{k+1} \|f\|. \quad (15)$$

Substituting (15) in Proposition 17 finishes the proof.  $\blacktriangleleft$

**Proof of Theorem 16.** Let  $\lambda \leq \frac{\varepsilon}{d^3 \alpha^{d-1}} \sqrt{\frac{\varepsilon}{(d+1)!}}$  and  $\eta = \frac{1 - \varepsilon/(k+1)}{(k+1)}$ . By simple calculation

$$\eta_{-1} = \frac{1 - \varepsilon}{(k+1) \alpha^k}.$$

Thus, since  $\|f\| \leq \eta_{-1}$ , Proposition 20 implies that

$$\|\text{DENSE}_{k-1}\| \leq 3k! \left( \frac{(k+1)^3 \alpha^k \lambda}{\varepsilon} \right)^2 \leq \frac{\varepsilon}{(k+2)(k+1)} \|f\|. \quad (16)$$

Substituting (16) in Proposition 17 finishes the proof.  $\blacktriangleleft$

We conclude this section by noting that the proof of lemma 2 from the introduction is exactly the same as the proof of Proposition 20, with the only difference that we start by setting  $\eta_\ell = \varepsilon$  and bound the fraction of dense  $\ell$ -faces instead of the dense  $(k-1)$ -faces.

## 5 Cohomologies are double balanced

Previous works could only obtain complexes with some constant lower bound on the size of their cohomologies [13, 9, 14]. We show that for high dimensional expanders (in a topological sense), all of their cohomology elements are double balanced. We then utilize the  $\delta_1$ -expansion of double balanced sets in order to obtain a lower bound on their size, achieving an *exponential* improvement upon the current state of the art.

We start by proving Theorem 6 from the introduction, which we restate here in a formal way.

► **Theorem 23** (Cohomologies are double balanced). *Let  $X$  be a  $d$ -dimensional complex such that every non-trivial link in  $X$  is a  $\beta$ -coboundary expander. For every  $\ell < k < d$ , any  $k$ -cohomology element is  $\frac{\ell+1}{\beta}$ -double balanced in dimension  $\ell$ .*

**Proof.** Let  $f \in H^k(X)$  be a  $k$ -cohomology and  $\sigma \in X(\ell)$  be an  $\ell$ -face. Consider a  $(k-\ell)$ -face  $\tau \in \delta(f_\sigma)$ . Let us denote  $\sigma = \{v_0, v_1, \dots, v_\ell\}$  and  $\tau = \{v_{\ell+1}, v_{\ell+2}, \dots, v_{k+1}\}$ . By definition

$$\sum_{i=\ell+1}^{k+1} f(\sigma \cup \tau \setminus v_i) = \sum_{i=\ell+1}^{k+1} f_\sigma(\tau \setminus v_i) \neq 0,$$

where the inequality holds since  $\tau \in \delta(f_\sigma)$ . Since  $f$  is a  $k$ -cohomology, it holds that

$$\sum_{i=0}^{k+1} f(\sigma \cup \tau \setminus v_i) = 0.$$

Therefore, there must exist  $0 \leq j \leq \ell$  such that  $f(\sigma \cup \tau \setminus v_j) \neq 0$ . By definition of restriction and localization, it means that

$$(f_{\sigma \setminus v_j})^{v_j}(\tau) = (f_{\sigma \setminus v_j})(\tau) \neq 0.$$

In other words, for every  $\tau \in \delta(f_\sigma)$ , there exists a vertex  $v \in \sigma$  such that  $\tau \in (f_{\sigma \setminus v})^v$ . It follows that

$$\|\delta(f_\sigma)\| \leq \sum_{v \in \sigma} \|(f_{\sigma \setminus v})^v\|. \quad (17)$$

Now, since  $f$  is a  $k$ -cohomology,  $f$  is minimal and hence also locally minimal. The  $\beta$ -coboundary expansion of the links implies that

$$\|\delta(f_\sigma)\| \geq \beta \|f_\sigma\|. \quad (18)$$

Combining (17) and (18) implies that

$$\|f_\sigma\| \leq \frac{1}{\beta} \sum_{v \in \sigma} \|(f_{\sigma \setminus v})^v\| = \frac{\ell+1}{\beta} \mathbb{E}_{v \in \sigma} \|(f_{\sigma \setminus v})^v\|.$$

This complete the proof. ◀

We conclude by proving Theorem 7 from the introduction, which we restate here in a formal way.

► **Theorem 24** (Lower bound on cohomology elements). *For every  $d \geq 2$ ,  $\beta > 0$  and  $\varepsilon > 0$  there exists  $\lambda = \lambda(d, \beta, \varepsilon)$  such that the following holds. Let  $X$  be a  $d$ -dimensional  $\lambda$ -one-sided local spectral expander such that every non-trivial link in  $X$  is a  $\beta$ -coboundary expander. For every  $k < d$ , any  $k$ -cohomology element  $f \in H^k(X)$  satisfies*

$$\|f\| \geq \frac{(1 - \varepsilon)\beta^k}{(k + 1)!}.$$

**Proof.** Assume towards contradiction that there exists  $f \in H^k(X)$  with  $\|f\| < \frac{(1 - \varepsilon)\beta^k}{(k + 1)!}$ . By Theorem 23,  $f$  is  $((\ell + 1)/\beta)$ -double balanced in dimension  $\ell$  for every  $\ell < k$ . Then Theorem 16 implies<sup>5</sup> that  $\|\delta_1(f)\| > 0$  in contradiction to  $f$  being a cohomology elements (i.e.,  $\delta(f) = 0$ ). ◀

---

## References

- 1 V. L. Alev, F. G. Jeronimo, and M. Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–201, 2019.
- 2 V. L. Alev and L. C. Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1198–1211, 2020.
- 3 N. Anari, V. Jain, F. Koehler, H. T. Pham, and T-D. Vuong. Entropic independence in high-dimensional expanders: Modified log-sobolev inequalities for fractionally log-concave polynomials and the ising model. *arXiv preprint*, 2021. [arXiv:2106.04105](#).
- 4 N. Anari, K. Liu, and S. Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *arXiv preprint*, 2020. [arXiv:2001.00303](#).
- 5 N. Anari, K. Liu, S. Oveis Gharan, and C. Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1–12, 2019.
- 6 M. Bafna, M. Hopkins, T. Kaufman, and S. Lovett. Hypercontractivity on High Dimensional Expanders. *arXiv preprint*, 2021. [arXiv:2111.09444](#).
- 7 Y. Dikstein and I. Dinur. Agreement testing theorems on layered set systems. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1495–1524. IEEE, 2019.
- 8 I. Dinur, Y. Filmus, P. Harsha, and M. Tulsiani. Explicit SoS lower bounds from high-dimensional expanders. *arXiv preprint*, 2020. [arXiv:2009.05218](#).
- 9 S. Evra and T. Kaufman. Bounded degree cosystolic expanders of every dimension. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 36–48, 2016.
- 10 S. Evra, T. Kaufman, and G. Zemor. Decodable quantum LDPC codes beyond the  $\sqrt{n}$  distance barrier using high dimensional expanders. In *61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2020. to appear.
- 11 M. Gromov. Singularities, Expanders and Topology of Maps. Part 2: from Combinatorics to Topology Via Algebraic Isoperimetry. *Geometric And Functional Analysis*, 20(2):416–526, 2010.
- 12 M. Hopkins and T-C. Lin. Explicit Lower Bounds Against  $\Omega(n)$ -Rounds of Sum-of-Squares. *arXiv preprint*, 2022. [arXiv:2204.11469](#).

---

<sup>5</sup> It is implied by the proof of Theorem 16 if we consider for every dimension  $\ell$  its own double balance constant  $(\ell + 1)/\beta$  rather than bounding all constants by the largest one.

- 13 T. Kaufman, D. Kazhdan, and A. Lubotzky. Ramanujan Complexes and Bounded Degree Topological Expanders. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 484–493, 2014.
- 14 T. Kaufman and D. Mass. Unique-Neighbor-Like Expansion and Group-Independent Cosystolic Expansion. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
- 15 P. Keevash, N. Lifshitz, E. Long, and D. Minzer. Hypercontractivity for global functions and sharp thresholds. *arXiv preprint*, 2019. [arXiv:1906.05568](#).
- 16 S. Khot, D. Minzer, D. Moshkovitz, and M. Safra. Small set expansion in the johnson graph. In *Electron. Colloquium Comput. Complex.*, volume 25, page 78, 2018.
- 17 S. Khot, D. Minzer, and M. Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601. IEEE, 2018.
- 18 A. Leverrier and G. Zémor. Quantum tanner codes. *arXiv preprint*, 2022. [arXiv:2202.13641](#).
- 19 N. Linial and R. Meshulam. Homological connectivity of random 2-complexes. *Combinatorica*, 26(4):475–487, 2006.
- 20 P. Panteleev and G. Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. *arXiv preprint*, 2021. [arXiv:2111.03654](#).



# Fast and Perfect Sampling of Subgraphs and Polymer Systems

Antonio Blanca ✉

Department of Computer Science and Engineering,  
Pennsylvania State University, University Park, PA, USA

Sarah Cannon ✉

Department of Mathematical Sciences, Claremont McKenna College, CA, USA

Will Perkins ✉

Department of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

---

## Abstract

We give an efficient perfect sampling algorithm for weighted, connected induced subgraphs (or *graphlets*) of rooted, bounded degree graphs. Our algorithm utilizes a vertex-percolation process with a carefully chosen rejection filter and works under a percolation subcriticality condition. We show that this condition is optimal in the sense that the task of (approximately) sampling weighted rooted graphlets becomes impossible in finite expected time for infinite graphs and intractable for finite graphs when the condition does not hold. We apply our sampling algorithm as a subroutine to give near linear-time perfect sampling algorithms for polymer models and weighted non-rooted graphlets in finite graphs, two widely studied yet very different problems. This new perfect sampling algorithm for polymer models gives improved sampling algorithms for spin systems at low temperatures on expander graphs and unbalanced bipartite graphs, among other applications.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Random walks and Markov chains

**Keywords and phrases** Random Sampling, perfect sampling, graphlets, polymer models, spin systems, percolation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.4

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2202.05907> [6]

**Funding** *Sarah Cannon*: Supported in part by NSF grants DMS-1803325 and CCF-2104795.

*Will Perkins*: Supported in part by NSF grant DMS-1847451.

**Acknowledgements** This work was carried out as part of the AIM SQuaRE workshop “Connections between computational and physical phase transitions.” We thank Tyler Helmuth, Alexandre Stauffer, and Izabella Stuhl for many helpful conversations.

## 1 Introduction

Sampling is a fundamental computational task: given a specification of a probability distribution on a (large) set of combinatorial objects, output a random object with the specified distribution or with a distribution close to the specified distribution. This task becomes challenging when the specification of the distribution is much more succinct than the set of objects, and one wants to sample using time and space commensurate with the specification. Fundamental examples include sampling from Markov random fields and probabilistic graphical models and sampling substructures of graphs. We will address both of these examples here and connect them in a new way.



© Antonio Blanca, Sarah Cannon, and Will Perkins;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 4; pp. 4:1–4:18



Leibniz International Proceedings in Informatics  
LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We consider a natural sampling problem: given a bounded-degree graph  $G$ , sample a *graphlet* (a connected, vertex-induced subgraph) of  $G$  containing a fixed vertex  $r$  with probability proportional to an exponential in the size of the subgraph. That is, sample a graphlet  $S$  containing vertex  $r$  with probability proportional to  $\lambda^{|S|}$ , where  $\lambda > 0$  is a distribution parameter and  $|S|$  denotes the number of vertices in  $S$ . In this paper we are concerned with small values of  $\lambda$ , where the expected size of a sampled graphlet is much smaller than the size of the graph.

Sampling graphlets is an important task in data science, network analysis, bioinformatics, and sociology, as it allows us to gain information about massive graphs from small sections of it; see, e.g., [38, 28, 41, 4]. A number of variants of the problem have consequently been studied, including sampling graphlets of a given size uniformly at random or sampling weighted graphlets of all sizes [46, 5, 37, 15, 10, 11, 43, 1, 42, 45, 12, 9]. The variant we consider here, i.e., sample a graphlet  $S$  with probability proportional to  $\lambda^{|S|}$ , arises as a key subroutine in recent sampling algorithms for spin systems (hard-core model, Ising model, Potts model, etc.) in the regime of strong interactions via *polymer models* described below in Section 1.2; see [32, 13, 40, 25, 7, 36, 35, 14, 17].

One major limitation of previous sampling algorithms for graphlets and polymer models (those in, e.g., [32, 40, 45, 16, 24], among others) is the use of exhaustive enumeration of graphlets of a given size; this requires restrictive parameter regimes or large polynomial running times, with the maximum degree  $\Delta$  of the graph appearing in the exponent of the polynomial. Here we design a fast perfect sampling algorithm for weighted graphlets based on a vertex percolation process combined with a rejection filter. This method bypasses the enumeration barrier and allows us to design perfect sampling algorithms for a number of applications, substantially improving upon existing algorithms in three ways: 1) our algorithms have considerably faster running times, with no dependence on  $\Delta$  in the exponent; 2) our algorithms return perfect, rather than approximate, samples from the desired distributions; and 3) our algorithms are conceptually simple and practical to implement.

Our algorithm proceeds as follows. First, run a vertex percolation process on the graph  $G$  beginning at vertex  $r$  in a breadth-first search manner, repeatedly adding each adjacent vertex to the graphlet with a carefully-chosen probability  $p$ . Once the percolation process terminates, the graphlet is accepted as the random sample with a certain probability that depends on the graphlet and rejected otherwise; if the graphlet is rejected, the algorithm restarts another percolation process from  $r$ . Because of the careful way we choose the percolation and rejection probabilities, we can prove the final accepted sample is drawn exactly from the desired distribution and the expected running time is bounded by a constant that depends only on  $\lambda$  and the maximum degree  $\Delta$ .

## 1.1 Sampling rooted graphlets

Our key contribution is a new algorithm for perfectly sampling weighted graphlets containing a given vertex  $r$ . Formally, let  $G = (V, E)$  be a finite or infinite graph of maximum degree  $\Delta$ . For  $r \in V$ , let  $\mathcal{S}(G, r)$  be the set of all connected, vertex-induced subgraphs of  $G$  containing  $r$ . (The subgraph induced by  $U \subseteq V$  has vertex set  $U$  and includes all the edges of  $G$  with both endpoints in  $U$ .) We call  $r$  the root of  $G$  and the elements of  $\mathcal{S}(G, r)$  graphlets rooted at  $r$ . For  $\lambda > 0$  define the probability distribution  $\nu_{G,r,\lambda}$  on  $\mathcal{S}(G, r)$  by

$$\nu_{G,r,\lambda}(S) = \frac{\lambda^{|S|}}{Z_{G,r,\lambda}}, \quad \text{where} \quad Z_{G,r,\lambda} = \sum_{\hat{S} \in \mathcal{S}(G,r)} \lambda^{|\hat{S}|}. \quad (1)$$



The distribution is well defined when the normalizing constant  $Z_{G,r,\lambda}$ , known as the partition function, is finite. This is the case for every graph of maximum degree  $\Delta$  and every  $r$  when  $\lambda$  is below the critical threshold:

$$\lambda_*(\Delta) = \frac{(\Delta - 2)^{\Delta-2}}{(\Delta - 1)^{\Delta-1}}; \quad (2)$$

(see Lemma 3 below). We give an efficient perfect sampling algorithm for  $\nu_{G,r,\lambda}$  for  $\lambda < \lambda_*(\Delta)$ .

► **Theorem 1.** *Fix  $\Delta \geq 3$  and let  $\lambda < \lambda_*(\Delta)$ . There is a randomized algorithm that for any graph  $G = (V, E)$  of maximum degree  $\Delta$  and any  $r \in V$  outputs a graphlet distributed according to  $\nu_{G,r,\lambda}$  with expected running time bounded by a constant that depends only on  $\Delta$  and  $\lambda$ .*

We assume a model of computation that allows for querying the adjacency list of a given vertex in a bounded degree graph in constant time, the standard model used in the study of sublinear algorithms [27]. We also assume access to a stream of perfectly random real numbers in  $[0, 1]$ . The model of computation is chosen for consistency; in particular, our methods extend to other models, only requiring to adjust the running time to account for any additional computational overhead.

Previous algorithms to generate  $\varepsilon$ -approximate samples from  $\nu_{G,r,\lambda}$  (e.g., those in [32, 45, 16, 24]) exhaustively enumerate all graphlets of size  $\leq k$ , for some  $k$  that depends on the error parameter  $\varepsilon$  that describes how accurate the sample must be. This results in algorithms with  $(1/\varepsilon)^{O(\log \Delta)}$  running times. Applications such as sampling from polymer models require multiple samples from  $\nu_{G,r,\lambda}$  and, consequently, have small error tolerance per sample; in particular, they require  $\varepsilon \ll 1/n$ , which results in inefficient algorithms with overall running time  $n^{O(\log \Delta)}$ . The algorithm in Theorem 1, on the other hand, is an exact sampler whose expected running time depends only on  $\Delta$  and  $\lambda$  and thus provides a significant advantage in applications as we detail below.

We also show that Theorem 1 is sharp in two ways. First, we establish that there is no polynomial-time approximate sampling algorithm for  $\nu_{G,r,\lambda}$  when  $\lambda > \lambda_*(\Delta)$  for the class of graphs of maximum degree at most  $\Delta$  unless  $\text{RP}=\text{NP}$ . Second, in the infinite setting, the normalizing constant  $Z_{G,r,\lambda}$  may diverge (and consequently the distribution  $\nu_{G,r,\lambda}$  is not well-defined) when  $\lambda > \lambda_*(\Delta)$ ; conversely, we prove that  $Z_{G,r,\lambda}$  is finite on every graph of maximum degree  $\Delta$  when  $\lambda \leq \lambda_*(\Delta)$ .

► **Lemma 2.** *If for every finite graph  $G = (V, E)$  of maximum degree  $\Delta \geq 3$  and every  $r \in V$ , there is a polynomial-time approximate sampler for  $\nu_{G,r,\lambda}$  when  $\lambda > \lambda_*(\Delta)$ , then  $\text{RP}=\text{NP}$ .*

► **Lemma 3.** *The partition function  $Z_{G,r,\lambda}$  is finite for every (possibly infinite) graph  $G = (V, E)$  of maximum degree  $\Delta$  and every  $r \in V$  if and only if  $\lambda \leq \lambda_*(\Delta)$ .*

The proofs of these lemmas are omitted, but can be found in the full version of this paper [6].

Finally, we mention that the algorithmic result in Theorem 1 cannot be extended even to the case  $\lambda = \lambda_*(\Delta)$ : for the infinite  $\Delta$ -regular tree, we can show that the expected size of a graphlet sampled from  $\nu_{G,r,\lambda}$  is infinite when  $\lambda = \lambda_*(\Delta)$ , and so it is impossible to have sampling algorithms with finite expected running time. In summary, the algorithm in Theorem 1 for  $\lambda < \lambda_*(\Delta)$ , combined with the hardness/impossibility results in Lemmas 2 and 3 for  $\lambda > \lambda_*(\Delta)$ , provide a resolution to the computational problem of sampling from  $\nu_{G,r,\lambda}$  on graphs of maximum degree at most  $\Delta$ .

As mentioned, our sampling algorithm is based on exploring the connected component of  $r$  in a vertex-percolation process. We carefully choose a specific percolation parameter  $p \in (0, 1)$  as a function of  $\lambda$  and  $\Delta$  (see Lemma 9). We then perform breadth-first search (BFS) from  $r$ , labeling each new vertex encountered “active” with probability  $p$  and “inactive” with probability  $1 - p$  independently over all vertices; we continue the BFS exploring only unexplored neighbors of active vertices. In this way we uncover the “active” component of  $r$ , call it  $\gamma$ . We then accept  $\gamma$  with a given probability depending on  $\lambda$ ,  $\Delta$ ,  $|\gamma|$  and  $|\partial\gamma|$ , where  $\partial\gamma$  denotes the set of vertices outside of  $\gamma$  that are adjacent to  $\gamma$ . If we reject  $\gamma$ , we begin again with a new percolation process. We note that only when  $\lambda < \lambda_*(\Delta)$  does there exist a suitable percolation probability  $p$  that results in a subcritical percolation process, so that the size of the active component has finite expectation and exponential tails. Sampling a graphlet by exploring a random component and performing a rejection step has been used in the past (most notably in the recent work of Bressan [9] to sample uniformly random graphlets of size  $k$ ; see also [2]). The weighted model we sample from is particularly well suited to this type of exploration algorithm because of the direct connection to a subcritical percolation process.

We prove a more general version of Theorem 1 in Section 2, allowing for vertex-labeled graphlets and modifications of the weights by multiplication by a non-negative function bounded by 1. These generalizations are needed for the application to polymer models in Section 1.2.

## 1.2 Sampling from polymer models

We use our algorithm for sampling weighted rooted graphlets to design fast and perfect samplers for polymer models. Polymer models are systems of interacting geometric objects representing defects from pure ground states (i.e., most likely configurations) in spin systems on graphs in classical statistical physics [29, 39, 22]. These geometric objects are most often represented by vertex-labeled graphlets from a given host graph. Recently, polymer models have found application as an algorithmic tool to sample from spin systems on various classes of graphs in strong interaction regimes; see, e.g., [32, 13, 40, 16, 31, 24, 25, 7, 36, 35, 14, 20, 17]. In these applications, the problem of sampling weighted vertex-labeled rooted graphlets emerged as a significant computational barrier.

We will work with *subset polymer models* in which all polymers are vertex-labeled graphlets from a host graph  $G = (V, E)$ . These models were defined in [29] and generalized in [39]. Such a polymer model consists of:

- A set  $\mathcal{C} = \mathcal{C}(G)$  of polymers, each of which is a graphlet in  $G$  with the vertices of the graphlet labeled with colors from a set  $\Sigma$  of size  $q$ .
- Weights  $w_\gamma \geq 0$  for each  $\gamma \in \mathcal{C}$ .
- An incompatibility relation  $\approx$  defined by connectivity. We say two polymers  $\gamma, \gamma' \in \mathcal{C}$  are *incompatible* and write  $\gamma \approx \gamma'$  if the union of their corresponding vertices induces a connected subgraph in  $G$ . Otherwise they are *compatible* and we write  $\gamma \sim \gamma'$ .

Let  $\Omega(\mathcal{C})$  denote the set of all sets of pairwise compatible polymers from  $\mathcal{C}$ . The polymer model is the Gibbs distribution  $\mu$  on  $\Omega(\mathcal{C})$  defined by

$$\mu(X) = \frac{\prod_{\gamma \in X} w_\gamma}{Z(\mathcal{C})}, \quad \text{where} \quad Z(\mathcal{C}) = \sum_{X \in \Omega(\mathcal{C})} \prod_{\gamma \in X} w_\gamma$$

is the polymer model partition function. We say the weights of a polymer model are *computationally feasible* if  $w_\gamma$  can be computed in time polynomial in  $|\gamma|$ . The size  $|\gamma|$  of a polymer  $\gamma$  is the number of vertices in the corresponding graphlet.

■ **Table 1** Comparison of conditions and running times of known polymer sampling algorithms.

| Condition                   | Bound on exponential decay of weights          | Running Time         | Type of Sampler |
|-----------------------------|------------------------------------------------|----------------------|-----------------|
| Kotecký–Preiss [32, 35, 13] | $w_\gamma \leq (e^2 q \Delta)^{- \gamma }$     | $n^{O(\log \Delta)}$ | approximate     |
| Polymer Sampling [16, 24]   | $w_\gamma \leq (e^5 q^3 \Delta^3)^{- \gamma }$ | $O(n \log n)$        | approximate     |
| Clique dynamics [23]        | $w_\gamma \leq (eq \Delta)^{- \gamma }$        | $n^{O(\log \Delta)}$ | approximate     |
| This work (Theorem 4)       | $w_\gamma \leq (eq \Delta)^{- \gamma }$        | $O(n \log n)$        | perfect         |

We will assume without loss of generality that all vertex-labeled graphlets of  $G$ , including each individual vertex  $v \in V$ , are elements of  $\mathcal{C}$ , by setting  $w_\gamma = 0$  when necessary. We let  $\mathcal{C}_v$  be all polymers containing vertex  $v$ .

Algorithms for sampling polymer models fall into two classes: those based on truncating the so-called cluster expansion of a polymer model to approximate a partition function and using self-reducibility to sample, and those based on Markov chains on the set of collections of compatible polymers. The cluster expansion approach, while giving polynomial-time algorithms, generally is relatively inefficient, with the degree of the polynomial bound on the running time growing with the degree of the underlying graph; e.g., running time  $n^{O(\log \Delta)}$  in  $n$ -vertex graph of maximum degree  $\Delta$ . The Markov chain approach in principle can be much faster (near linear time in the size of the graph) but runs into one hitch: a stricter condition on the parameters of the model is needed to perform one update step of the Markov chain (the “polymer sampling condition” in [16, 24]). We solve this problem by adapting our rooted graphlet sampler to sample polymers models, leading to a near linear-time perfect sampling algorithm for polymer models under the least restrictive conditions known (see Table 1).

► **Theorem 4.** *Consider a subset polymer model on a family of  $n$ -vertex graphs of maximum degree  $\Delta$  with computationally feasible weights satisfying:*

$$w_\gamma \leq \lambda^{|\gamma|} \text{ for all } \gamma \in \mathcal{C} \text{ where } \lambda < \lambda_*(\Delta, q) := \frac{(\Delta - 2)^{\Delta-2}}{q(\Delta - 1)^{\Delta-1}}. \quad (3)$$

Suppose further that for all vertex  $v$ ,

$$\sum_{\gamma \not\ni v} |\gamma| w_\gamma < 1 + \sum_{\gamma \in \mathcal{C}_v} w_\gamma. \quad (4)$$

Then, there is a perfect sampling algorithm for  $\mu$  with expected running time  $O(n \log n)$ .

The threshold defined in (3) is the generalization of the critical threshold for rooted graphlet sampling to the labeled case (taking  $q = 1$  recovers the definition in (2)). Theorem 4 improves upon the known results for sampling from polymer models in two ways. For a very general class of polymer models, our algorithm simultaneously provides *perfect sampling* with *near-linear running time* under weak conditions on the polymer weights. We now review previous works to illustrate these improvements; see the accompanying Table 1.

A number of conditions on polymer weights have been used to provide efficient sampling algorithms. The first papers in this direction (including [32, 35, 13]) used the Kotecký–Preiss condition for convergence of the cluster expansion of the polymer model partition function [39]:  $\sum_{\gamma' \sim \gamma} w_{\gamma'} e^{|\gamma'|} \leq |\gamma| \quad \forall \gamma \in \mathcal{C}$ . This condition is typically verified by ensuring that:

$$\sum_{\gamma \ni v} w_\gamma e^{|\gamma|} \leq 1 \quad \forall v \in V. \quad (5)$$

Since the number of vertex-labeled rooted graphlets of size  $k$  in a maximum degree  $\Delta$  graph grows roughly like  $(eq\Delta)^{k-1}$  (see [8]), weights of polymers of size  $k$  must decay roughly like  $(e^2q\Delta)^{-k}$  for the polymer model to satisfy (5), with the extra factor of  $e$  coming from the exponential in the left hand side of the condition (5).

The major downside to the algorithms based on the cluster expansion, i.e., those using (5) or the Kotecký–Preiss condition, is that the running times obtained are of the form  $n^{O(\log \Delta)}$ . Subsequent works, namely [16, 24], addressed this downside but at the cost of a significantly stricter condition on the polymer weights.

In [16], the authors devised a new Markov chain algorithm for sampling from polymer models. The condition on the polymer weights for rapid mixing of this chain is somewhat less restrictive than the Kotecký–Preiss condition; it is the Polymer Mixing condition:

$$\sum_{\gamma' \sim \gamma} |\gamma'| w_{\gamma'} \leq \theta |\gamma| \quad \forall \gamma \in \mathcal{C} \text{ for some } \theta \in (0, 1). \quad (6)$$

This requires weights of polymers of size  $k$  to decay like  $(eq\Delta)^{-k}$ , a savings of a factor  $e$  in the base of the exponent over (5). However, to implement a single step of this Markov chain in constant expected time, a stronger condition (the Polymer Sampling condition) was required:

$$w_\gamma \leq (e^5 \Delta^3 q^3)^{-|\gamma|}. \quad (7)$$

This is a significant loss of a factor  $e^3 \Delta^2 q^2$  in the base of the exponent compared to (5), but the resulting sampling algorithm does run in near linear time.

In [23], the authors use a different Markov chain condition, the Clique Dynamics condition, similar to (6), which requires weights of polymers of size  $k$  to decay like  $(eq\Delta)^{-k}$ , saving the same factor  $e$  over (5). Their running times, though, are again of the form  $n^{O(\log \Delta)}$  since implementing one step of their Markov chain involves enumerating rooted polymers of size  $O(\log n)$ .

Our results are a “best-of-both-worlds” for polymer sampling: under the conditions (3) and (4) that both require polymer weights to decay like  $(eq\Delta)^{-k}$  – the precise conditions are similar to but slightly less restrictive than the polymer mixing condition (6) – we obtain a near linear time algorithm. Moreover, unlike any of the previous results, our algorithm is a perfect sampler.

To conclude this section, we comment briefly on the algorithm we design to sample from  $\mu$ . Our starting point is the polymer dynamics Markov chain from [16]. We use it to implement a Coupling from the Past (CFTP) algorithm (see [44]). To do so efficiently (in terms of the number of steps of the Markov chain), we design a new “bounding Markov chain” for the polymer dynamics, a method pioneered in [33, 30], and to implement each step of the Markov chain efficiently, we turn to our sampler for weighted rooted graphlets from Theorem 1.

### 1.3 Applications to spin systems

Our new algorithm for sampling subset polymer models can be used as a subroutine in essentially all previous applications of polymer models for spin system sampling at low temperatures, including those in [35, 13, 40, 16, 31, 24, 25, 14, 20, 17]. This results in faster sampling algorithms under less restrictive conditions on model parameters in all those settings. As examples, we fleshed out here the details in two of these applications; more details are provided in the full version of this paper [6].

**Hard-core model on bipartite graphs.** The hard-core model on a graph  $G$  is the probability distribution  $\mu_G^{hc}$  on  $\mathcal{I}(G)$ , the set of all independent sets of  $G$ , with

$$\mu_G^{hc}(I) = \frac{\lambda^{|I|}}{Z_G^{hc}(\lambda)}, \quad \text{where} \quad Z_G^{hc}(\lambda) = \sum_{I \in \mathcal{I}(G)} \lambda^{|I|}. \quad (8)$$

The complexity of approximate counting and sampling from  $\mu_G^{hc}$  on bounded-degree graphs is well understood: there is a computational threshold at some  $\lambda_c(\Delta)$ , with efficient algorithms for  $\lambda < \lambda_c(\Delta)$  [49, 3, 18, 19] and hardness above the threshold (no polynomial-time algorithms unless NP=RP) [47, 26, 48]. However, on bipartite graphs, the complexity of these problems is unresolved and is captured by the class #BIS (approximately counting independent sets in bipartite graphs) defined by Dyer, Goldberg, Greenhill, and Jerrum [21].

Theorem 4 implies the existence of a fast perfect sampling algorithm for the hard-core model in a certain class of bipartite graphs called *unbalanced bipartite graphs*, considered in [13, 23].

► **Corollary 5.** *There is a perfect sampling algorithm for  $\mu_G^{hc}$  running in expected time  $O(n \log n)$  for  $n$ -vertex bipartite graphs  $G$  with bipartition  $(L, R)$ , with maximum degree  $\Delta_L$  in  $L$ , maximum degree  $\Delta_R$  in  $R$ , and minimum degree  $\delta_R$  in  $R$  if*

$$\lambda(1 + (1 + e)(\Delta_L - 1)\Delta_R) < (1 + \lambda)^{\delta_R/\Delta_L}. \quad (9)$$

Approximate sampling algorithms with large polynomial run times were previously given for this problem when  $6\lambda\Delta_L\Delta_R < (1 + \lambda)^{\delta_R/\Delta_L}$  in [13] and when  $3.3353\lambda\Delta_L\Delta_R < (1 + \lambda)^{\delta_R/\Delta_L}$  in [23]. Our result applies to a comparable parameter range: inequality (9) holds, for instance, when  $(1 + e)\lambda\Delta_L\Delta_R < (1 + \lambda)^{\delta_R/\Delta_L}$ , or when  $3\lambda\Delta_L\Delta_R < (1 + \lambda)^{\delta_R/\Delta_L}$  and  $\Delta_L < 6$ . More importantly, our algorithm is the first to achieve perfect sampling and near-linear running time.

**Potts model on expander graphs.** The  $Q$ -color ferromagnetic Potts model on a graph  $G = (V, E)$  is the probability distribution  $\mu_G^{potts}$  on the set of  $Q$ -colorings of the vertices of  $G$ ; i.e.,  $\{1, \dots, Q\}^V$ . Each  $Q$ -coloring  $\sigma$  is assigned probability  $\mu_G^{potts}(\sigma) \propto e^{\beta m(G, \sigma)}$ , where  $m(G, \sigma)$  is the number of monochromatic edges of  $G$  under the coloring  $\sigma$  and  $\beta > 0$  is a model parameter. When the parameter  $\beta$  is large, and  $G$  has some structure (e.g.,  $G$  is an expander graph), typical configurations drawn from  $\mu_G^{potts}$  are dominated by one of the  $Q$  colors; that is, there is phase coexistence in the model. This enables sampling using subset polymer models.

Recall that an  $n$ -vertex graph  $G = (V, E)$  is an  $\alpha$ -expander if for all subsets  $S \subseteq V$  with  $|S| \leq n/2$ , the number of edges in  $E$  with exactly one endpoint in  $S$  is at least  $\alpha|S|$ .

► **Corollary 6.** *Consider the  $Q$ -color ferromagnetic Potts model on an  $\alpha$ -expander  $n$ -vertex graph of maximum degree  $\Delta$ . Suppose*

$$\beta \geq \frac{1 + \log\left(\frac{\Delta+1}{e\Delta} + 1\right) + \log((Q-1)\Delta)}{\alpha}. \quad (10)$$

*Then there is a sampling algorithm with expected running time  $O(n \log n)$  that outputs a sample  $\sigma$  with distribution  $\hat{\mu}$  so that  $\|\hat{\mu} - \mu_G^{potts}\|_{TV} \leq e^{-\Omega(n)}$ .*

Previously, [16] provided a  $\varepsilon$ -approximate sample for  $\mu_G^{potts}$  in time  $O(n \log(n/\varepsilon) \log(1/\varepsilon))$  whenever  $\beta \geq \frac{5+3\log((Q-1)\Delta)}{\alpha}$ . Condition (10) holds when  $\beta \geq \frac{1.2+\log((Q-1)\Delta)}{\alpha}$ , so our algorithm applies to a larger range of parameters and removes the dependence on  $\varepsilon$  from the running time. We do not achieve perfect sampling in this application only because the subset polymer models used give approximations of  $\mu_G^{potts}$  rather than describing  $\mu_G^{potts}$  exactly.

## 1.4 Sampling unrooted graphlets in finite graphs

As another application of our algorithm for sampling weighted rooted graphlets, we consider next the problem of sampling weighted *unrooted* graphlets in a finite graph. Given a finite graph  $G$ , let  $\mathcal{S}(G)$  be the set of all graphlets of  $G$ . Define the distribution  $\nu_{G,\lambda}$  on  $\mathcal{S}(G)$  by

$$\nu_{G,\lambda}(\gamma) = \frac{\lambda^{|\gamma|}}{Z_{G,\lambda}}, \quad \text{where } Z_{G,\lambda} = \sum_{\gamma \in \mathcal{S}(G)} \lambda^{|\gamma|}.$$

Read-McFarland and Štefankovič [45] gave a polynomial-time approximate sampling algorithm for  $\nu_{G,\lambda}$  for the class of maximum-degree  $\Delta$  graphs when  $\lambda < \lambda_*(\Delta)$  and prove that there is no such algorithm for  $\lambda \in (\lambda_*(\Delta), 1)$  unless  $\text{NP}=\text{RP}$ <sup>1</sup>. We give a new algorithm for this problem, covering the entire  $\lambda < \lambda_*(\Delta)$  regime, and improving on the result of [45] in two ways: (i) our running time is constant in expectation (with no dependence on  $n$ ), while the running time of the  $\varepsilon$ -approximate sampler in [45] is  $n \cdot (1/\varepsilon)^{O(\log \Delta)}$ ; and (ii) our algorithm outputs a perfect sample instead of an approximate one (and thus the running time has no dependence on any approximation parameter).

► **Theorem 7.** *Fix  $\Delta \geq 3$  and let  $\lambda < \lambda_*(\Delta)$ . Then for the class of finite graphs of maximum degree  $\Delta$  there is a randomized algorithm running in constant expected time that outputs a perfect sample from  $\nu_{G,\lambda}$ . The expected running time is bounded as a function of  $\Delta$  and  $\lambda$ .*

The algorithm we use for this theorem is a modification of the one for sampling rooted graphlets. We pick a uniformly random  $v \in V$ , run the same BFS percolation exploration, and accept the connected component of  $v$  with an adjusted probability (to account for the fact that a graphlet can be generated from any of its vertices). The acceptance probability is bounded away from 0 and so the algorithm runs in constant expected time. As mentioned earlier, the  $\varepsilon$ -approximate sampling algorithm from [45] is based on the exhaustive enumeration of all subgraphs of size  $\leq k$ , for some  $k$  that depends on  $\varepsilon$ . Our new algorithm entirely bypasses this enumeration barrier.

## 2 Graphlet sampling: algorithms

In this section we present our efficient perfect sampling algorithm for weighted, vertex-labeled graphlets containing a fixed vertex  $r$  from a maximum degree  $\Delta$  graph; in particular, in Section 2.1, we prove a generalized version of Theorem 1 from the introduction. We also provide in Section 2.2 our algorithm for sampling weighted graphlets (i.e., the unrooted, unlabeled case) and establish Theorem 7.

### 2.1 Sampling rooted vertex-labeled graphlets

Let  $G = (V, E)$  be a (possibly infinite) graph of maximum degree  $\Delta$ . For  $U \subseteq V$ , let  $G[U]$  denote the corresponding vertex-induced subgraph of  $G$ ; specifically,  $G[U] = (U, E(U))$ , where  $E(U) \subseteq E$  is the set of edges of  $G$  with both endpoints in  $U$ . A vertex-induced subgraph is a *graphlet* if it is connected. For  $r \in V$ , let  $\mathcal{S}(G, r)$  be the set of all graphlets of  $G$  that contain vertex  $r$ . We call the graphlets in  $\mathcal{S}(G, r)$  the graphlets rooted at  $r$ .

<sup>1</sup> In [45], the threshold is incorrectly stated as  $\lambda < \lambda_*(\Delta + 1)$ ; this is due to a minor error interchanging the infinite  $\Delta$ -regular tree with the infinite  $\Delta$ -ary tree; with this small correction their analysis goes through with the bound on  $\lambda$  as stated here.

Let  $\Sigma = \{1, \dots, q\}$  be a set of vertex labels or colors, and let  $\mathcal{S}(G, r, q) = \bigcup_{S \in \mathcal{S}(G, r)} \Sigma^S$  be the set of all vertex-labeled graphlets rooted at  $r$ . Given a real parameter  $\lambda > 0$ , we assign to each rooted vertex-labeled graphlet  $\gamma \in \mathcal{S}(G, r, q) \cup \{\emptyset\}$  with  $|\gamma|$  vertices the weight  $w_\gamma = \lambda^{|\gamma|} f(\gamma)$ , where  $f : \mathcal{S}(G, r, q) \cup \{\emptyset\} \rightarrow [0, 1]$ . Note that  $0 \leq w_\gamma \leq \lambda^{|\gamma|}$ , which will be important for later analysis.

Define the probability distribution  $\nu_{G, r, \lambda}$  on  $\mathcal{S}(G, r, q) \cup \{\emptyset\}$  by setting

$$\nu_{G, r, \lambda}(\gamma) = \frac{w_\gamma}{Z(G, r, \lambda)}, \quad (11)$$

where  $Z(G, r, \lambda) = \sum_{\gamma' \in \mathcal{S}(G, r, q) \cup \{\emptyset\}} w_{\gamma'}$ . We assume that  $G$ ,  $f$ ,  $q$  and  $\lambda$  are such that  $Z(G, r, \lambda)$  is finite, so that this distribution is well defined. When  $q = 1$  and  $f(\gamma) = \mathbb{1}(\gamma \neq \emptyset)$ ,  $\nu_{G, r, \lambda}$  corresponds exactly to the distribution defined in (1) over the unlabeled graphlets of  $G$  rooted at  $r$ .

We consider the problem of sampling from  $\nu_{G, r, \lambda}$ ; this more general version of the sampling problem is later used as a subroutine for sampling polymer systems in Section 3. Let

$$\lambda_*(\Delta, q) := \frac{(\Delta - 2)^{\Delta-2}}{q(\Delta - 1)^{\Delta-1}};$$

cf., (2). Our main algorithmic result for sampling colored rooted graphlets is the following.

► **Theorem 8.** *Suppose  $\Delta \geq 3$ ,  $\lambda > 0$ , and  $q \geq 1$  are such that  $\lambda < \lambda_*(\Delta, q)$  and let  $a > 0$  be a fixed constant. There is a randomized algorithm to exactly sample from  $\nu_{G, r, \lambda}$  for graphs  $G$  of maximum degree  $\Delta$  and functions  $f : \mathcal{S}(G, r, q) \cup \{\emptyset\} \rightarrow [0, 1]$  where  $f(\gamma)$  is computable in time  $O(|\gamma|^a)$ ; this randomized algorithm has expected running time bounded by  $C \cdot Z_{G, r, \lambda}^{-1}$ , where  $C > 0$  is a constant that depends only on  $q$ ,  $\lambda$ ,  $\Delta$  and  $a$ .*

Theorem 1 from the introduction corresponds to the special case when  $q = 1$  and  $f(\gamma) = \mathbb{1}(\gamma \neq \emptyset)$  (in this case  $Z_{G, r, \lambda} \geq \lambda$ ). Other mild assumptions on the function  $f$ , e.g.,  $f(\emptyset) = 1$  or  $f(r) = 1$ , ensure that  $Z_{G, r, \lambda}$  is bounded away from 0 and, consequently, that the sampling algorithm in the theorem has constant expected running time.

As a warm-up, let us consider first our algorithm for sampling labeled rooted graphlets on a finite graph  $G = (V, E)$  with  $f = 1$ , and purposely omit certain non-essential implementation details for clarity. First, we find  $p \in (0, 1)$  such that  $\frac{p}{q}(1 - p)^{\Delta-2} = \lambda$ ; this choice of  $p$  will be justified in what follows. The algorithm then repeats the following process until a vertex-labeled graphlet is accepted:

1. Each vertex of the graph is independently assigned with probability  $p$  a uniform random color from  $\{1, \dots, q\}$ , or it is marked as “not colored” with the probability  $1 - p$ .
2. Let  $\gamma$  be the vertex-labeled graphlet from  $\mathcal{S}(G, r, q) \cup \{\emptyset\}$  corresponding the colored connected component of  $r$ ; i.e., the set of vertices connected to  $r$  by at least one path of colored vertices.
3. Observe that the probability of obtaining  $\gamma$  is  $(p/q)^{|\gamma|} (1 - p)^{|\partial\gamma|}$ , where  $\partial\gamma$  denotes to set of vertices in  $G$  that are not in  $\gamma$  but are adjacent to a vertex in  $\gamma$  (with a slight abuse of notation, we let  $|\partial\emptyset| = 1$ ). Our aim is to output  $\gamma$  with probability  $\propto \lambda^{|\gamma|}$  which has no dependence on  $\partial\gamma$ . Therefore, we use a “rejection filter” and only accept  $\gamma$  with probability  $(1 - p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|}$ , so that the probability that  $\gamma$  is the output becomes:

$$\left(\frac{p}{q}\right)^{|\gamma|} (1 - p)^{|\partial\gamma|} (1 - p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|} = (1 - p)^2 \left(\frac{p}{q}(1 - p)^{\Delta-2}\right)^{|\gamma|} = (1 - p)^2 \lambda^{|\gamma|}. \quad (12)$$

From (12), the choice of  $p$  such that  $\frac{p}{q}(1 - p)^{\Delta-2} = \lambda$  is apparent. We will prove that only when  $\lambda < \lambda_*(\Delta, q)$  there exists  $p \in (0, 1)$  such that  $\frac{p}{q}(1 - p)^{\Delta-2} = \lambda$ . In the actual implementation of the algorithm, it will in fact suffice to find an approximation for  $p$ .



We comment briefly on the intuition for the rejection filter. The acceptance probability must include a factor of  $(1-p)^{-|\partial\gamma|}$ , so that the final acceptance probability depends on  $|\gamma|$  but not on  $|\partial\gamma|$ . However,  $(1-p)^{-|\partial\gamma|} > 1$  is not a valid probability, so we use instead  $(1-p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|}$ , which is at most 1 since  $(\Delta-2)|\gamma|+2 \geq |\partial\gamma|$ . This bound on  $|\partial\gamma|$  is best possible since it is tight for the  $\Delta$ -regular tree. We note that using looser bounds for  $|\partial\gamma|$  affects the range of the parameter  $\lambda$  for which we can find  $p \in (0, 1)$  so that  $\frac{p}{q}(1-p)^{\Delta-2} = \lambda$ .

Finally, we mention that the algorithm as described requires  $\Omega(|V|)$  time per iteration and cannot be extended to infinite graphs. This is easily corrected by assigning colors starting from  $r$  and revealing only the colored component of  $r$  in a breadth-first fashion. The threshold  $\lambda_*(\Delta, q)$  is sharp in the sense that only when  $\lambda < \lambda_*(\Delta, q)$  is the value of  $p$  such that the revealing process is a sub-critical process that creates a small component with high probability. This ensures the algorithm can be implemented efficiently. In particular, we stress that our algorithm avoids exhaustively enumerating labeled graphlets, as done in previous methods [16].

Before giving the implementation details of our algorithm and proving Theorem 8, we consider the problem of finding  $p \in (0, 1)$  such that  $\frac{p}{q}(1-p)^{\Delta-2} = \lambda$ . For  $\Delta \geq 3$  and  $q \geq 1$ , consider the real function  $g(x) = \frac{x}{q}(1-x)^{\Delta-2}$ . It can be readily checked that the function  $g$  is continuous and differentiable in  $[0, 1]$ , has a unique maximum at  $x = \frac{1}{\Delta-1}$  with  $g(\frac{1}{\Delta-1}) = \lambda_*(\Delta, q)$ , is increasing in  $[0, \frac{1}{\Delta-1}]$ , and decreasing in  $[\frac{1}{\Delta-1}, 1]$ . This implies that only when  $\lambda < \lambda_*(\Delta, q)$ , there exists a value of  $p \in [0, \frac{1}{\Delta-1})$  such that  $g(p) = \lambda$ . In particular, when  $\lambda > \lambda_*(\Delta, q)$ , there is no value of  $p$  for which  $g(p) = \lambda$  and when  $\lambda = \lambda_*(\Delta, q)$ , the only possible value is  $p = \frac{1}{\Delta-1}$ . The latter case would result in a *critical* percolation process, corresponding to the fact that the expected size of a graphlet from  $\nu_{G,r,\lambda}$  has no uniform upper bound in the class of graphs of maximum degree  $\Delta$ ; in fact, it is infinite on the  $\Delta$ -regular tree. We can find a suitable approximation for  $p$  when  $\lambda < \lambda_*(\Delta, q)$  via a simple (binary search) procedure.

► **Lemma 9.** *For any  $\lambda \in [0, \lambda_*(\Delta, q))$  we can find rational numbers  $\hat{p} \in [0, \frac{1}{\Delta-1})$  and  $\hat{\lambda} \in [\lambda, \lambda_*(\Delta, q)]$  such that  $g(\hat{p}) = \hat{\lambda}$  in  $O(|\log \frac{1}{\Delta q(\lambda_* - \lambda)}|)$  time.*

The proof of this lemma appears after the proof of Theorem 8. We now prove Theorem 8, including giving a more detailed version of the algorithm outlined above that includes the previously omitted implementation details and allows for general functions  $f : \mathcal{S}(G, r, q) \cup \{\emptyset\} \rightarrow [0, 1]$ .

**Proof of Theorem 8.** For ease of notation, let  $\lambda_* = \lambda_*(\Delta, q)$ . Our algorithm to sample from  $\nu_{G,r,\lambda}$  when  $\lambda < \lambda_*$  explores from  $r$  in a breadth-first manner and stops once it has revealed the colored connected component of  $r$ . It proceeds as follows:

1. Find  $\hat{p} \in [0, \frac{1}{\Delta-1})$  and  $\hat{\lambda} \in [\lambda, \lambda_*)$  such that  $g(\hat{p}) = \hat{\lambda}$ . This can be done in time  $O(|\log \frac{1}{\Delta q(\lambda_* - \lambda)}|)$  per Lemma 9.
2. Let  $Q$  be a queue. With probability  $1 - \hat{p}$  do not add  $r$  to  $Q$ ; otherwise, assign  $r$  a color uniformly at random from  $\{1, \dots, q\}$  and add  $r$  to  $Q$ . Mark  $r$  as explored.
3. While  $Q \neq \emptyset$ , repeat the following:
  - (3.1) Pop a vertex  $v$  from  $Q$ .
  - (3.2) For each unexplored neighbor  $w$  of  $v$ , with probability  $1 - \hat{p}$  do not add  $w$  to  $Q$ ; otherwise, assign  $w$  a color uniformly at random from  $\{1, \dots, q\}$  and add  $w$  to  $Q$ . Mark  $w$  as explored (regardless of whether it was added to  $Q$  or not).



4. Let  $\gamma$  be the vertex-labeled graphlet from  $\mathcal{S}(G, r, q) \cup \{\emptyset\}$  corresponding the colored connected component of  $r$ . Accept  $\gamma$  with probability:

$$f(\gamma) \cdot (1 - \hat{p})^{(\Delta-2)|\gamma|+2-|\partial\gamma|} \left(\frac{\lambda}{\hat{\lambda}}\right)^{|\gamma|}.$$

5. If  $\gamma$  is rejected, go to Step 2 and repeat.

The probability of obtaining  $\gamma \in \mathcal{S}(G, r, q) \cup \{\emptyset\}$  in an iteration of the algorithm is:

$$\left(\frac{\hat{p}}{q}\right)^{|\gamma|} (1 - \hat{p})^{|\partial\gamma|} \cdot f(\gamma) (1 - \hat{p})^{(\Delta-2)|\gamma|+2-|\partial\gamma|} \left(\frac{\lambda}{\hat{\lambda}}\right)^{|\gamma|} = (1 - \hat{p})^2 f(\gamma) \lambda^{|\gamma|} = (1 - \hat{p})^2 w_\gamma,$$

and thus the overall acceptance probability in an iteration is:

$$\rho := (1 - \hat{p})^2 \sum_{\gamma \in \mathcal{S}(G, r, q) \cup \{\emptyset\}} w_\gamma = (1 - \hat{p})^2 Z_{G, r, \lambda}.$$

Then,

$$\Pr[\gamma \in \mathcal{S}(G, r, q) \cup \{\emptyset\} \text{ is the output}] = \sum_{t \geq 1} (1 - \hat{p})^2 w_\gamma (1 - \rho)^{t-1} = \frac{(1 - \hat{p})^2 w_\gamma}{\rho} = \nu_{G, r, \lambda}(\gamma).$$

We next bound the expected running time of the algorithm. We claim first that expected running per iteration is at most a constant that depends only on  $a$ ,  $\Delta$  and  $q$ . If  $\gamma$  is the configuration generated in an iteration, it is discovered in  $O(|\gamma| + |\partial\gamma|) = O(|\gamma|)$  time and, by assumption,  $f(\gamma)$  can be computed in at most  $O(|\gamma|^a)$  time, for suitable a constant  $a > 0$ . Let  $\hat{\mu}$  the output distribution of Step 3 of the algorithm. Then, there exists a constant  $C = C(q, \Delta) > 0$  such that the expected running time of each iteration is at most:

$$C \sum_{\gamma \in \mathcal{S}(G, r, q) \cup \emptyset} |\gamma|^{\max\{a, 1\}} \Pr_{\hat{\mu}}[\gamma] = C \cdot \mathbb{E}_{\hat{\mu}}[|\gamma|^{\max\{1, a\}}]. \quad (13)$$

We show next that  $|\gamma|$  (under  $\hat{\mu}$ ) is stochastically dominated by a random variable  $W = X + Y$  (i.e.,  $|\gamma| \prec W$ ), where  $X$  and  $Y$  are i.i.d. random variables corresponding to the cluster size of a homogeneous Galton-Watson tree with offspring distribution  $\text{Bin}(\Delta - 1, \hat{p})$ . To see this, first note that  $|\gamma| \prec L$ , where  $L$  is the cluster size of a heterogeneous Galton-Watson tree, in which the root vertex has offspring distribution  $\text{Bin}(\Delta, \hat{p})$  and every other vertex has offspring distribution  $\text{Bin}(\Delta - 1, \hat{p})$ . This is because the branching process generating  $\gamma$  includes the root only with probability  $\hat{p}$  (the root is always present in the Galton-Watson tree), and, in addition, it considers at most  $\Delta$  (from the root) or  $\Delta - 1$  (from any other vertex) potential branches. In turn, we have that  $L \prec X + Y$ , since we can couple the first  $\Delta - 1$  branches of the root with  $X$  (starting at the root) and the remaining branch with  $Y$  (starting at the child of the root not coupled with  $X$ ).

It is well-known that  $X$  and  $Y$  have finite moments when  $(\Delta - 1)\hat{p} < 1$  (see, e.g., [34]). In particular, there exists a constant  $A = A(a, \Delta, \hat{p}) > 0$  such that

$$\mathbb{E}_{\hat{\mu}}[|\gamma|^a] \leq \mathbb{E}[L^a] \leq \mathbb{E}[(X + Y)^a] \leq 2^a (\mathbb{E}[X^a] + \mathbb{E}[Y^a]) \leq A. \quad (14)$$

This together with (13) shows that the expected running time in each iteration of the algorithm is bounded by  $C \cdot A$ .

Now, let  $R$  be the number of times Steps 2–5 are repeated, let  $T$  be the overall the running time of the algorithm. Then:

$$\mathbb{E}[T] = \sum_{t \geq 1} \mathbb{E}[T \mid R = t] \Pr[R = t] \leq C \cdot A \cdot \sum_{t \geq 1} t(1 - \rho)^{t-1} \rho \leq \frac{CA}{\rho}, \quad (15)$$

and the result follows.  $\blacktriangleleft$

We conclude this section with the proof of Lemma 9.

**Proof of Lemma 9.** It suffices to find  $\hat{p} \in [p, \frac{1}{\Delta-1}]$ . This can be done via binary search in  $t$  steps, provided  $t \geq 0$  is such that  $\frac{1}{\Delta-1} \cdot \frac{1}{2^t} \leq \frac{1}{\Delta-1} - p$ . Since  $g' \leq \frac{1}{q}$ , it follows from the mean value theorem that  $q(\lambda_* - \lambda) \leq \frac{1}{\Delta-1} - p$ . Thus for the binary search to require at most  $t$  steps it is sufficient to pick  $t$  so that  $\frac{1}{\Delta-1} \cdot \frac{1}{2^t} \leq q(\lambda_* - \lambda)$ , and the result follows. ◀

## 2.2 Sampling unrooted graphlets

We consider next the problem of sampling weighted graphlets from a finite graph  $G = (V, E)$  of maximum degree  $\Delta$ ; specifically, in this variant of the sampling problem we consider unrooted, unlabeled, weighted graphlets of  $G$ . Let  $\mathcal{S}(G)$  be the set of all graphlets of  $G$ . We define the probability distribution  $\nu_{G,\lambda}$  on  $\mathcal{S}(G)$  by setting

$$\nu_{G,\lambda}(S) = \frac{\lambda^{|S|}}{Z_{G,\lambda}},$$

where  $Z_{G,\lambda} = \sum_{S' \in \mathcal{S}(G)} \lambda^{|S'|}$ . The problem of (approximately) sampling from  $\nu_{G,\lambda}$  is quite natural. In [45], it was established that this problem is computationally hard when  $\lambda > \lambda_*(\Delta) = \frac{(\Delta-2)^{\Delta-2}}{(\Delta-1)^{\Delta-1}}$ ; an  $\varepsilon$ -approximate sampling algorithm was also given in [45] for the case when  $\lambda < \lambda_*(\Delta)$  with running time  $n \cdot (1/\varepsilon)^{O(\log \Delta)}$ . We now provide the proof of Theorem 7, which says we can perfectly sample from  $\nu_{G,\lambda}$  in constant expected time when  $\lambda < \lambda_*(\Delta)$ .

**Proof of Theorem 7.** For ease of notation, we set  $\lambda_* = \lambda_*(\Delta)$  throughout this proof. Our algorithm to sample from  $\nu_{G,\lambda}$  is based on the algorithm to sample from  $\nu_{G,r,\lambda}$  (the rooted, vertex-labeled, weighted case). The idea is to pick a root uniformly at random and run the algorithm for the rooted case from this random vertex with the rejection filter adjusted to account for the fact that a graphlet can be generated from any of its vertices. It proceeds as follows:

1. Find  $\hat{p} \in [0, \frac{1}{\Delta-1})$  and  $\hat{\lambda} \in [\lambda, \lambda_*)$  such that  $g(\hat{p}) = \hat{\lambda}$  using the method from Lemma 9.
2. Pick a vertex  $r \in V$  uniformly at random.
3. Let  $Q$  be a queue. With probability  $\hat{p}$  add  $r$  to  $Q$  and mark it as colored. Mark  $r$  as explored.
4. While  $Q \neq \emptyset$ , repeat the following:
  - (4.1) Pop a vertex  $v$  from  $Q$ .
  - (4.2) For each unexplored neighbor  $w$  of  $v$ , with probability  $\hat{p}$  add  $w$  to  $Q$  and mark  $w$  as colored. Mark  $w$  as explored.
5. Let  $S \in \mathcal{S}(G)$  be the graphlet corresponding to the colored connected component of  $v$ . Accept  $S$  with probability:

$$\frac{1}{|S|} \cdot (1 - \hat{p})^{(\Delta-2)|S|+2-|\partial S|} \left( \frac{\lambda}{\hat{\lambda}} \right)^{|S|}.$$

6. If  $S$  is rejected, go back to Step 2 and repeat.

The analysis of this algorithm is similar to that in the proof of Theorem 8. Let  $n = |V|$ . The probability that the algorithm outputs  $S$  in an iteration is:

$$\sum_{v \in S} \frac{1}{n} \cdot \hat{p}^{|S|} (1 - \hat{p})^{|\partial S|} \cdot \frac{1}{|S|} \cdot (1 - \hat{p})^{(\Delta-2)|S|+2-|\partial S|} \left( \frac{\lambda}{\hat{\lambda}} \right)^{|S|} = \frac{(1 - \hat{p})^2 \lambda^{|S|}}{n}. \quad (16)$$

Hence, conditioned on acceptance, the probability of obtaining  $S \in \mathcal{S}(G)$  is thus  $\nu_{G,\lambda}(S)$ , and so the output distribution of the algorithm is  $\nu_{G,\lambda}$ .

For the running time of the algorithm, we note that Step 4 of the algorithm is analogous to Step 3 of the algorithm in the proof of Theorem 8, and so the expected running time of each round is also bounded by a constant  $C = C(\Delta, \hat{p}) > 0$ . Let  $T$  be the overall the running time of the algorithm. From (16), we have that the overall acceptance probability in a round is  $\rho = \frac{(1-\hat{p})^2 Z(G, \lambda)}{n}$ . Then, as in (15), we deduce that  $\mathbb{E}[T] = O(nZ(G, \lambda)^{-1})$ . Since  $Z(G, \lambda) \geq n\lambda$ , we have  $\mathbb{E}[T] = O(1)$ . ◀

### 3 Applications to Polymer Models

In this section, we show how to use our algorithm for sampling rooted vertex-labeled graphlets from Section 2 to sample from subset polymer models and prove Theorem 4.

Consider a subset polymer model on an  $n$ -vertex graph  $G = (V, E)$ ; see Section 1.2 for the definition. Recall that we use  $\mathcal{C}_v$  for the set of all polymers containing vertex  $v \in V$ , and let  $\gamma_\emptyset$  denote the empty polymer. Define the distribution  $\nu_v$  on  $\mathcal{C}_v \cup \{\gamma_\emptyset\}$  by

$$\nu_v(\gamma) = \frac{w_\gamma}{\sum_{\hat{\gamma} \in \mathcal{C}_v \cup \{\gamma_\emptyset\}} w_{\hat{\gamma}}},$$

where we assign  $w_{\gamma_\emptyset} = 1$ . The following Markov chain on  $\Omega(\mathcal{C})$ , introduced in [16], has stationary distribution  $\mu$  and mixes rapidly in  $O(n \log n)$  steps under the polymer mixing condition (6).

**Polymer dynamics.** Given a configuration  $X_t \in \Omega(\mathcal{C})$ , form  $X_{t+1}$  as follows:

1. Pick  $v \in V$  uniformly at random and let  $S_v = \{\gamma \in X_t : v \in \gamma\}$  (note that  $S_v$  is either empty or contains 1 polymer).
2. With probability  $1/2$ , let  $X_{t+1} = X_t \setminus S_v$ .
3. With probability  $1/2$  (exclusively of step 2), sample  $\gamma$  from  $\nu_v$ . Let  $X_{t+1} = X_t \cup \{\gamma\}$  if  $X_t \cup \{\gamma\} \in \Omega(\mathcal{C})$  and let  $X_{t+1} = X_t$  otherwise.

To implement a single update step, one must sample from  $\nu_v$  in Step 3. To do so efficiently, in [16] the much stricter polymer sampling condition (7) was required. Here we give a fast perfect sampler for  $\nu_v$  under a much weaker condition.

► **Theorem 10.** *Consider a subset polymer model on a family of  $n$ -vertex graphs of maximum degree  $\Delta \geq 3$  with computationally feasible weights that satisfy  $w_\gamma \leq \lambda^{|\gamma|}$  for some  $\lambda < \lambda_*(\Delta, q)$ . There is a randomized algorithm to sample perfectly from  $\nu_v$  for any  $v \in V$  with expected running time bounded by a function of  $\lambda, \Delta$ , and  $q$ .*

**Proof.** This follows from Theorem 8. ◀

Using this theorem and the fast mixing result for the polymer dynamics of [16], one can approximately sample from  $\mu$  whenever both the polymer mixing condition (6) and the assumptions of Theorem 10 hold. We further improve this by giving a perfect sampling algorithm that works whenever a new condition (4) is satisfied (our algorithm also requires the assumptions in Theorem 10). In all known examples, condition (4) is more permissive than the polymer mixing condition (6) from [16].

#### 3.1 Perfect Sampling for polymer systems: Proof of Theorem 4

As mentioned, the polymer dynamics from [16] is not a perfect sampler. We propose here a different algorithm to output a perfect sample from  $\mu$ . Our algorithm uses the coupling from the past method [44] and the notion of bounding Markov chains [33, 30] to efficiently implement it.

We proceed with the proof of Theorem 4. We start with the description of a grand coupling for the polymer dynamics, which is then used to implement a coupling from the past algorithm. For an  $n$ -vertex graph  $G = (V, E)$ , let  $\{X_t^\Gamma\}$  denote an instance of the polymer dynamics started from the polymer configuration  $\Gamma \in \Omega(\mathcal{C})$ . For all  $\Gamma \in \Omega(\mathcal{C})$ , the chains  $\{X_t^\Gamma\}$  are coupled by choosing the same uniform random vertex  $v \in V$ , the same polymer  $\gamma$  sampled from  $\nu_v$ , and the same uniform random number in  $[0, 1]$  to decide whether to remove  $S_v$  (Step 2) or to add  $\gamma$  (Step 3). A coupling from the past algorithm will find a time  $-T$  such the grand coupling started from all possible states at time  $-T$  coalesces to a single state by time 0. This guarantees that the output of the algorithm, that is the state at time 0, has distribution  $\mu$  (see Theorem 1 from [44]). Such a  $T$  can be found with a binary search procedure. Unfortunately, implementing the coupling from the past algorithm in this manner for the polymer dynamics Markov chain is infeasible in our setting, since it requires simulating an exponential number of copies of the polymer dynamics, one from each  $\Gamma \in \Omega(\mathcal{C})$ .

To work around this, we consider a bounding Markov chain for the polymer dynamics rather than the polymer dynamics chain itself. Bounding Markov chains were pioneered in [33, 30] as a method for efficiently implementing coupling from the past. The bounding chain for the polymer dynamics has state space  $\Omega(\mathcal{C}) \times 2^{\mathcal{C}}$  and will be denoted by  $\{B_t, D_t\}$ , where  $B_t \in \Omega(\mathcal{C})$  and  $D_t \subseteq \mathcal{C}$  are sets of polymers. The chain will maintain throughout that all polymers in  $B_t$  are compatible and that every polymer in  $B_t$  is compatible with every polymer in  $D_t$ . The polymers in  $D_t$  do *not* need to be compatible with each other. A step of the bounding Markov chain is defined next.

**Polymer Dynamics Bounding Chain.** Given  $\{B_t, D_t\}$ , the chain generates  $\{B_{t+1}, D_{t+1}\}$  by:

1. Uniformly at random, select  $v \in V$ .
2. With probability  $1/2$ , remove all polymers containing  $v$  by setting  $B_{t+1} = B_t \setminus \mathcal{C}_v$  and  $D_{t+1} = D_t \setminus \mathcal{C}_v$ .
3. With the remaining probability  $1/2$ , draw a sample  $\gamma$  according to  $\nu_v$  and:
  - a. If  $\gamma$  is compatible with  $B_t$  and  $\gamma$  is compatible with  $D_t \setminus \{\gamma\}$ , let  $B_{t+1} = B_t \cup \{\gamma\}$  and let  $D_{t+1} = D_t \setminus \{\gamma\}$ .
  - b. Else if  $\gamma$  is compatible with  $B_t$  but  $\gamma$  is not compatible with  $D_t$ , let  $B_{t+1} = B_t$  and let  $D_{t+1} = D_t \cup \{\gamma\}$ .
  - c. If  $\gamma$  is incompatible with  $B_t$ , do nothing:  $B_{t+1} = B_t$  and  $D_{t+1} = D_t$ .

Observe that polymers are only added to  $B_t$  if they are compatible with all other polymers in  $B_t$ ; hence, if  $B_0$  is a valid polymer configuration, so is  $B_t$  for all  $t \geq 0$ .

To implement a step of the polymer dynamics bounding chain it suffices to pick a vertex  $v \in V$  uniformly at random, a uniform random number in  $[0, 1]$ , and a polymer  $\gamma$  from  $\nu_v$ , just like for the polymer dynamics. Hence, we can couple the evolution of  $\{B_t, D_t\}$  with the grand coupling of the polymer dynamics described earlier. If we set  $B_0 = \emptyset$  and  $D_0 = \mathcal{C}$ , it can be checked that for all  $\Gamma \in \Omega(\mathcal{C})$  and all  $t \geq 0$ :

$$B_t \subseteq X_t^\Gamma \subseteq B_t \cup D_t.$$

Indeed, this holds initially for  $t = 0$ , and the grand coupling ensures that whenever a polymer is removed from  $X_t^\Gamma$  it is also removed from  $B_t$ , and whenever a polymer is added to  $X_t^\Gamma$  it is also added to  $B_t$  or  $D_t$ . Consequently,  $\{B_t, D_t\}$  is a bounding chain for the polymer dynamics. In particular, the first time  $B_t = B_t \cup D_t$ , all instances  $X_t^\Gamma$  have necessarily coalesced to the same configuration. This bounding chain allows us to implement the coupling from the past algorithm efficiently, as follows.

**Coupling from the Past.** Set  $k = 1$ .

- (A) For  $t = -2^k, -2^k + 1, \dots, -2^{k-1}$  generate  $\rho_t = (v_t, \gamma_t, r_t)$  by choosing  $v_t \in V$  uniformly at random,  $r_t \in [0, 1]$  uniformly at random, and by sampling  $\gamma_t \in \mathcal{C}_{v_t}$  from  $\nu_{v_t}$ .
- (B) Set  $B_{-2^k} = \emptyset$  and  $D_{-2^k} = \mathcal{C}$ .
- (C) Simulate the polymer dynamics bounding chain from time  $-2^k$  to time 0 using  $\rho_{-2^k}, \dots, \rho_{-1}$ .
- (D) If  $B_0 = B_0 \cup D_0$ , then output  $B_0$ ; otherwise set  $k \rightarrow k + 1$  and repeat the process from Step (A).

This implementation of the coupling from the past algorithm provides a perfect sample from  $\mu$ ; see [44]. It remains for us to show that it can be efficiently implemented. For this, we show first that the expected number of steps of the polymer dynamics bounding chain throughout the execution of the algorithm is  $O(n \log n)$ . Afterwards, we will show how to implement steps so that they can be executed in amortized constant expected time.

► **Lemma 11.** *Suppose the subset polymer model satisfies condition (4). Then, the expected number of steps of the polymer dynamics bounding chain in the coupling from past algorithm is  $O(n \log n)$ .*

**Proof.** See the full version [6], where a potential  $\phi_t$  describing the size of  $D_t$  is introduced and shown to decrease quickly. ◀

It remains to consider how to efficiently implement the steps of the polymer dynamics bounding chain. This is subtle because  $D_t$  may initially contain an exponentially large number of polymers, and care is thus needed in how  $D_t$  is represented and stored. The following lemma says we can represent and update  $B_t$  and  $D_t$  efficiently.

► **Lemma 12.** *There exists a compact representation of  $B_t$  and  $D_t$  that uses  $O(n + t)$  space in expectation. Using this representation, each iteration of the Polymer Dynamics Bounding Chain can be executed in amortized constant expected time.*

**Proof.** See the full version [6]. ◀

---

## References

- 1 Matteo Agostini, Marco Bressan, and Shahrzad Haddadan. Mixing time bounds for graphlet random walks. *Information Processing Letters*, 152:105851, 2019.
- 2 Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *arXiv preprint*, 2021. [arXiv:2106.15992](#).
- 3 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *SIAM Journal on Computing: Special Section, FOCS 2020*, pages FOCS20:1–FOCS20:37, 2022.
- 4 Kim Baskerville, Peter Grassberger, and Maya Paczuski. Graph animals, subgraph sampling, and motif search in large networks. *Physical Review E*, 76(3):036107, 2007.
- 5 Mansurul A Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *2012 IEEE 12th International Conference on Data Mining*, pages 91–100. IEEE, 2012.
- 6 Antonio Blanca, Sarah Cannon, and Will Perkins. Fast and perfect sampling of subgraphs and polymer systems. *arXiv preprint*, 2022. [arXiv:2202.05907](#).
- 7 Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on  $\mathbb{Z}^d$  at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 738–751, 2020.

- 8 Christian Borgs, Jennifer Chayes, Jeff Kahn, and László Lovász. Left and right convergence of graphs with bounded degree. *Random Structures & Algorithms*, 42(1):1–28, 2013.
- 9 Marco Bressan. Efficient and near-optimal algorithms for sampling connected subgraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1132–1143, 2021.
- 10 Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of the tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 557–566, 2017.
- 11 Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):1–25, 2018.
- 12 Marco Bressan, Stefano Leucci, and Alessandro Panconesi. Faster motif counting via succinct color coding and adaptive sampling. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(6):1–27, 2021.
- 13 Sarah Cannon and Will Perkins. Counting independent sets in unbalanced bipartite graphs. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1456–1466, 2020.
- 14 Charlie Carlson, Ewan Davies, and Alexandra Kolla. Efficient algorithms for the Potts model on small-set expanders. *arXiv preprint*, 2020. [arXiv:2003.01154](#).
- 15 Xiaowei Chen, Yongkun Li, Pinghui Wang, and John CS Lui. A general framework for estimating graphlet statistics via random walk. *Proceedings of the VLDB Endowment*, 10(3), 2016.
- 16 Zongchen Chen, Andreas Galanis, Leslie A Goldberg, Will Perkins, James Stewart, and Eric Vigoda. Fast algorithms at low temperatures via Markov chains. *Random Structures & Algorithms*, 58(2):294–321, 2021.
- 17 Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Sampling colorings and independent sets of random regular bipartite graphs in the non-uniqueness region. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2198–2207. SIAM, 2022.
- 18 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1307–1318. IEEE, 2020.
- 19 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1537–1550, 2021.
- 20 Matthew Coulson, Ewan Davies, Alexandra Kolla, Viresh Patel, and Guus Regts. Statistical physics approaches to unique games. In *35th Computational Complexity Conference (CCC)*, 2020.
- 21 Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.
- 22 Sacha Friedli and Yvan Velenik. *Statistical mechanics of lattice systems: a concrete mathematical introduction*. Cambridge University Press, 2017.
- 23 Tobias Friedrich, Andreas Göbel, Martin S Krejca, and Marcus Pappik. Polymer dynamics via cliques: New conditions for approximations. *arXiv preprint*, 2020. [arXiv:2007.08293](#).
- 24 Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast algorithms for general spin systems on bipartite expanders. *ACM Transactions on Computation Theory (TOCT)*, 13(4):1–18, 2021.
- 25 Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast mixing via polymers for random graphs with unbounded degree. *Information and Computation*, 285:104894, 2022.
- 26 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.



- 27 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*, pages 406–415, 1997.
- 28 Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.
- 29 Christian Gruber and Hervé Kunz. General properties of polymer systems. *Communications in Mathematical Physics*, 22(2):133–161, 1971.
- 30 Olle Haggstrom and Karin Nelander. On exact simulation of Markov random fields using coupling from the past. *Scandinavian Journal of Statistics*, 26(3):395–411, 1999.
- 31 Tyler Helmuth, Matthew Jenssen, and Will Perkins. Finite-size scaling, phase coexistence, and algorithms for the random cluster model on random graphs. *Annales de l’Institut Henri Poincaré B. To appear*, 2022.
- 32 Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov–Sinai theory. *Probability Theory and Related Fields*, 176(3):851–895, 2020.
- 33 Mark Huber. Perfect sampling using bounding chains. *The Annals of Applied Probability*, 14(2):734–753, 2004.
- 34 Svante Janson, Andrzej Rucinski, and Tomasz Luczak. *Random graphs*. John Wiley & Sons, 2011.
- 35 Matthew Jenssen, Peter Keevash, and Will Perkins. Algorithms for #BIS-hard problems on expander graphs. *SIAM Journal on Computing*, 49(4):681–710, 2020.
- 36 Matthew Jenssen, Aditya Potukuchi, and Will Perkins. Approximately counting independent sets in bipartite graphs via graph containers. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 499–516. SIAM, 2022.
- 37 Madhav Jha, C Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 495–505, 2015.
- 38 Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.
- 39 Roman Kotecký and David Preiss. Cluster expansion for abstract polymer models. *Communications in Mathematical Physics*, 103(3):491–498, 1986.
- 40 Chao Liao, Jiabao Lin, Pinyan Lu, and Zhenyu Mao. Counting independent sets and colorings on random regular bipartite graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- 41 Xuesong Lu and Stéphane Bressan. Sampling connected induced subgraphs uniformly at random. In *International Conference on Scientific and Statistical Database Management*, pages 195–212. Springer, 2012.
- 42 Ryuta Matsuno and Aristides Gionis. Improved mixing time for  $k$ -subgraph sampling. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 568–576. SIAM, 2020.
- 43 Kirill Paramonov, Dmitry Shemetov, and James Sharpnack. Estimating graphlet statistics via lifting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 587–595, 2019.
- 44 James Gary Propp and David Bruce Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.
- 45 Andrew Read-McFarland and Daniel Štefankovič. The hardness of sampling connected subgraphs. In *Latin American Symposium on Theoretical Informatics*, pages 464–475. Springer, 2021.

- 46    Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.
- 47    Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 287–296. IEEE, 2010.
- 48    Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on  $d$ -regular graphs. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 361–369. IEEE, 2012.
- 49    Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149, 2006.



# High Dimensional Expansion Implies Amplified Local Testability

Tali Kaufman ✉

Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Izhar Oppenheim ✉

Department of Mathematics, Ben-Gurion University of the Negev, Be'er-Sheva, Israel

---

## Abstract

In this work, we define a notion of local testability of codes that is strictly stronger than the basic one (studied e.g., by recent works on high rate LTCs), and we term it *amplified local testability*. Amplified local testability is a notion close to the result of optimal testing for Reed-Muller codes achieved by Bhattacharyya et al.

We present a scheme to get amplified locally testable codes from high dimensional expanders. We show that single orbit Affine invariant codes, and in particular Reed-Muller codes, can be described via our scheme, and hence are amplified locally testable. This gives the strongest currently known testability result of single orbit affine invariant codes, strengthening the celebrated result of Kaufman and Sudan.

**2012 ACM Subject Classification** Theory of computation → Expander graphs and randomness extractors; Theory of computation → Error-correcting codes

**Keywords and phrases** Locally testable codes, High dimensional expanders, Amplified testing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.5

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.10488>

**Funding** *Tali Kaufman*: This work was partially funded by ERC grant no. 336283 and BSF grant no. 2012256.

*Izhar Oppenheim*: This work was partially funded by ISF grant no. 293/18.

## 1 Introduction

### High dimensional expansion implies amplified local testability

The aim of this work is to show that codes arising from high dimensional expanding set systems have a strong notion of local testability, which we call amplified locally testability (see exact definition below). Specifically, we define the notion of *High Dimensional Expanding System* (HDE-System) that is a two layer expanding set system that generalizes two layer set systems arising from high dimensional expanders. Using this new concept, we show that codes whose constraints form an HDE-System are amplified locally testable.

### Testability of well studied codes via high dimensional expansion

We further show that most well studied locally testable codes such as Reed-Muller codes and more generally affine-invariant codes are, in fact, HDE-System codes. Hence, their (amplified) local testability could be re-inferred using our current work; and could be attributed to the high dimensional expansion phenomenon. Specifically, we give a high dimension expansion based proof to the amplified local testability of single orbit affine invariant codes, that strengthen the well known result of Kaufman and Sudan [10].



© Tali Kaufman and Izhar Oppenheim;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 5; pp. 5:1–5:10



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

### Amplified local testability

In our work, we study a strong notion of local testability for a family of locally testable codes and show that this strong testing property is holds for HDE-System codes.

In order to better explain the notion of amplified local testability, we recall the following formulation of locally testable codes:

► **Definition 1** (Locally testable code). *Given a linear code  $C \subseteq \mathbb{F}_p^V$  defined by a set  $\mathcal{E}_C$  of  $k$ -query tests (i.e., tests that query  $k$ -bits of the codeword), define  $\text{rej} : \mathbb{F}_p^V \rightarrow [0, 1]$  where  $\text{rej}(\underline{c})$  is the fraction of  $k$ -query tests that  $\underline{c}$  fails (by definition,  $\underline{c} \in C$  if and only if  $\text{rej}(\underline{c}) = 0$ ). We refer to querying an equation in  $\mathcal{E}_C$  as the basic test of the code.*

*Let  $\mathcal{C}$  be a sequence of codes such that there is  $k = k(\mathcal{C})$  such that every  $C \in \mathcal{C}$  is defined by a set  $\mathcal{E}_C$  of  $k$ -query tests. We say that a family of linear codes  $\mathcal{C}$  is locally testable if there is a constant  $r_C > 0$  such that for every  $C \in \mathcal{C}$  the following robustness property holds: For every  $\underline{c} \in \mathbb{F}_p^V$ ,*

$$\text{rej}(\underline{c}) \geq r_C \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|,$$

*where  $\|\underline{c} - \underline{c}'\|$  is the fraction of the bits in which  $\underline{c}$  and  $\underline{c}'$  differ.*

Note that in the above definition the number of bits queried in the basic test for a code  $C \in \mathcal{C}$  is constant (independent of  $C$ ) and thus one does not care if  $r_C$  depends on  $k$ . For instance in the recent celebrated work of Dinur at el. [4], the basic test queries  $k$  bits, and  $r_C$  is of the order of  $\frac{1}{\sqrt{k}}$  (see [4, Theorem 4.5]).

However, a sequence of locally testable codes is usually a part of a larger family of codes where  $k$  does vary. The motivating example is (binary) Reed-Muller codes  $\text{RM}(d, n)$ , where  $d$  is the degree of the polynomial and  $n$  is the number of variables. When fixing  $d$  and considering the sequence of codes where  $n$  tends to infinity, it is a classical result that this sequence is a locally testable code with the number of bits queried in the basic test is  $k = 2^d$  (see [1]). If we consider the family of codes  $\text{RM}(d, n)$  where  $d$  also varies (and  $n$  is large enough with respect to  $d$ ), we get a larger family of codes in which  $k(C)$  is no longer constant. For this example of binary Reed-Muller codes, Bhattacharyya at el. [2] proved a striking result of optimal testing for binary Reed-Muller codes:

► **Theorem 2** ([2, Theorem 1]). *Let*

$$\mathcal{C} = \{\text{RM}(d, n) : d \in \mathbb{N}, d \geq 2, n \in \mathbb{N} \text{ sufficiently large with respect to } d\}$$

*be the family of all binary Reed-Muller codes. There is a constant  $r_{\text{RM}} > 0$  such that for every  $C = \text{RM}(d, n) \in \mathcal{C}$  with  $k(C) = 2^d$ , it holds for every  $\underline{c} \in \mathbb{F}_2^n$ , that*

$$\text{rej}(\underline{c}) \geq k(C) r_{\text{RM}} \min\left\{\min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|, \frac{1}{k(C)}\right\}.$$

In other words, [2] show that the family of all binary Reed-Muller codes is locally testable even when changing the degree! This Theorem can be interpreted as follows: As noted above, for every  $\text{RM}(d, n) \in \mathcal{C}$ ,  $k(\text{RM}(d, n)) = 2^d$ . Thus the above theorem states that for every  $\underline{c} \in \mathbb{F}_2^n$ , if  $\min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\| \leq \frac{1}{2^d}$ , then

$$\text{rej}(\underline{c}) \geq 2^d r_{\text{RM}} \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|$$

and if  $\min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\| \geq \frac{1}{2^d}$ , then

$$\text{rej}(\underline{c}) \geq r_{\text{RM}} \geq r_{\text{RM}} \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|.$$

Motivated by the above result, we define amplified local testability as a relaxation of optimal testability:

► **Definition 3** (Amplified locally testable codes). *Let  $\mathcal{C}$  be a family of codes such that every  $C \in \mathcal{C}$  is defined by a set  $\mathcal{E}_C$  of  $k(C)$ -query (basic) tests. We say that a family of linear codes  $\mathcal{C}$  is amplified locally testable if there are constants  $t_C \geq 1$  and  $r_C > 0$  such that for every  $C \in \mathcal{C}$  the following robustness property holds: For every  $\underline{c} \in \mathbb{F}_p^V$ ,*

$$\text{rej}(\underline{c}) \geq k(C)r_C \min \left\{ \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|, \frac{1}{(k(C))^{t_C}} \right\}.$$

► **Remark 4** (Role of  $t_C$ ). The best we can hope for amplified local testing is  $t_C = 1$ . If this happens, then the family has optimal local testability as in the result of [2]. Our methods below do not give optimal local testability, but only amplified local testability with  $t_C = 3$ .

► **Remark 5.** A similar relaxation of optimal testability was studied for lifted codes by Haramaty, Ron-Zewi and Sudan in [6].

In this work we show that a code which can be described via HDE-System, not only we can infer local testability for it, but rather we can infer amplified local testability for it. As already noted above, this is not the case of the the analysis of the family of codes of [4]: In [4] the basic test samples  $k$ -bits and  $r_C$  (in the notation of Definition 1 above) behaves like  $\frac{1}{\sqrt{k}}$  and thus decreases as  $k$  increases.

By applying our machinery to single orbit affine invariant codes, we get that these codes are amplified locally testable, which is the strongest notion of testability known for these codes, strengthening the well known work of Kaufman and Sudan [10].

### Local testability of single orbit affine invariant codes via HDE-System

In the following we refer to single orbit affine invariant codes which were shown to be locally testable by the Kaufman-Sudan work [10]. These codes contain the well known Reed-Muller codes. We show that they are HDE-System codes, so their local testability is implied by our current work. Kaufman and Sudan have shown that single orbit affine invariant codes which are characterized by  $k$ -weight constraints that form a single orbit are locally testable. We will show that the Kaufman-Sudan requirement allows to show that single orbit affine invariant codes are modelled over HDE-System and thus are amplified locally testable.

► **Theorem 6** (Testability of single orbit affine invariant codes – informal, for formal, see the related full version of this paper). *Let  $\mathcal{C}_{\text{affine-inv},p}$  be the family of all single orbit affine invariant codes  $C \subseteq \mathbb{F}_p^{\mathbb{K}(C)^{n(C)}}$  with*

$$|\mathbb{K}(C)|^{n(C)} \geq 2^{11} p^2 (k(C))^4,$$

*where  $k(C)$  is the size of the support of the constraint defining  $C$ . Then the family of all these codes is amplified locally testable. Explicitly, for every  $C \in \mathcal{C}_{\text{affine-inv},p}$  and every  $\underline{c} \in \mathbb{F}_p^{\mathbb{K}(C)^{n(C)}}$  it holds that*

$$\text{rej}(\underline{c}) \geq k(C) \frac{1}{2^{15} p^4} \min \left\{ \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|, \frac{1}{k(C)^3} \right\}.$$

We compare this result to the (non-amplified) local testing for affine invariant codes of Kaufman and Sudan [10, Theorem 2.9] who showed the following:

► **Theorem 7** ([10, Theorem 2.9]). *For every  $C \in \mathcal{C}_{\text{affine-inv},p}$  it holds that*

$$\text{rej}(\underline{c}) \geq \frac{1}{2} \min \left\{ \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|, \frac{1}{k(C)^2} \right\}.$$

Our Theorem and [10, Theorem 2.9] both give a rejection of  $\Omega(\frac{1}{k(C)^2})$  when  $\min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|$  is large. However, when  $\min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\| < \frac{1}{k(C)^3}$ , and  $k(C)$  is large, our result gives a much better rejection rate.

### Local testability via unique neighbor expansion

We show that  $\lambda$ -expanding HDE-System has some form of unique neighbor expansion property associated with it. We also show that if the HDE-system has a strong enough unique neighbor expansion property, then a linear code defined based on this system is amplified locally testable. We prove that this is the case for affine-invariant codes with the single orbit property. Thus, HDE-system provides a mechanism to get amplified local testability of codes.

## 2 Comparison to prior works

We already mentioned the celebrated work of Dinur at el. [4] that uses ideas from high dimensional expansion to construct locally testable codes with constant rate, distance and locality. As noted above, our work is in a different direction and achieves different goals (we do not achieve the result of [4], but do achieve amplified local testability).

Another work that seems superficially close to the methods of this paper is the recent work of Dikstein at el. [3] that also relies on ideas from high dimensional expansion to deduce local testability. The reader should note that there are major difference between the works:

- Our work has the benefit of deducing not only local testability, but rather amplified local testability which was not achieved in [3].
- As far as we know, the work of [3] does not apply to the family of affine invariant codes, but only to a sub-family of lifted codes. Thus, in terms of generality, our work seems to apply in a more general setting.
- The work of Dikstein at el. [3] relies on the idea that “global” local testability can be inferred from “local” local testability. I.e., in [3], the assumption is that the code contains many small (i.e., “local”) locally testable codes and by expansion considerations, it follows that the global code is locally testable. This is also the point of view of [7, 5, 8] that considered what can be thought of as “co-cycle codes” and the global testability was derived assuming they are composed of small local codes that are locally testable (aka “the links” code). In contrast to [3] (and to [7, 5, 8]), the focus of this current work is to get local testability of codes *directly* from high dimensional expansion phenomenon. Deducing local testability of codes directly from high dimensional expansion (without relying on any local code that is locally testable) is new and is achieved here for the first time.

It is also beneficial to compare the results of this paper to previous results regarding single orbit affine invariant codes. In [10, Theorem 2.9], it was shown that single orbit affine invariant codes are locally testable. Using our new machinery, we improve on this result, showing the the family of all single orbit affine invariant codes has amplified local testability. As noted above, a stronger result was known for Reed-Muller codes (which is a sub-family of the family of affine invariant codes), but, prior to our work, no general treatment was available to the entire family of single orbit affine invariant codes.

### 3 High Dimensional Expanding System (HDE-System)

Our main definition towards defining high dimensional expander codes is called High-Dimensional-Expanding-System or HDE-System for short.

We start by defining a  $(s, k, K)$ -Two layer system:

► **Definition 8** ( $(s, k, K)$ -Two layer system). *A two layer system  $X$  is a system  $X = (V, E, T)$  of three sets:*

1. *A finite set  $V$  whose elements are called vertices.*
2. *A set  $E \subseteq 2^V$  such that  $|\tau| = k$  for every  $\tau \in E$  and  $\bigcup_{\tau \in E} \tau = V$ .*
3. *A set  $T \subseteq 2^E$  such that  $|\sigma| = K$  for every  $\sigma \in T$  and  $\bigcup_{\sigma \in T} \sigma = E$ .*
4. *By abuse of notation, we will denote  $v \in \sigma$  for  $v \in V, \sigma \in T$  if there is  $\tau \in \sigma$  such that  $v \in \tau$ . Using this notation, for every  $\sigma \in T$  and every  $v \in \sigma$ ,*

$$2 \leq |\{\tau \in \sigma : v \in \tau\}| \leq s.$$

Roughly speaking, HDE-System is a two layer system with good expansion properties. In order to give the definition, we need to define several graphs associated with a two layer system. We note that all the graphs defined below will be actually considered as weighted graphs with a weight function induced by weights on  $T$ , but in the introduction we suppress this fact in order to keep things simple.

► **Definition 9** (The ground graph). *For a two layer system  $X = (V, E, T)$ , the ground graph of  $X$  is the graph whose vertices are  $V$  and edges are  $\{\{v, u\} : \exists \tau \in E, u, v \in \tau\}$ .*

► **Definition 10** (Link of a vertex). *For a two layer system  $X = (V, E, T)$  and  $v \in V$ , the link of  $v$  is the graph whose vertex set is  $E_v = \{\tau \in E : v \in \tau\}$  and whose edge set is*

$$T_v = \{\{\tau, \tau'\} : \tau \neq \tau' \text{ and } \exists \sigma \in T \text{ such that } \tau, \tau' \in \sigma\}.$$

► **Definition 11** (The non-intersecting graph). *For a two layer system  $X = (V, E, T)$ , the non-intersecting graph of  $X$  is a graph whose vertex set is  $E$  and edge set is*

$$\{\{\tau, \tau'\} : \tau \cap \tau' = \emptyset \text{ and } \exists \sigma \in T, \text{ such that } \tau, \tau' \in \sigma\}.$$

*This graph corresponds to the Non-Intersecting Walk, i.e., to the walk from a between elements of  $E$  that do NOT intersect (as subsets of  $V$ ) via a  $T$  element that contains both of them.*

An HDE-System is a two layer system  $X$  in which all these graphs are expanding. More precisely, for  $0 \leq \lambda < 1$ , we call a (weighted) graph  $G$  a  $\lambda$ -expander if it is connected and either the second largest eigenvalue of the is  $\leq \lambda$  or (which is less restrictive) its (generalized) Cheeger constant is  $\geq 1 - \lambda$  (see related full version of this paper for exact definition).

► **Definition 12** (High Dimensional Expanding System (HDE-System) – informal, for formal see the related full version of this paper). *For  $0 \leq \lambda < 1$ , a (weighted) two layer system  $X = (V, E, T)$  is called  $\lambda$ -expanding-HDE-System if the ground graph and the links of all the vertices are  $\lambda$ -expanders and the non-intersecting graph is either totally disconnected (i.e., it has no edges) or a  $\lambda$ -expander.*

### High Dimensional expanders imply HDE-System

Part of our motivation for the Definition of HDE-systems is to mimic the definition of high dimensional expanders based on simplicial complexes (called  $\lambda$ -local spectral expander – see [9, Definitions 2,3]). The simplest example is when  $Y$  is a 2-dimensional simplicial complex. In this case, we define a two layer system  $X = (V, E, T)$  as follows:  $V$  is the vertex set of  $Y$ ,  $E$  is the edge set of  $Y$  and  $T$  is the sets of triples of edges that form a triangle in  $Y$ . We note that in this case the parameters of  $X$  are  $s = 2, k = 2, K = 3$ . Note that the ground graph is the 1-skeleton of  $Y$ , the link of each vertex in  $X$  is the link in the simplicial complex and the non-intersecting graph is totally disconnected (since every two edges that are in the same triangle share a vertex). Thus, by definition if  $Y$  is a  $\lambda$ -local spectral expander, then the 1-skeleton of  $Y$  and all the links of  $Y$  are  $\lambda$ -expanders and it follows that  $X$  is  $\lambda$ -expanding.

### Expanding HDE's have unique neighbor expansion for small sets that are also locally small

Our main motivation for the definition of HDE-System is the ability to deduce unique neighbor expansion theorem from them, for “small” sets that are also “locally small”. This unique neighbor expansion theorem that we state below will play a major role in proving local testability based on HDE-System.

In order to state this Theorem, we will need the following definition:

► **Definition 13** ( $\delta$ -Locally-small set – informal, for formal see the related full version of this paper). *Let  $X = (V, E, T)$  be a two layer system and let  $A \subseteq E$  be a non-empty set. For a vertex  $v \in V$ , define  $A_v = \{\tau \in A : v \in \tau\}$ . For a constant  $0 \leq \delta < 1$ , a vertex  $v$  is called  $\delta$ -small if the size of  $A_v$  in the link of  $v$  (when accounting for the weight function on the link) is smaller than  $\delta$  fraction of the size of  $E_v$ . Vertices that are not  $\delta$ -small are called  $\delta$ -large. A set  $A \subseteq E$  is called  $\delta$ -locally small, if the fraction of its mass that is distributed on vertices that are  $\delta$ -large is negligible with respect to the total mass of  $A$ .*

Following we define a notion of unique neighbor expansion that applies for small sets that are also  $\delta$ -locally small.

► **Definition 14** (Unique neighbor expansion property – informal, for formal see the related full version of this paper). *We say that  $A \subset E$  has a unique neighbor expansion into  $T$  if there exists  $\sigma \in T$  that contains exactly one  $k$ -set from  $A$ . Let  $X = (V, E, T)$  be a two layer system and let  $A \subseteq E$  be a non-empty set. For constants  $\varepsilon_0 > 0$ ,  $\delta > 0$ , we say that  $X$  has  $(\delta, \varepsilon_0)$ -unique neighbor expansion property if for every non-empty set  $A \subseteq E$  and every  $\varepsilon < \varepsilon_0$  if  $A$  is  $\varepsilon$ -small (i.e., its mass is at most a  $\varepsilon$ -fraction of the total mass of  $E$ ) and  $\delta$ -locally small, then  $A$  has unique neighbor expansion into  $T$ .*

► **Theorem 15 (Main Theorem 1: Unique neighbor expansion property for HDE-System – informal, for formal see the related full version of this paper).** *Given a  $\lambda$ -expanding HDE-System  $X$ , with  $\lambda$  sufficiently small, there are  $\delta > 0$  and  $\varepsilon_0 > 0$  such that  $X$  has the  $(\delta, \varepsilon_0)$ -unique neighbor expansion property. Moreover, if  $s = 2$ , then  $\delta \rightarrow 1$  as  $\lambda \rightarrow 0$ .*

### On the ability to get unique neighbor expansion from HDE-systems

The idea behind the proof of Main Theorem 1 is to use the expansion of the links in order to derive unique neighbor expansion. The links are very good expanders so a set that is locally small has the property that its local views in the links expand a lot. Each link induces by its local view many “potential unique neighbors”. However, it could be that the local views

of the links will interfere and the “potential unique neighbors” by the “links opinion” will turn out to be non unique neighbors. Since the system is expanding the total interference between links is small and thus the overall unique neighbor property is implied.

#### 4 HDE-System Codes

Given a two layer system  $X = (V, E, T)$  as above, we want to use it as a “foundation” and for constructing a code. Such a construction is not unique and cannot be done for every  $X$ . However, for a code that “could be constructed via  $X$ ”, its testability could be inferred from the expansion properties of  $X$ .

Before describing this construction, we need to establish some terminology and notation: Let  $C \subseteq \mathbb{F}_p^V$  be a linear code (where  $V$  is a finite set) with a check matrix  $H$ .

- We denote by  $\mathcal{E} = \mathcal{E}(H)$  the rows  $H$  and we refer to  $\mathcal{E}$  as the constraints of the code (or  $k$ -constraints if they all have a support of size  $k$  – see below). Thus,  $\mathcal{E}$  are  $1 \times n$  vectors and for  $\underline{c} \in \mathbb{F}_p^V$ ,  $\underline{c} \in C$  if and only if for every  $\underline{e} \in \mathcal{E}$ ,  $\underline{e} \cdot \underline{c} = 0$  (recall that  $\underline{e}, \underline{c}$  are indexed by the elements in  $V$ , thus  $\underline{e} \cdot \underline{c} = \sum_v \underline{e}(v)\underline{c}(v)$ ).
- For  $\underline{e} \in \mathcal{E}$ , we define the support of  $\underline{e}$  as

$$\text{supp}(\underline{e}) = \{v \in V : \underline{e}(v) \neq 0\}.$$

- A *linear dependency* of  $\mathcal{E}$  is a function  $\text{ld} : \mathcal{E} \rightarrow \mathbb{F}_p$  such that for every  $\underline{c} \in \mathbb{F}_p^V$ ,  $\sum_{\underline{e} \in \mathcal{E}} \text{ld}(\underline{e})(\underline{e} \cdot \underline{c}) = 0$ . In other words, if we think of the row vector  $\underline{\text{ld}} = (\text{ld}(\underline{e}))_{\underline{e} \in \mathcal{E}}$ , then  $\underline{\text{ld}}H = \underline{0}$ . As above, the support of  $\text{ld}$  is the set

$$\text{supp}(\text{ld}) = \{\underline{e} \in \mathcal{E} : \text{ld}(\underline{e}) \neq 0\}.$$

► **Example 16.** Consider  $C \subseteq \mathbb{F}_2^V$ ,  $V = \{v_1, v_2\}$  given by the parity check matrix

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

If  $\underline{e}_i$  denotes the  $i$ -th row of  $H$ , then  $\text{ld} : \{\underline{e}_1, \underline{e}_2, \underline{e}_3\} \rightarrow \mathbb{F}_2$  defined by

$$\text{ld}(\underline{e}_i) = 1, \forall i = 1, 2, 3,$$

is a linear dependency. Indeed,

$$\underline{\text{ld}} = (1 \quad 1 \quad 1),$$

and one can verify that  $\underline{\text{ld}}H = \underline{0}$ .

► **Definition 17** (Code modelled over a two layer system). *Let  $X = (V, E, T)$  be a two layer system. A code  $C$  is said to be modelled over  $X$  if the following holds:*

- *There is a prime power  $p$  such that  $C \subseteq \mathbb{F}_p^V$ .*
- *There is a check matrix  $H$  and  $\mathcal{E} = \mathcal{E}(H)$  such that*

$$E = \{\text{supp}(\underline{e}) : \underline{e} \in \mathcal{E}\},$$

*and such that for every  $\underline{e}_1, \underline{e}_2 \in \mathcal{E}$ , if  $\underline{e}_1 \neq \underline{e}_2$ , then  $\text{supp}(\underline{e}_1) \neq \text{supp}(\underline{e}_2)$ . In other words, there is a bijection  $\Phi : \mathcal{E} \rightarrow E$  given by  $\Phi(\underline{e}) = \text{supp}(\underline{e})$ . Note that under this assumption, the size of the support of all the constraints is  $k$  (the constant of the system  $X$ ) and we refer to the elements of  $\mathcal{E}$  as the  $k$ -constraints of the code, when there is no chance for ambiguity.*



■ There is a set  $\mathcal{T}$  of linear dependencies such that

$$T = \{\{\text{supp}(\underline{e}) : \underline{e} \in \text{supp}(\text{ld})\} : \text{ld} \in \mathcal{T}\}.$$

► **Example 18.** Let  $X = (V, E, T)$  the following two layer system:  $V = \{v_1, v_2, v_3\}$ ,  $E = \{\tau_{i,j} = \{v_i, v_j\} : 1 \leq i < j \leq 3\}$  and  $T = \{\sigma = \{\tau_{i,j} : 1 \leq i < j \leq 3\}\}$ . Then for every prime power  $p$ , we can define a code  $C \subseteq \mathbb{F}_p^V$  modelled over  $X$  as follows: define the check matrix of the code to be

$$H = \begin{pmatrix} 1 & p-1 & 0 \\ 0 & 1 & p-1 \\ p-1 & 0 & 1 \end{pmatrix}.$$

One can see that for this matrix the support of the  $i$ -th row is  $\{v_i, v_{i+1 \bmod 3}\} \in E$  and that no two rows have the same support. Further define a linear dependency  $\text{ld} : \mathcal{E} \rightarrow \mathbb{F}_p$  to be the constant function 1, thus one can verify that the support of  $\text{ld}$  is  $\sigma \in T$  and that this is indeed a linear dependency.

Our motivation for considering codes modelled over two layer system is the following

► **Theorem 19 (Main Theorem 2:** Codes modelled over two layer systems with unique neighbor property are amplified locally testable – informal, for formal see the related full version of this paper). *For every  $p$  prime,  $t' \in \mathbb{N}, t' > 0$ ,  $\mu > 0$  and  $\delta > \frac{p-1}{p}$ , let  $\mathcal{C}(\delta, p, t', \mu)$  be the family of  $p$ -ary codes (i.e., codes of the form  $C \subseteq \mathbb{F}_p^{V(C)}$ ) modelled over two layer systems such that*

$$\mathcal{C}(\delta, p, t', \mu) = \left\{ C : \exists \varepsilon_0(C) > 0 \text{ such that } C \text{ has the } (\delta, \varepsilon_0(C))\text{-unique neighbor property and } \varepsilon_0 \geq \frac{\mu}{k(C)^{t'}} \right\}.$$

*Then the family  $\mathcal{C}(\delta, p, t', \mu)$  is amplified locally testable with  $t_C = t' + 1$ .*

### On the ability to get amplified local testability from unique neighbor expansion

We will explain how to get amplified local testability from unique neighbor expansion for  $p = 2$  (this is to avoid carrying  $p$  as a constant) and  $\delta = \frac{3}{4}$ .

We assume we are in a situation that we have a code that is modelled over an HDE-system. Thus, we know that each  $k$ -constraint of the code is participating in a linear dependency. This means that on every dependency, if there is one violated constraint that touches it, there must be another one that touches it.

We are given a vector  $\underline{c}$  that falsifies  $|A|$  constraints from the code and we want to show that such a vector is close to the code. We can try to correct it by flipping variables such that this flipping reduces the number of violated constraints by a fixed proportion: Assume that each bit is a member of  $N$  equations and we change the value of the bit if the number of falsified equations containing it is more than  $\frac{3}{4}N$ . In this case, flipping the bit corrects many equations (since all the equations that were false are now true and vice-versa): i.e., flipping the bit corrects at least  $\frac{1}{2}N$  equations. Let us compute what is the maximal number of steps for such a correcting procedure to stop: we assumed that there were the corrupted code word had  $|A|$  falsified equations, i.e.,  $\text{rej}(\underline{c}) = \frac{|A|}{|E|}$ . Thus, the number of bits flipping in the correction procedure is at most  $\frac{|A|}{\frac{1}{2}N} = \frac{2|A|}{N}$ . and in the end of this procedure, each vertex is  $\frac{3}{4}$ -locally small.

Assume the HDE-System on which the code is modelled has  $(\delta, \varepsilon_0)$ -unique neighbor expansion and that  $\frac{|A|}{|E|} < \varepsilon_0$ . The unique neighbor expansion implies that there are linear



dependencies that “sees” only one violating constraint. However, as we said, this is not possible. So when we arrived at a situation where no more flipping is possible, we, in fact, arrived at a codeword that is close to our initial vector. Explicitly, the fraction of flipped bits is less or equal to  $\frac{2|A|}{|V|}$ .

Note that if  $|V|$  denotes the number of bits, then the number of equations is  $|E| = \frac{|V|N}{k}$  (each equation contains  $k$ -bits and each bit is a member of  $N$  equations). It follows that the number of flipped bits is less or equal to

$$\frac{2|A|}{|V|} = \frac{2|A|}{|E|} \frac{|E|}{|V|} = \frac{2|E|}{N|V|} \frac{|A|}{|E|} = \frac{2}{k} \text{rej}(\underline{c})$$

and thus

$$\text{rej}(\underline{c}) \geq k \frac{1}{2} \min_{\underline{c}' \in C} \|\underline{c} - \underline{c}'\|$$

as needed.

► **Definition 20** (HDE-System Code). *We call a code  $C$  as above a HDE-System-code if it is modelled over a  $\lambda$ -expanding HDE-System.*

### Codes that give rise to HDE-System with $s = 2$ are amplified locally testable

By Main Theorem 2, the family  $\mathcal{C}_{\delta,p}$  of codes  $C \subseteq \mathbb{F}_p^V$  modelled a two layer systems with a  $(\delta, \varepsilon_0(C))$ -unique neighbor property are locally testable given that  $\delta > \frac{p-1}{p}$ . We have furthered showed (see Main Theorem 1 above) that given any  $\delta < 1$ , there is  $\lambda$  sufficiently small such that every  $\lambda$ -expanding HDE-System with  $s = 2$  has the  $(\delta, \varepsilon_0)$ -unique neighbor property (where  $\varepsilon_0$  depends on the parameters of the HDE-System). Thus, overall we get that the family of all codes  $C \subseteq \mathbb{F}_p^V$  modelled over  $\lambda$ -expanding HDE-System with  $s = 2$  (and  $\lambda$  sufficiently small) is amplified locally testable.

► **Corollary 21** (Codes modelled over expanding-HDE-System with  $s = 2$  are amplified locally testable – informal, for formal see the related full version of this paper). *The family of all codes  $C \subseteq \mathbb{F}_p^V$  of  $k(C)$ -constraints modelled over expanding HDE systems with  $s = 2$  is amplified locally testable. Moreover, under some mild assumptions (passing to a large sub-family)  $t_C = 3$  where  $t_C$  is as in Definition 3.*

Above, we stated Theorem 6 regarding amplified local testability of single orbit affine invariant codes. This Theorem is deduced from the above Corollary, because we show that single orbit affine invariant codes are modelled over expanding HDE systems with  $s = 2$ .

### Local testability when $s > 2$

The main focus of this work is proving amplified local testability for codes modelled over HDE-systems with  $s = 2$  as all our examples satisfy the  $s = 2$  assumption (Reed-Muller codes and single orbit affine invariant codes satisfy  $s = 2$ ). We further have a more general treatment for codes modelled over HDE-system with general  $s \geq 3$  under some extra-assumptions (although we currently do not have examples for such codes). Roughly speaking, for the case of  $s \geq 3$  we need the extra assumption that the code is composed of local small codes that are locally testable. The difficulty in the case where  $s \geq 3$  is that the bit flipping argument we described above can only correct the code to be  $\frac{p-1}{p}$ -locally small, while “The unique neighbor Theorem” says that we can deduce the  $(\delta, \varepsilon_0)$ -unique neighbor property from

expansion given that  $\delta < \frac{1}{s-1}$ . Thus, in the case where  $s \geq 3$ , we may not be able correct a corrupted codeword by bit flipping to a setting in which we can apply our unique neighbor argument. This difficulty is dealt by adding the assumption of “local” local testability that guarantees that correcting by bit flipping converges to a word that is  $\delta$ -locally small (and thus we can use our previous machinery). This new method requires some additional definitions and we refer the reader to the related full version of this paper for further details.

### Distance of HDE codes

An additional result is that for codes modelled over HDE-systems, the distance of the code can be bounded in terms of the expansion of the HDE system.

---

### References

- 1 Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Trans. Inform. Theory*, 51(11):4032–4039, 2005. doi:10.1109/TIT.2005.856958.
- 2 Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of Reed-Muller codes. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010*, pages 488–497. IEEE Computer Soc., Los Alamitos, CA, 2010.
- 3 Yotam Dikstein, Irit Dinur, Prahladh Harsha, and Noga Ron-Zewi. Locally testable codes via high-dimensional expanders, 2020. arXiv:2005.01045.
- 4 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahrar Mozes. Locally testable codes with constant rate, distance, and locality, 2021. arXiv:2111.04808.
- 5 Shai Evra and Tali Kaufman. Bounded degree cosystolic expanders of every dimension. In *STOC’16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 36–48. ACM, New York, 2016. doi:10.1145/2897518.2897543.
- 6 Elad Haramaty, Noga Ron-Zewi, and Madhu Sudan. Absolutely sound testing of lifted codes. *Theory Comput.*, 11:299–338, 2015. doi:10.4086/toc.2015.v011a012.
- 7 Tali Kaufman, David Kazhdan, and Alexander Lubotzky. Ramanujan complexes and bounded degree topological expanders. In *55th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2014*, pages 484–493. IEEE Computer Soc., Los Alamitos, CA, 2014. doi:10.1109/FOCS.2014.58.
- 8 Tali Kaufman and David Mass. Cosystolic expanders over any abelian group. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:134, 2018. URL: <https://eccc.weizmann.ac.il/report/2018/134>.
- 9 Tali Kaufman and Izhar Oppenheim. High order random walks: beyond spectral gap. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 116 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 47, 17. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.
- 10 Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *STOC’08*, pages 403–412. ACM, New York, 2008. doi:10.1145/1374376.1374434.

# Polynomial Bounds on Parallel Repetition for All 3-Player Games with Binary Inputs

Uma Girish ✉

Princeton University, NJ, USA

Kunal Mittal ✉

Princeton University, NJ, USA

Ran Raz ✉

Princeton University, NJ, USA

Wei Zhan ✉

Princeton University, NJ, USA

---

## Abstract

We prove that for every 3-player (3-prover) game  $\mathcal{G}$  with value less than one, whose query distribution has the support  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$  of Hamming weight one vectors, the value of the  $n$ -fold parallel repetition  $\mathcal{G}^{\otimes n}$  decays polynomially fast to zero; that is, there is a constant  $c = c(\mathcal{G}) > 0$  such that the value of the game  $\mathcal{G}^{\otimes n}$  is at most  $n^{-c}$ .

Following the recent work of Girish, Holmgren, Mittal, Raz and Zhan (STOC 2022), our result is the missing piece that implies a similar bound for a much more general class of multiplayer games: For **every** 3-player game  $\mathcal{G}$  over *binary questions* and *arbitrary answer lengths*, with value less than 1, there is a constant  $c = c(\mathcal{G}) > 0$  such that the value of the game  $\mathcal{G}^{\otimes n}$  is at most  $n^{-c}$ .

Our proof technique is new and requires many new ideas. For example, we make use of the Level- $k$  inequalities from Boolean Fourier Analysis, which, to the best of our knowledge, have not been explored in this context prior to our work.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational complexity and cryptography

**Keywords and phrases** Parallel repetition, Multi-prover games, Fourier analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.6

**Category** RANDOM

**Funding** *Uma Girish*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award, by the National Science Foundation grants No. CCF-1714779, CCF-2007462 and by the IBM Phd Fellowship.

*Kunal Mittal*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Wei Zhan*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

**Acknowledgements** We thank Justin Holmgren for important conversations and collaboration in early stages of this work.



© Uma Girish, Kunal Mittal, Ran Raz, and Wei Zhan;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 6; pp. 6:1–6:17



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Our main object of study is multiplayer (multiprover) games. A  $k$ -player game  $\mathcal{G}$  consists of  $k$  players who are playing against a referee. The game begins by the referee sampling a  $k$ -tuple of questions  $(x^1, \dots, x^k)$  from some global distribution  $Q$ . The referee then gives the question  $x^j$  to the  $j^{\text{th}}$  player, for each  $j \in [k]$ , based on which they give back an answer  $a^j$ . Finally, the referee evaluates a predicate  $V((x^1, \dots, x^k), (a^1, \dots, a^k))$  and says that the players win if and only if the predicate evaluates to true. The value  $\text{val}(\mathcal{G})$  of the game is defined to be the maximum winning probability for the players. Note that the probability here is over the randomness used by the referee to sample  $(x^1, \dots, x^k) \sim Q$ , and the maximum is over the strategies used by the players.

Given a game  $\mathcal{G}$  with value  $\text{val}(\mathcal{G}) < 1$ , it is natural to consider the parallel repetition of the game  $\mathcal{G}$ , defined as follows: In the  $n$ -fold repetition  $\mathcal{G}^{\otimes n}$  of the game  $\mathcal{G}$ , the referee independently samples questions for  $n$  copies of the game  $\mathcal{G}$ ; that is, the referee samples  $(x_i^1, \dots, x_i^k) \sim Q$  independently for  $i \in [n]$ . Then, the referee *simultaneously* gives questions  $x_1^j, \dots, x_n^j$  to the  $j^{\text{th}}$  player, for each  $j \in [k]$ , who then gives back answers  $a_1^j, \dots, a_n^j$ . The players are said to win the game if for each  $i \in [n]$ , the predicate  $V((x_i^1, \dots, x_i^k), (a_i^1, \dots, a_i^k))$  evaluates to true.

With the above definition of the  $n$ -fold repeated game  $\mathcal{G}^{\otimes n}$ , it is interesting to study the behavior of  $\text{val}(\mathcal{G}^{\otimes n})$  with respect to  $n$ , and the initial parameters of the game  $\mathcal{G}$  [14]. Observe that  $\text{val}(\mathcal{G}^{\otimes n}) \geq \text{val}(\mathcal{G})^n$ , since any strategy that achieves value  $\text{val}(\mathcal{G})$  in the game  $\mathcal{G}$ , when repeated independently for all copies  $i \in [n]$ , achieves the value  $\text{val}(\mathcal{G})^n$  in the game  $\mathcal{G}^{\otimes n}$ . While one would expect such an inequality to be tight, this is far from true; there are games such that  $\text{val}(\mathcal{G}^{\otimes n})$  is exponentially larger (with respect to  $n$ ) compared to  $\text{val}(\mathcal{G})^n$ . The crucial reason why this can happen is that in the game  $\mathcal{G}^{\otimes n}$  the players are allowed to correlate their answers among different copies  $i \in [n]$  of the game. That is, it is not necessary (and not optimal) for every player's answer for the  $i^{\text{th}}$  copy of the game to depend only on the  $i^{\text{th}}$  question they receive.

Nevertheless, Raz [30] proved that for any 2-player game  $\mathcal{G}$  with  $\text{val}(\mathcal{G}) < 1$ , it holds that  $\text{val}(\mathcal{G}^{\otimes n}) = 2^{-\Omega(n)}$ . This, and related techniques and results, turned out to be sufficient for a large number of applications: in the theory of interactive proofs [5], PCPs and hardness of approximation [4, 11, 18], geometry of foams [12, 23, 1], quantum information [8], and communication complexity [27, 2, 7]. The reader is referred to this survey [31] for more details. There have been many subsequent improvements that improve the constants in the bounds, and even get better bounds based on the value  $\text{val}(\mathcal{G})$  of the initial game [20, 29, 3, 32, 10, 6].

The case of 2-player games, hence, is fairly well-understood with regards to the operation of parallel repetition. On the other hand, despite much effort, the general question of parallel repetition for multiplayer games remains wide open. The only general bound, by [33], that applies to all  $k$ -player games, says that if  $\text{val}(\mathcal{G}) < 1$ , then  $\text{val}(\mathcal{G}^{\otimes n}) \leq \frac{1}{\alpha(n)}$ , where  $\alpha(n)$  is a function which grows like the (extremely slowly growing) inverse Ackermann function. The weak bounds here result from a black-box use of the Density Hales-Jewett Theorem [15, 28] from extremal combinatorics.

While there are some known potential applications of bounds on parallel repetition of multiplayer games, for example, [24] show a connection between parallel repetition and super-linear lower bounds for non-uniform Turing machines, we believe that the notion of parallel repetition is so basic that it deserves attention in its own right. As mentioned by [9], there are many problems in complexity theory that are inherently high dimensional, and which share this sudden difficulty of being tractable beyond dimension two. For example, whereas direct sum and direct product theorems are known for two-party communication

complexity, no such results are known for multiparty communication complexity in the number-on-forehead model (which is deeply related to proving new lower bounds in circuit complexity), for seemingly similar reasons to why there has not been much progress on multiplayer parallel repetition.

Recent work, however, has made some progress on proving parallel repetition bounds for some special classes of multiplayer games:

1. **Connected Games:** Dinur, Harsha, Venkat and Yuen [9] consider games which satisfy a certain connectedness property and show that the value of any such game satisfies an exponential decay bound under parallel repetition: if  $\text{val}(\mathcal{G}) < 1$  then  $\text{val}(\mathcal{G}^{\otimes n}) = 2^{-\Omega(n)}$ . A  $k$ -player game  $\mathcal{G}$  is said to have this connectedness property if the graph  $\mathcal{H}_{\mathcal{G}}$  defined as follows is connected: The vertices of the graph are all the possible  $k$ -tuples of questions to the players (which are asked with non-zero probability), and there is an edge between two such  $k$ -tuples if they differ in the question to exactly one of the  $k$  players. The proof for these games uses information theoretic techniques, and builds on the works on 2-player games by [30, 20].
2. **The GHZ Games:** [21, 16] show that any game  $\mathcal{G}$  over the set of questions

$$\left\{ (x, y, z) \in \{0, 1\}^3 : x + y + z = 0 \pmod{2} \right\}$$

satisfies a polynomial bound on the value of parallel repetition: if  $\text{val}(\mathcal{G}) < 1$  then  $\text{val}(\mathcal{G}^{\otimes n}) = n^{-\Omega(1)}$ . For such games, all vertices in the graph  $\mathcal{H}_{\mathcal{G}}$  (as defined above in point 1) are isolated, and the techniques of [9] fail to be applicable.

The known proofs for this case use Fourier analytic techniques that crucially rely on the fact that the inputs to the players define a linear subspace of  $\mathbb{F}_2^3$ .

3. A recent work [17] considers the problem of parallel repetition for 3-player games in which each player is asked a binary question. They do a case analysis of all such games and divide the general problem into the following cases:
  - a. Connected games or games that are essentially 2-player games: An exponential decay bound is known [30, 9].
  - b. Games over the question set of the GHZ game (see point 2): A polynomial decay bound is known.
  - c. Games over the question set  $\left\{ (x, y, z) \in \{0, 1\}^3 : x + y + z \neq 2 \right\}$ : They show that such games fall into a class of games which they call **playerwise connected games**, a generalization of the class of connected games. Informally, a game  $\mathcal{G}$  is said to be playerwise connected if the *projection* of the graph  $\mathcal{H}_{\mathcal{G}}$  onto each of the  $k$ -players is connected. They show that any playerwise connected game satisfies a polynomial decay bound in the value of parallel repetition: if  $\text{val}(\mathcal{G}) < 1$  then  $\text{val}(\mathcal{G}^{\otimes n}) = n^{-\Omega(1)}$ .
  - d. Games over the question set  $\left\{ (x, y, z) \in \{0, 1\}^3 : z = xy \right\}$ : They call this the four-point AND distribution, and show that any such game satisfies a polynomial bound in the value of parallel repetition.
  - e. Games over the set of questions  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$  of Hamming weight one: They do not prove a general bound for games in this class, but rather only for games where the answers given by each of the three players is also binary. Under this extra assumption, they are in fact able to prove an exponential decay bound under parallel repetition.

A very interesting game which they consider is the **anti-correlation game** defined as follows: The referee samples the questions  $(x^1, x^2, x^3) \in \mathcal{S}$  uniformly at random, and the two players who receive the input 0 must produce different outputs in  $\{0, 1\}$ . This game has the special property that while its *non-signalling* value is less than 1, the non-signalling value does not decrease at all under parallel repetition [22].

The main topic of interest of the current paper are games described above in point 3e, that is, all games over the question set  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ . The work [17] shows that any bounds for a special subclass of such games qualitatively translate to the same bounds for all games in this class. In particular, polynomial decay bounds for the value of parallel repetition for the following subclass of games implies polynomial decay bounds for all games over the question set  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ :

► **Definition 1.** Let  $k \in \mathbb{N}$ , and let  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ . We define a 3-player game  $\mathcal{G}_k$  on 3 players Alice, Bob and Charlie as follows:

1. The referee samples  $(x, y, z) \in \mathcal{S}$  uniformly at random, and gives  $x, y, z$  to the three players respectively.
2. The players answer  $a \in \{0, 1\}^k, b \in \{0, 1\}^k, c \in [k]$  respectively.
3. The winning predicate is defined as:

$$V_k((x, y, z), (a, b, c)) = \begin{cases} b_c = 0, & \text{if } (x, y, z) = (1, 0, 0) \\ a_c = 0, & \text{if } (x, y, z) = (0, 1, 0) \\ \forall i \in [k], a_i + b_i \geq 1, & \text{if } (x, y, z) = (0, 0, 1) \end{cases}.$$

In other words, two randomly chosen players receive 0 as input and the third player gets a 1 as input. The predicate only depends on the two players who get 0 as input, and only those two players play the game. If Charlie and Alice (or Bob) are playing, Charlie must point to an index where Alice (or Bob) outputs 0. On the other hand, if Alice and Bob are playing, they must each output  $k$ -bit strings such that the bit-wise-OR of the two strings is the all 1s string.

Our main result is a polynomial decay bound on the parallel repetition for all games in the above subclass:

► **Theorem 2.** There exists an absolute constant  $c > 0$  such that the following holds: For every  $k \in \mathbb{N}$ , and for every sufficiently large  $n \in \mathbb{N}$ , it holds that  $\text{val}(\mathcal{G}_k^{\otimes n}) \leq n^{-c}$ , where the game  $\mathcal{G}_k$  is as defined in Definition 1.

Based on the previous discussion, combined with the works [30, 9, 21, 16, 17], our theorem implies the following:

► **Theorem 3.** Let  $\mathcal{G}$  be any 3-player game over binary questions, and arbitrary finite length answers, such that  $\text{val}(\mathcal{G}) < 1$ . Then, there exists a constant  $c = c(\mathcal{G}) > 0$ , such that for every  $n \in \mathbb{N}$ , it holds that  $\text{val}(\mathcal{G}_k^{\otimes n}) \leq n^{-c}$ .

We remark that Hazla, Holenstein and Rao [19] consider games over the same question set  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ , and show barriers for proving parallel repetition bounds for such games using the *forbidden subgraph method* [13]. Our result builds new techniques that do not fit into the above framework, and are able to bypass these barriers.

Next, in Section 1.1, we give an overview of the proof of Theorem 2. We note that our proof introduces several new ideas, which we believe are very general and can possibly extend to much larger classes of games. For example, in one of the steps, we use Fourier Analysis over the boolean hypercube, and in particular the Level- $k$  inequalities; to the best of our knowledge, such use in the context of parallel repetition is new.



## 1.1 Proof Overview

Fix some  $k \in \mathbb{N}$  and consider the  $n$ -fold repeated game  $\mathcal{G}_k^{\otimes n}$  (see Definition 1). We'll use the term *coordinate* to mean a tuple  $(i, j)$  with  $i \in [n]$  and  $j \in [k]$ , that indexes an answer for Alice or Bob. Recall that in each copy of the game  $\mathcal{G}_k$ , only the two players who receive input 0 affect the winning predicate, and we say that they are the ones who *play*.

The high-level intuition is as follows: In order to win, Alice and Bob cannot both answer 0 at the same coordinate. On the other hand, suppose that they indeed only answer 0 in two fixed disjoint subsets of coordinates each of their own, then Charlie's answer in each copy of the game  $\mathcal{G}_k$  actually reveals which player he is playing with, which is too much information for Charlie to have.

We note, however, that this intuition is too simplistic and the actual proof is much more complicated, because in each coordinate only two out of the 3 players play. Nevertheless, our proof can be viewed as a rigorous execution of the intuition, by finding a large enough product event  $E_1 \times E_2$  on Alice's and Bob's inputs in which the above presumption holds true. More specifically, to prove by contradiction we assume that the winning probability is at least  $n^{-c}$  (where  $c > 0$  is a small constant), and the proof is carried out in three steps:

### Remove coordinates that Alice and Bob lose (Section 4)

We remove the coordinates where Alice and Bob both play and simultaneously output 0 with non-negligible (at least  $n^{-O(c)}$ ) probability, by fixing their inputs and outputs in these coordinates. The fixing of outputs gives rise to the product event  $E_1 \times E_2$  on the remaining coordinates. We need to ensure that the probability of both  $E_2$  and the winning event  $W$  remain  $n^{-O(c)}$ , while the rounds of removal are few so that  $E_1$  is also not extremely small. This is done by a potential function argument that tracks both  $P(E_2|E_1)$  and  $P(W|E_1, E_2)$ , while the latter has higher weight than the former in the potential function. The potential function is non-decreasing, and increases by a non-negligible amount every time we exclude the losing part by fixing, thus guaranteeing the above-mentioned requirements as the probabilities cannot exceed 1.

We remark that proving a similar bound with only  $P(W|E_1, E_2)$  being  $n^{-O(c)}$ , and  $P(E_1, E_2)$  being  $2^{-n^{O(c)}}$  is not too hard. However for the latter part of our proof, we need that  $P(E_2|E_1)$  is also at least  $n^{-O(c)}$ . Hence, when removing coordinates, we fix the inputs and outputs in a very delicate manner, and analyze the evolution of potential function accordingly.

### Establish independence of Alice's and Bob's answers (Section 5)

Now that in each coordinate, Alice and Bob rarely both simultaneously output 0, we would like to strengthen the claim so that in each coordinate either Alice or Bob answers 0 with negligible probability. In other words, in each coordinate their answers are close to being independent. For a fixed coordinate, we consider Alice's output as a boolean function of her input, and the average of her output given Bob's input is exactly the sum of Fourier coefficients in the subcube where Bob receives 1. If we take average over any large event for Bob, then every Fourier coefficient, except the first one, will contribute negligibly to the result, meaning Bob answering 0 is close to being independent of Alice's answer.

This is not true, of course, unless Bob receives 1 with small enough probability. Fortunately the first step does not depend on the query distribution, and therefore we can change the query distribution at the very beginning, from uniform to the one where  $(0, 1, 0)$  has probability close to (but still polynomially larger than)  $1/n$ . It turns out that the change of distribution does not affect the parallel repetition property. With the right distribution, we bound the contributions of the Fourier coefficients as claimed above using Level- $k$  inequalities.

### Bound winning probability for Charlie (Section 6)

The previous steps indicate that Alice and Bob each owns a fixed set of coordinates where they output 0 with non-negligible probabilities, and the two sets are disjoint. Now consider an input  $(x, y, z)$ . Among the copies of games where Charlie needs to answer (Charlie receives 0), let  $G_1$  (and  $G_2$ ) be the copies where Charlie's answer points at a coordinate that Alice (and Bob) owns. On the other hand, in each coordinate they do not own, Alice and Bob output 0 with only negligible probability, so let  $B_1$  (and  $B_2$ ) be the copies where Alice's (and Bob's) answer string contains 0 outside the coordinates they own. Note that  $B_1$  depends only on  $x$ ,  $B_2$  depends only on  $y$ , while  $G_1$  and  $G_2$  depend only on  $z$ .

In order to win,  $G_1 \cup B_1$  have to cover all the copies that Alice plays with Charlie, which is the 1's in  $y$ , and  $G_2 \cup B_2$  have to cover all the copies that Bob plays with Charlie, which is the 1's in  $x$ . But for a typical input  $(x, y, z)$ , where both  $|x|$  and  $|z|$  are close to  $n/2$ ,  $G_1$  and  $B_1$  intersect with the 1's in  $y$  in proportion to their sizes. That means  $G_1$  has to cover almost all the copies that Charlie plays, and thus  $G_2 \cup B_2$  is not large enough to cover the 1's in  $x$ , as  $G_1$  and  $G_2$  are disjoint while  $B_1$  and  $B_2$  are negligibly small. This contradicts the fact that the winning probability is high, even conditioned on  $E_1 \times E_2$ .

## 2 Preliminaries

We use  $\log$  to denote the logarithm under base 2, with the convention that  $\log 0 = -\infty$ . Let  $\mathbb{N} = \{1, 2, \dots\}$  be the set of natural numbers. For every  $n \in \mathbb{N}$ , let  $[n]$  be the set  $\{1, 2, \dots, n\}$ .

For every  $x \in \{0, 1\}^n$ ,  $i \in [n]$  and  $S \subseteq [n]$ , we use  $x_i \in \{0, 1\}$  to denote the bit on index  $i$ , and  $x_S \in \{0, 1\}^{|S|}$  to denote the substring of  $x$  on  $S$ . Let  $\mathbf{1}(x) \subseteq [n]$  be the set of indices  $i$  where  $x_i = 1$ , and let  $|x| = |\mathbf{1}(x)|$  be the Hamming weight of  $x$ . We also define a partial order on  $\{0, 1\}^n$  such that  $x \geq y$  if and only if  $x_i = 1$  whenever  $y_i = 1$ .

For a random variable  $X$ , we use  $\text{supp}(X)$  to denote its support. We define a *fixing* of the random variable  $X$  to be an event that assigns  $X$  to be some fixed value in  $\text{supp}(X)$ . We equate every subset  $E \subseteq \text{supp}(X)$  to an event on  $X$ . We use  $P(E)$  to denote the probability of an event  $E$  under the distribution  $P$ .

► **Lemma 4** (Chernoff Bounds, see [25]). *Let  $X_1, \dots, X_n \in \{0, 1\}$  be independent random variables each with mean  $\mu$ , and let  $X = \sum_{i=1}^n X_i$ . Then, for all  $\delta \in (0, 1)$ , it holds that*

$$\Pr[X \leq (1 - \delta)\mu n] \leq e^{-\frac{\delta^2 \mu n}{2}}, \quad \Pr[X \geq (1 + \delta)\mu n] \leq e^{-\frac{\delta^2 \mu n}{3}}.$$

► **Lemma 5.** *Let  $P$  be a distribution and  $A, B$  be two events such that  $P(A \wedge B) > 0$ . Let  $X$  be a random variable with finite support, and let  $\mathcal{X} = \{x : P(X = x|B) > 0\}$ , and let  $x_0 \in \mathcal{X}$  be a fixed element such that  $P(X = x_0) \geq \delta$ .*

*For each  $x \in \mathcal{X}$ , we define  $\Phi(x) = \log P(A|B, X = x) + \frac{1}{2} \log P(B|X = x)$ , and let  $\Phi = \log P(A|B) + \frac{1}{2} \log P(B) < 0$ . Then, for every  $0 < \varepsilon < 1$ , it holds that either  $\Phi(x_0) \geq \Phi - \varepsilon$ , or  $P(X \in \mathcal{X} \wedge \Phi(X) \geq \Phi + \frac{1}{8}\delta\varepsilon) \geq 2^{2\Phi} \cdot \frac{1}{4}\delta^2\varepsilon$ .*

**Proof.** The proof is deferred to the appendix. ◀

### 2.1 Fourier Analysis

For every  $x, y \in \{0, 1\}^n$ , let  $x \cdot y$  be their inner product in  $\mathbb{Z}$ . Given a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ , let  $\hat{f} : \{0, 1\}^n \rightarrow \mathbb{R}$  be its Fourier coefficients, defined as

$$\hat{f}(u) = \frac{1}{2^n} \sum_{x \in \{0, 1\}^n} (-1)^{x \cdot u} f(x).$$



We will use the following equation on the sum of the Fourier coefficients in a subcube, which follows from Plancherel's theorem: For every  $y \in \{0, 1\}^n$ , we have

$$\sum_{u \leq y} \widehat{f}(u) = \frac{1}{2^{n-|y|}} \sum_{x \cdot y = 0} f(x).$$

We will also use the following version of the Level- $k$  inequality, proved in the appendix:

► **Lemma 6.** *Let  $f : \{0, 1\}^n \rightarrow \{-1, 0, 1\}$  be a function with  $\frac{1}{2^n} \sum_x |f(x)| = \alpha$ . Then for every  $k \in \mathbb{N}$ ,*

$$\sum_{|u|=k} |\widehat{f}(u)| \leq (2en \cdot \max\{1, \ln(1/\alpha)\})^{k/2} \cdot \alpha.$$

## 2.2 Multi-player Games

The notations we use here follows mostly from [17].

► **Definition 7** (Multiplayer Game). *A  $k$ -player game  $\mathcal{G}$  is a tuple  $\mathcal{G} = (\mathcal{X}, \mathcal{A}, Q, V)$ , where the question set  $\mathcal{X} = \mathcal{X}^1 \times \dots \times \mathcal{X}^k$ , and the answer set  $\mathcal{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^k$  are finite sets,  $Q$  is a probability distribution over  $\mathcal{X}$ , and  $V : \mathcal{X} \times \mathcal{A} \rightarrow \{0, 1\}$  is a predicate.*

► **Definition 8** (Game Value). *Let  $\mathcal{G} = (\mathcal{X}, \mathcal{A}, Q, V)$  be a  $k$ -player game. The value  $\text{val}(\mathcal{G})$  of the game  $\mathcal{G}$  is defined as*

$$\text{val}(\mathcal{G}) = \max_{f^1, \dots, f^k} \Pr_{X \sim Q} \left( V(X, (f^1(X^1), \dots, f^k(X^k))) = 1 \right),$$

where the maximum is over all sequence of functions  $(f^j : \mathcal{X}^j \rightarrow \mathcal{A}^j)_{j \in [k]}$ , which we call player strategies.

We note that the value of the game is unchanged even if we allow the player strategies to be randomized, that is, we allow the strategies to depend on some additional shared and private randomness.

► **Definition 9** (Parallel Repetition of a game). *Let  $\mathcal{G} = (\mathcal{X}, \mathcal{A}, Q, V)$  be a  $k$ -player game. We define its  $n$ -fold repetition as  $\mathcal{G}^{\otimes n} = (\mathcal{X}^{\otimes n}, \mathcal{A}^{\otimes n}, P, V^{\otimes n})$ . The sets  $\mathcal{X}^{\otimes n}$  and  $\mathcal{A}^{\otimes n}$  are defined to be the  $n$ -fold product of the sets  $\mathcal{X}$  and  $\mathcal{A}$  with themselves respectively. The distribution  $P$  is the  $n$ -fold product of the distribution  $Q$  with itself, that is,  $P(X = x) = \prod_{i=1}^n Q(X_i = x_i)$ . The predicate  $V^{\otimes n}$  is defined as  $V^{\otimes n}(x, a) = \bigwedge_{i=1}^n V(x_i, a_i)$ .*

In this paper we mostly deal with 3-player games, and we use the notation  $\mathcal{G} = (\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \mathcal{A} \times \mathcal{B} \times \mathcal{C}, Q, V)$ . That is, we use  $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$  in places of  $\mathcal{X}^1, \mathcal{X}^2, \mathcal{X}^3$  and use  $\mathcal{A}, \mathcal{B}, \mathcal{C}$  in places of  $\mathcal{A}^1, \mathcal{A}^2, \mathcal{A}^3$ . We also refer to the three players as Alice, Bob and Charlie.

The proof of the following useful lemma is essentially the same as Lemma 3.14 in [17], and is deferred to the appendix.

► **Lemma 10.** *Let  $\mathcal{G}_1 = (\mathcal{X}, \mathcal{A}, Q_1, V)$  and  $\mathcal{G}_2 = (\mathcal{X}, \mathcal{A}, Q_2, V)$  be two multi-player games where only the distributions are different. Let  $\lambda \in [0, 1]$  be such that for every  $x \in \mathcal{X}$ ,  $Q_1(X = x) \geq \lambda Q_2(X = x)$ . Then for every  $n \in \mathbb{N}$ , it holds that*

$$\text{val}(\mathcal{G}_1^{\otimes n}) \leq e^{-\lambda n/8} + \text{val}(\mathcal{G}_2^{\otimes \lfloor \lambda n/2 \rfloor}).$$

### 3 Main Results

► **Definition 11.** Let  $U$  be the uniform distribution over  $\mathcal{S} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ . For every  $k \in \mathbb{N}$  and every distribution  $Q$  over  $\mathcal{S}$ , we define a 3-player game  $\mathcal{G}_k(Q) = (\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \mathcal{A}_k \times \mathcal{B}_k \times \mathcal{C}_k, Q, V_k)$  with  $\mathcal{X} = \mathcal{Y} = \mathcal{Z} = \{0, 1\}$  as follows:

- (a)  $\mathcal{A}_k = \mathcal{B}_k = \{0, 1\}^k$  and  $\mathcal{C}_k = [k]$ .
- (b) For all  $(x, y, z) \in \mathcal{S}$  and  $(a, b, c) \in \mathcal{A}_k \times \mathcal{B}_k \times \mathcal{C}_k$ ,

$$V_k((x, y, z), (a, b, c)) = \begin{cases} b_c = 0, & \text{if } (x, y, z) = (1, 0, 0) \\ a_c = 0, & \text{if } (x, y, z) = (0, 1, 0) \\ \forall i \in [k], a_i + b_i \geq 1, & \text{if } (x, y, z) = (0, 0, 1) \end{cases}$$

► **Theorem 12.** For every  $k \in \mathbb{N}$ , there exists  $N_k \in \mathbb{N}$  such that for every  $n \in \mathbb{N}, n \geq N_k$ , it holds that  $\text{val}(\mathcal{G}_k(U)^{\otimes n}) \leq n^{-1/2000}$ .

Based on the results in [17, Section 8.2] and our discussions in the introduction, Theorem 12 implies the following bound on the parallel repetitions of 3-player games with binary inputs:

► **Theorem 13.** Let  $\mathcal{G} = (\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \mathcal{A} \times \mathcal{B} \times \mathcal{C}, Q, V)$  be any 3-player game with  $\mathcal{X} = \mathcal{Y} = \mathcal{Z} = \{0, 1\}$ , and such that  $\text{val}(\mathcal{G}) < 1$ . Then there exists a constant  $c = c(\mathcal{G}) > 0$  such that for every  $n \in \mathbb{N}$ , it holds that  $\text{val}(\mathcal{G}^{\otimes n}) \leq n^{-c}$ .

The rest of our paper is devoted to proving Theorem 12.

#### 3.1 Change the distribution

In order to prove Theorem 12, from now on we assume  $\text{val}(\mathcal{G}_k(U)^{\otimes n_1}) \geq n_1^{-1/2000}$  for some large enough  $n_1 \in \mathbb{N}$ , and eventually derive a contradiction. The first thing to do is changing the distribution so that Bob gets input 1 with small probability.

► **Definition 14.** Let  $n = \lfloor n_1/3 \rfloor$  and  $c = 1/1000$ . Let  $Q$  be the distribution over  $\mathcal{S}$  such that  $(0, 1, 0)$  has probability  $n^{-1+100c}$ , while  $(1, 0, 0)$  and  $(0, 0, 1)$  both have probability  $\frac{1}{2} - \frac{1}{2}n^{-1+100c}$  each.

▷ **Claim 15.**  $\text{val}(\mathcal{G}_k(Q)^{\otimes n}) \geq n^{-c}$ .

*Proof.* Let  $\lambda = 2/3$ , and thus we have  $1/3 \geq \lambda Q((X, Y, Z) = (x, y, z))$  for all  $(x, y, z) \in \mathcal{S}$ . Applying Lemma 10 on  $\mathcal{G}_k(U)$  and  $\mathcal{G}_k(Q)$  gives

$$\begin{aligned} \text{val}(\mathcal{G}_k(Q)^{\otimes \lfloor n_1/3 \rfloor}) &\geq \text{val}(\mathcal{G}_k(U)^{\otimes n_1}) - e^{-\lambda n_1/8} \\ &\geq n_1^{-c/2} - e^{-n/4} \geq n^{-c}. \end{aligned} \quad \triangleleft$$

Let  $P$  be the distribution  $Q^{\otimes n}$ , and let  $(X, Y, Z) \in \mathcal{S}^n$  be the random variables that represent the inputs to the three players under distribution  $P$ . Let  $f, g : \{0, 1\}^n \rightarrow \{0, 1\}^{n \times k}$  and  $h : \{0, 1\}^n \rightarrow [k]^n$  be strategies that achieve the value  $\text{val}(\mathcal{G}_k(Q)^{\otimes n})$ , and let  $W$  be the event that  $(f, g, h)$  wins on the inputs  $(X, Y, Z)$ , so that we have  $P(W) \geq n^{-c}$ .

#### 4 Remove Coordinates with (0, 0) Answers

► **Lemma 16.** *There exist  $S \subseteq [n]$ , a fixing  $F$  of  $(X_S, Y_S, Z_S)$ , and two events  $E_1 \subseteq \mathcal{X}^{\otimes n}, E_2 \subseteq \mathcal{Y}^{\otimes n}$  for Alice and Bob respectively, such that the following holds:*

- (a)  $|S| \leq n^{28c}$  and  $P(E_1|F) \geq e^{-n^{30c}}$ .
- (b)  $P(E_2|E_1, F) \geq n^{-2c}$  and  $P(W|E_1, E_2, F) \geq n^{-c}$ .
- (c) For every  $i \notin S$  and  $j \in [k]$ , it holds that

$$P((X_i, Y_i, Z_i) = (0, 0, 1) \wedge f_{i,j}(X) = 0 \wedge g_{i,j}(Y) = 0 | E_1, E_2, F) \leq n^{-7c}.$$

**Proof.** Initially let  $S = \emptyset$  and  $E_1 = \mathcal{X}^{\otimes n}, E_2 = \mathcal{Y}^{\otimes n}$ . We iterate the process described below to update  $S, F, E_1$  and  $E_2$  until requirement (c) is met. During the process, we examine the potential function

$$\begin{aligned} \Phi(E_1, E_2, F) &= \log P(W|E_1, E_2, F) + \frac{1}{2} \log P(E_2|E_1, F) \\ &= \log P(W, E_2|E_1, F) - \frac{1}{2} \log P(E_2|E_1, F), \end{aligned}$$

and ensure that the potential function  $\Phi(E_1, E_2, F)$  strictly increases for each iteration. Notice that initially we have

$$\Phi(E_1, E_2, F) = \log P(W) \geq -c \log n.$$

And as long as  $\Phi(E_1, E_2, F) \geq -c \log n$ , requirement (b) is always satisfied.

1. Let  $i \notin S, j \in [k]$  be a coordinate such that requirement (c) is violated, that is

$$P((X_i, Y_i, Z_i) = (0, 0, 1) \wedge f_{i,j}(X) = 0 \wedge g_{i,j}(Y) = 0 | E_1, E_2, F) > n^{-7c},$$

which, with the help of requirement (b), implies that

$$P((X_i, Y_i, Z_i) = (0, 0, 1) | E_1, F) > n^{-9c}, \tag{1}$$

$$P(f_{i,j}(X) = 0 | E_1, F, (X_i, Y_i, Z_i) = (0, 0, 1)) > n^{-9c}, \tag{2}$$

$$P(g_{i,j}(Y) = 0 | E_1, E_2, F, (X_i, Y_i, Z_i) = (0, 0, 1), f_{i,j}(X) = 0) > n^{-7c}. \tag{3}$$

Add  $i$  to the set  $S$ . The process stops if no such coordinate  $(i, j)$  exists.

2. Apply Lemma 5 on  $(X_i, Y_i, Z_i)$  over the distribution  $P$  conditioned on  $E_1 \wedge F$ , with  $\varepsilon = n^{-18c}$  and  $\delta = n^{-9c}$ . Since  $P((X_i, Y_i, Z_i) = (0, 0, 1) | E_1, E_2, F) > 0$ , by (1) we have either

$$\Phi(E_1, E_2, F \wedge (X_i, Y_i, Z_i) = (0, 0, 1)) \geq \Phi(E_1, E_2, F) - n^{-18c},$$

in which case we update  $F$  to  $F \wedge (X_i, Y_i, Z_i) = (0, 0, 1)$  and proceed to step 3; Or there exists  $(x, y, z) \in \{(1, 0, 0), (0, 1, 0)\}$  such that  $P((X_i, Y_i, Z_i) = (x, y, z) | E_1, E_2, F) > 0$ , and

$$\Phi(E_1, E_2, F \wedge (X_i, Y_i, Z_i) = (x, y, z)) \geq \Phi(E_1, E_2, F) + \frac{1}{8} n^{-27c},$$

$$P((X_i, Y_i, Z_i) = (x, y, z) | E_1, F) \geq 2^{2\Phi(E_1, E_2, F)} \cdot \frac{1}{8} n^{-36c},$$

in which case we update  $F$  to  $F \wedge (X_i, Y_i, Z_i) = (x, y, z)$  and iterate back from step 1.

3. Apply Lemma 5 on  $f_{i,j}(X)$  over the distribution  $P$  conditioned on  $E_1 \wedge F$ , with  $\varepsilon = n^{-8c}$  and  $\delta = n^{-9c}$ . Since  $P(f_{i,j}(X) = 0|E_1, E_2, F) > 0$ , by (2) we have either

$$\Phi(E_1 \wedge f_{i,j}(X) = 0, E_2, F) \geq \Phi(E_1, E_2, F) - n^{-8c},$$

in which case we update  $E_1$  to  $E_1 \wedge f_{i,j}(X) = 0$  and proceed to step 4; Or we have  $P(f_{i,j}(X) = 1|E_1, E_2, F) > 0$ , and

$$\Phi(E_1 \wedge f_{i,j}(X) = 1, E_2, F) \geq \Phi(E_1, E_2, F) + \frac{1}{8}n^{-17c},$$

$$P(f_{i,j}(X) = 1|E_1, F) \geq 2^{2\Phi(E_1, E_2, F)} \cdot \frac{1}{4}n^{-26c},$$

in which case we update  $E_1$  to  $E_1 \wedge f_{i,j}(X) = 1$  and iterate back from step 1.

4. Update  $E_2$  to  $E_2 \wedge g_{i,j}(Y) = 1$  and iterate back from step 1. Now that  $F$  implies  $(X_i, Y_i, Z_i) = (0, 0, 1)$  and  $E_1$  implies  $f_{i,j}(X) = 0$ , by the definition of the game (Definition 11) we know that  $W$  implies  $g_{i,j}(Y) = 1$ . Therefore, by (3), the increment of potential function in this step is

$$\begin{aligned} & \Phi(E_1, E_2 \wedge g_{i,j}(Y) = 1, F) - \Phi(E_1, E_2, F) \\ &= \frac{1}{2} \log P(E_2|E_1, F) - \frac{1}{2} \log P(E_2 \wedge g_{i,j}(Y) = 1|E_1, F) \\ &= -\frac{1}{2} \log P(g_{i,j}(Y) = 1|E_1, E_2, F) \\ &\geq \frac{1}{2} P(g_{i,j}(Y) = 0|E_1, E_2, F) \\ &\geq \frac{1}{2} n^{-7c}. \end{aligned}$$

Depending on the choices, in each iteration the potential function increases by at least either  $\frac{1}{8}n^{-27c}$ , or  $\frac{1}{8}n^{-17c} - n^{-18c}$ , or  $\frac{1}{2}n^{-7c} - n^{-8c} - n^{-18c}$ , which are all lower bounded by  $\frac{1}{8}n^{-27c}$ . This means that the potential function is indeed strictly increasing in each iteration, and thus requirement (b) is met. Since it always holds  $\Phi(E_1, E_2, F) \leq 0$ , this also means that the process will eventually stop, and the total number of iterations is at most  $8n^{27c} \cdot c \log n$ . In other words,  $|S| \leq 8n^{27c} \cdot c \log n \leq n^{28c}$ .

Finally, in order to bound  $P(E_1|F)$ , we prove in below that  $P(E_1|F)$  gets multiplied by at least a factor of  $n^{-70c}$  in each iteration. Since initially  $P(E_1|F) = 1$ , this implies that eventually after at most  $n^{28c}$  iterations, we have  $P(E_1|F) \geq (n^{-70c})^{n^{28c}} \geq e^{-n^{30c}}$ . In each iteration, when  $F$  gets updated to  $F \wedge (X_i, Y_i, Z_i) = (x, y, z)$  for some  $(x, y, z) \in \mathcal{S}$ ,  $P(E_1|F)$  changes by a factor of

$$\begin{aligned} \frac{P(E_1|F, (X_i, Y_i, Z_i) = (x, y, z))}{P(E_1|F)} &\geq P((X_i, Y_i, Z_i) = (x, y, z)|E_1, F) \\ &\geq \min \left\{ n^{-9c}, 2^{2\Phi(E_1, E_2, F)} \cdot \frac{1}{8}n^{-36c} \right\} \geq \frac{1}{8}n^{-38c}. \end{aligned}$$

The last line is because  $\Phi(E_1, E_2, F) \geq -c \log n$ . Furthermore, if step 3 is executed and  $E_1$  gets updated to  $E_1 \wedge f_{i,j}(X) = b$  for some  $b \in \{0, 1\}$ ,  $P(E_1|F)$  further changes by a factor of

$$\begin{aligned} \frac{P(E_1 \wedge f_{i,j}(X) = b|F)}{P(E_1|F)} &= P(f_{i,j}(X) = b|E_1, F) \\ &\geq \min \left\{ n^{-9c}, 2^{2\Phi(E_1, E_2, F)} \cdot \frac{1}{4}n^{-26c} \right\} \geq \frac{1}{8}n^{-30c}. \end{aligned}$$

The last line is because at step 3,  $\Phi(E_1, E_2, F) \geq -c \log n - n^{-18c} \geq -2c \log n$ . Note that in step 4 only  $E_2$  changes and  $P(E_1|F)$  does not change. So overall,  $P(E_1|F)$  changes by a factor of at least  $\frac{1}{8}n^{-38c} \cdot \frac{1}{8}n^{-30c} \geq n^{-70c}$ . ◀

Notice that the fixing  $F$  is independent of the remaining inputs in  $[n] \setminus S$ . For the rest of the paper, we change  $W$  to the event that  $(f, g, h)$  wins the copies of  $\mathcal{G}_k(Q)$  in  $[n] \setminus S$ , and change  $E_1, E_2, f, g, h$  to their relevant restrictions to the copies in  $[n] \setminus S$ , under the fixing  $F$ . Since  $|S| \leq n^{28c} = o(n)$ , by also changing  $c$  from  $\frac{1}{1000}$  to  $\frac{1}{1000} \cdot \frac{\log n}{\log(n-|S|)} < \frac{1}{999}$ , we can safely assume that  $S = \emptyset$  and remove  $F$  from the probability conditions, while the distribution  $Q$  remains the same and Lemma 16 still holds. This significantly simplifies the discussions later on.

## 5 Almost Independence of Answers in each Coordinate

Let  $E_1, E_2$  be specified as in the previous section. In this section, we prove the following lemma:

► **Lemma 17.** *For every  $i \in [n]$  and  $j \in [k]$ , at least one of the following holds:*

$$P(X_i = 0 \wedge f_{i,j}(X) = 0 | E_1, E_2) \leq n^{-3c}, \text{ or } P(Y_i = 0 \wedge g_{i,j}(Y) = 0 | E_1, E_2) \leq n^{-3c}.$$

We prove the above lemma using Fourier analysis. Fix some  $i \in [n]$  and  $j \in [k]$ . Define  $a : \{0, 1\}^n \rightarrow \{-1, 0, 1\}$  over the inputs of Alice as follows: For every  $x \in \{0, 1\}^n$ ,

$$a(x) = \begin{cases} 0 & \text{if } x \notin E_1, \\ -1 & \text{if } x \in E_1 \text{ and } x_i = 0 \text{ and } f_{i,j}(x) = 0, \\ 1 & \text{otherwise,} \end{cases}$$

and let  $b(x) = |a(x)|$ . Let  $\alpha = \frac{1}{2^n} \sum_x b(x) = \widehat{b}(0^n)$ . In the appendix we show the following lower bound on  $\alpha$ , due to the specific distribution  $Q$  in Definition 14:

► **Proposition 18.**  $\alpha \geq e^{-n^{130c}}$ .

► **Lemma 19.** *For every event  $E \subseteq \mathcal{Y}^{\otimes n}$  on  $Y$  with  $P(E) > 0$ , we have*

$$|\mathbf{E}[a(X)|E] - \widehat{a}(0^n)| \leq \frac{1}{P(E)} \cdot n^{-1/3} \alpha.$$

**Proof.** Since  $P(X_i = 1 | Y_i = 0) = 1/2$ , we have

$$\begin{aligned} \mathbf{E}[a(X)|E] &= \sum_{y \in E} \mathbf{E}[a(X)|Y = y] \cdot P(Y = y|E) = \sum_{y \in E} \frac{1}{2^{n-|y|}} \sum_{x: y=0} a(x) \cdot P(Y = y|E) \\ &= \sum_{y \in E} \sum_{u \leq y} \widehat{a}(u) \cdot P(Y = y|E) = \sum_{u \in \{0,1\}^n} \widehat{a}(u) \cdot P(Y \geq u|E). \end{aligned}$$

Using Lemma 6 on  $a$ , with the fact that  $\ln(1/\alpha) \leq n^{130c}$ , we get

$$\begin{aligned} |\mathbf{E}[a(X)|E] - \widehat{a}(0^n)| &\leq \sum_{u \neq 0^n} |\widehat{a}(u)| \cdot P(Y \geq u|E) \\ &\leq \frac{1}{P(E)} \sum_{u \neq 0^n} |\widehat{a}(u)| \cdot P(Y \geq u) \\ &\leq \frac{1}{P(E)} \sum_{\ell=1}^n (2en \cdot \max\{1, \ln(1/\alpha)\})^{\ell/2} \cdot \alpha \cdot (n^{-1+100c})^\ell \\ &\leq \frac{1}{P(E)} \cdot n^{-1/3} \alpha. \end{aligned}$$

With the exact same proof on  $b$ , we can also get

► **Lemma 20.** *For every event  $E \subseteq \mathcal{Y}^{\otimes n}$  on  $Y$  with  $P(E) > 0$ , we have*

$$|P(E_1|E) - \alpha| = |\mathbf{E}[b(X)|E] - \widehat{b}(0^n)| \leq \frac{1}{P(E)} \cdot n^{-1/3} \alpha.$$

In particular, when  $E = \mathcal{Y}^{\otimes n}$  we get  $P(E_1) \geq (1 - n^{-1/3})\alpha$ .

► **Corollary 21.** *For every event  $E \subseteq \mathcal{Y}^{\otimes n}$  on  $Y$  with  $P(E|E_1) > 0$ , we have*

$$\left| \mathbf{E}[a(X)|E_1, E] - \frac{\widehat{a}(0^n)}{\alpha} \right| \leq \frac{1}{P(E|E_1)} \cdot n^{-1/4}.$$

**Proof.** Since  $a(x) \neq 0$  only when  $x \in E_1$ , we have

$$\begin{aligned} \left| \mathbf{E}[a(X)|E_1, E] - \frac{\widehat{a}(0^n)}{\alpha} \right| &= \left| \frac{\mathbf{E}[a(X)|E]}{P(E_1|E)} - \frac{\widehat{a}(0^n)}{\alpha} \right| \\ &\leq \left| \frac{\mathbf{E}[a(X)|E]}{P(E_1|E)} - \frac{\widehat{a}(0^n)}{P(E_1|E)} \right| + \left| \frac{\widehat{a}(0^n)}{P(E_1|E)} - \frac{\widehat{a}(0^n)}{\alpha} \right| \\ &\leq \left| \frac{\mathbf{E}[a(X)|E]}{P(E_1|E)} - \frac{\widehat{a}(0^n)}{P(E_1|E)} \right| + \left| \frac{\alpha}{P(E_1|E)} - 1 \right| \quad (|\widehat{a}(0^n)| \leq \alpha) \\ &\leq \frac{2}{P(E_1 \wedge E)} \cdot n^{-1/3} \alpha \quad (\text{Lemmas 19 and 20}) \\ &= \frac{1}{P(E|E_1)} \cdot n^{-1/3} \cdot \frac{2\alpha}{P(E_1)} \\ &\leq \frac{1}{P(E|E_1)} \cdot n^{-1/4}. \quad (\text{Lemma 20}) \blacktriangleleft \end{aligned}$$

**Proof for Lemma 17.** Suppose that

$$P(Y_i = 0 \wedge g_{i,j}(Y) = 0 | E_1, E_2) > n^{-3c}.$$

Let  $E$  be the event  $E_2 \wedge Y_i = 0 \wedge g_{i,j}(Y) = 0$ . By argument (c) in Lemma 16, we have

$$P(X_i = 0 \wedge f_{i,j}(X) = 0 | E_1, E) \leq n^{-4c}.$$

Therefore  $\mathbf{E}[a(X)|E_1, E] \geq 1 - 2n^{-4c}$ . Since  $P(E_2|E_1) \geq n^{-2c}$  and  $P(E|E_1) = P(E|E_1, E_2) \cdot P(E_2|E_1) \geq n^{-5c}$ , by two applications of Corollary 21 (one on the event  $E$  and one on the event  $E_2$ ) we have

$$\begin{aligned} \mathbf{E}[a(X)|E_1, E_2] &\geq \mathbf{E}[a(X)|E_1, E] - \frac{1}{P(E_2|E_1)} \cdot n^{-1/4} - \frac{1}{P(E|E_1)} \cdot n^{-1/4} \\ &\geq 1 - 2n^{-4c} - (n^{2c} + n^{5c}) \cdot n^{-1/4} \\ &\geq 1 - 2n^{-3c}. \end{aligned}$$

This implies that  $P(X_i = 0 \wedge f_{i,j}(X) = 0 | E_1, E_2) \leq n^{-3c}$ . ◀

## 6 Independence Implies Low Winning Probability

For every  $i \in [n]$ , let

$$G_{1,i} = \left\{ j \in [k] \mid P(X_i = 0 \wedge f_{i,j}(X) = 0 | E_1, E_2) > n^{-3c} \right\},$$

$$G_{2,i} = \left\{ j \in [k] \mid P(Y_i = 0 \wedge g_{i,j}(Y) = 0 | E_1, E_2) > n^{-3c} \right\}.$$

Then Lemma 17 implies that  $G_{1,i} \cap G_{2,i} = \emptyset$ . For each  $x, y \in \{0, 1\}^n$ , let

$$B_1(x) = \{i \in [n] \mid x_i = 0 \wedge \exists j \notin G_{1,i}, f_{i,j}(x) = 0\},$$

$$B_2(y) = \{i \in [n] \mid y_i = 0 \wedge \exists j \notin G_{2,i}, g_{i,j}(y) = 0\}.$$

And for each  $z \in \{0, 1\}^n$ , let

$$G_1(z) = \{i \in [n] \mid z_i = 0 \wedge h_i(z) \in G_{1,i}\},$$

$$G_2(z) = \{i \in [n] \mid z_i = 0 \wedge h_i(z) \in G_{2,i}\}.$$

► **Lemma 22.** *Suppose  $(f, g, h)$  wins on the inputs  $(x, y, z)$ . Then at least one of the following holds:*

- (a)  $|x| \leq \frac{2}{5}n$  or  $|z| \leq \frac{2}{5}n$ ,
- (b)  $|B_1(x)| \geq n^{1-c}$  or  $|B_2(y)| \geq n^{1-c}$ ,
- (c)  $|B_1(x)| < n^{1-c}$  and  $|B_1(x) \cap \mathbf{1}(y)| \geq 4n^{-c} \cdot |y|$ ,
- (d)  $|G_1(z)| < \frac{1}{4}n - n^{1-c}$  and  $|G_1(z) \cap \mathbf{1}(y)| \geq (1 - 4n^{-c}) \cdot |y|$ .

**Proof.** Since  $G_{1,i} \cap G_{2,i} = \emptyset$  for every  $i$ , we know that  $G_1(z) \cap G_2(z) = \emptyset$  for every  $z$ . On the other hand, by the definition of the game (Definition 11), in order to win it must hold

$$\mathbf{1}(y) \subseteq G_1(z) \cup B_1(x) \quad (\text{since } x_{i, h_i(z)} = 0 \text{ when } x_i = z_i = 0)$$

$$\mathbf{1}(x) \subseteq G_2(z) \cup B_2(y) \quad (\text{since } y_{i, h_i(z)} = 0 \text{ when } y_i = z_i = 0)$$

Now suppose none of the items (a) to (d) holds. Since

$$|y| \leq |G_1(z) \cap \mathbf{1}(y)| + |B_1(x) \cap \mathbf{1}(y)|,$$

it implies that  $|G_1(z)| \geq \frac{1}{4}n - n^{1-c}$ . Therefore we have

$$|x| \leq |G_2(z)| + |B_2(y)| \leq n - |z| - |G_1(z)| + |B_2(y)| \leq \frac{7}{20}n + 2n^{1-c},$$

which contradicts the fact that  $|x| \geq \frac{2}{5}n$ . ◀

► **Proposition 23.**  $P(|X| - n/2| \geq n/10) \leq e^{-n/200}$ . The same holds when replacing  $X$  with  $Z$ .

**Proof.** This is a direct application of the Chernoff Bound (Lemma 4). ◀

Another careful application of the Chernoff Bound shows the following, and we defer the proof to the appendix:

► **Lemma 24.** *Let  $m \geq n^{1-c}$ , and  $M : \{0, 1\}^n \rightarrow 2^{[n]}$  satisfies  $M(x) \cap \mathbf{1}(x) = \emptyset$  for all  $x \in \{0, 1\}^n$ . Then we have*

$$P\left(|M(X)| < m \wedge |M(X) \cap \mathbf{1}(Y)| \geq \frac{4m}{n} \cdot |Y|\right) \leq e^{-n^{90c}}.$$

And the same holds when replacing  $X$  with  $Z$ .

Now we can bound the probability for each item in Lemma 22, conditioned on  $E_1 \wedge E_2$ . Recall that  $P(E_1 \wedge E_2) \geq e^{-n^{30c}} n^{-2c}$  by Lemma 16.

(a) By Proposition 23 we have

$$P(|X| \leq 2n/5 | E_1, E_2) \leq \frac{1}{P(E_1 \wedge E_2)} \cdot e^{-n/200} \leq e^{-n/300}.$$

Similarly we have  $P(|Z| \leq 2n/5 | E_1, E_2) \leq e^{-n/300}$ .

(b) For each  $i \in [n]$ , by the definitions of  $G_{1,i}, G_{2,i}$  and  $B_1(x), B_2(x)$ , using the union bound over  $j \in [k]$  we get

$$P(i \in B_1(X) | E_1, E_2) \leq kn^{-3c}, \quad P(i \in B_2(Y) | E_1, E_2) \leq kn^{-3c}.$$

Therefore we can bound the expectations of  $|B_1(X)|$  and  $|B_2(Y)|$ :

$$\mathbf{E}[|B_1(X)| | E_1, E_2] \leq kn^{1-3c}, \quad \mathbf{E}[|B_2(Y)| | E_1, E_2] \leq kn^{1-3c}.$$

Thus by Markov's inequality we have

$$P(|B_1(X)| \geq n^{1-c} | E_1, E_2) \leq kn^{-2c}, \quad P(|B_2(Y)| \geq n^{1-c} | E_1, E_2) \leq kn^{-2c}.$$

(c) Applying Lemma 24 on  $B_1(X)$  with  $m = n^{1-c}$ , we have

$$\begin{aligned} & P\left(|B_1(X)| < n^{1-c} \wedge |B_1(X) \cap \mathbf{1}(Y)| \geq 4n^{-c} \cdot |Y| \mid E_1, E_2\right) \\ & \leq \frac{1}{P(E_1 \wedge E_2)} \cdot e^{-n^{90c}} \leq e^{-n^{80c}}. \end{aligned}$$

(d) Same as (c), but applying Lemma 24 on  $G_1(Z)$  with  $m = \frac{1}{4}n - n^{1-c} \geq n^{1-c}$ , we get

$$P\left(|G_1(Z)| < \frac{1}{4}n - n^{1-c} \wedge |G_1(Z) \cap \mathbf{1}(Y)| \geq (1 - 4n^{-c}) \cdot |Y| \mid E_1, E_2\right) \leq e^{-n^{80c}}.$$

Putting everything together by a union bound, we get

$$P(W | E_1, E_2) \leq 2e^{-n/300} + 2kn^{-2c} + 2e^{-n^{80c}} < n^{-c},$$

as  $k$  is a constant and  $n$  is sufficiently large. This leads to a contradiction to the result (b) in Lemma 16, which refutes the assumption in Claim 15, and thus proves Theorem 12.

---

## References

- 1 Noga Alon and Bo'az Klartag. Economical toric spines via Cheeger's inequality. *J. Topol. Anal.*, 1(2):101–111, 2009.
- 2 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013. (also in STOC 2010).
- 3 Boaz Barak, Anup Rao, Ran Raz, Ricky Rosen, and Ronen Shaltiel. Strong parallel repetition theorem for free projection games. In *APPROX-RANDOM*, pages 352–365, 2009.
- 4 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. (also in FOCS 1995).
- 5 Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *STOC*, pages 113–131, 1988.
- 6 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. In *STOC*, pages 335–340, 2015.
- 7 Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *FOCS*, pages 746–755, 2013.
- 8 Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *CCC*, pages 236–249, 2004.



- 9 Irit Dinur, Prahladh Harsha, Rakesh Venkat, and Henry Yuen. Multiplayer parallel repetition for expanding games. In *ITCS*, volume 67 of *LIPICs*, pages Art. No. 37, 16, 2017.
- 10 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014.
- 11 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *J. ACM*, 45(4):634–652, 1998. (also in *STOC* 1996).
- 12 Uriel Feige, Guy Kindler, and Ryan O’Donnell. Understanding parallel repetition requires understanding foams. In *CCC*, pages 179–192, 2007.
- 13 Uriel Feige and Oleg Verbitsky. Error reduction by parallel repetition—a negative result. *Combinatorica*, 22(4):461–478, 2002.
- 14 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoret. Comput. Sci.*, 134(2):545–557, 1994.
- 15 H. Furstenberg and Y. Katznelson. A density version of the Hales-Jewett theorem. *J. Anal. Math.*, 57:64–119, 1991.
- 16 Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for the GHZ game: A simpler proof. In *APPROX-RANDOM*, pages 62:1–62:19, 2021.
- 17 Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for all 3-player games over binary alphabet. In *STOC*, 2022.
- 18 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. (also in *STOC* 1997).
- 19 Jan Hazla, Thomas Holenstein, and Anup Rao. Forbidden subgraph bounds for parallel repetition and the density hales-jewett theorem. *CoRR*, abs/1604.05757, 2016. [arXiv:1604.05757](#).
- 20 Thomas Holenstein. Parallel repetition: simplifications and the no-signaling case. *Theory Comput.*, 5:141–172, 2009. (also in *STOC* 2007).
- 21 Justin Holmgren and Ran Raz. A parallel repetition theorem for the GHZ game. *CoRR*, abs/2008.05059, 2020. [arXiv:2008.05059](#).
- 22 Justin Holmgren and Lisa Yang. The parallel repetition of non-signaling games: counterexamples and dichotomy. In *STOC*, pages 185–192, 2019.
- 23 Guy Kindler, Ryan O’Donnell, Anup Rao, and Avi Wigderson. Spherical cubes and rounding in high dimensions. In *FOCS*, pages 189–198, 2008.
- 24 Kunal Mittal and Ran Raz. Block rigidity: strong multiplayer parallel repetition implies super-linear lower bounds for Turing machines. In *ITCS*, pages Art. No. 71, 15, 2021.
- 25 Michael Mitzenmacher and Eli Upfal. *Probability and computing*. Cambridge University Press, Cambridge, 2005. Randomized algorithms and probabilistic analysis.
- 26 Ryan O’Donnell. *Analysis of Boolean functions*. Cambridge University Press, New York, 2014.
- 27 Itzhak Parnafes, Ran Raz, and Avi Wigderson. Direct product results and the GCD problem, in old and new communication models. In *STOC*, pages 363–372, 1997.
- 28 D. H. J. Polymath. A new proof of the density Hales-Jewett theorem. *Ann. of Math. (2)*, 175(3):1283–1327, 2012.
- 29 Anup Rao. Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011. (also in *STOC* 2008).
- 30 Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. (also in *STOC* 1995).
- 31 Ran Raz. Parallel repetition of two prover games. In *CCC*, pages 3–6, 2010.
- 32 Ran Raz and Ricky Rosen. A strong parallel repetition theorem for projection games on expanders. In *CCC*, pages 247–257, 2012.
- 33 Oleg Verbitsky. Towards the parallel repetition conjecture. *Theoret. Comput. Sci.*, 157(2):277–282, 1996.

**A** Deferred Proofs**Proof for Lemma 5**

**Proof.** By Jensen's inequality, we have

$$\begin{aligned} P(A|B)^2 \cdot P(B) &= \left( \sum_{x \in \mathcal{X}} P(X = x|B) \cdot P(A|B, X = x) \right)^2 \cdot P(B) \\ &\leq \sum_{x \in \mathcal{X}} P(X = x|B) \cdot P(A|B, X = x)^2 \cdot P(B) \\ &= \sum_{x \in \mathcal{X}} P(X = x) \cdot P(A|B, X = x)^2 \cdot P(B|X = x). \end{aligned}$$

Suppose that  $\Phi(x_0) < \Phi - \varepsilon$ , which implies that

$$\begin{aligned} P(A|B, X = x_0)^2 \cdot P(B|X = x_0) &< P(A|B)^2 \cdot P(B) \cdot 2^{-2\varepsilon} \\ &\leq P(A|B)^2 \cdot P(B) \cdot (1 - \varepsilon/4). \end{aligned}$$

On the other hand, since  $\delta, \varepsilon \leq 1$  we have  $\log(1 + \delta\varepsilon/4) \geq \delta\varepsilon/4$ , and thus in order to satisfy  $\Phi(x) \geq \Phi + \frac{1}{8}\delta\varepsilon$  it suffices to have

$$P(A|B, X = x)^2 \cdot P(B|X = x) \geq P(A|B)^2 \cdot P(B) \cdot (1 + \delta\varepsilon/4). \quad (4)$$

Let  $\mathcal{X}_1 \subset \mathcal{X}$  be the set of  $x \in \mathcal{X}$ ,  $x \neq x_0$  that satisfies (4). Since  $P(A|B, X = x)^2 \cdot P(B|X = x) \leq 1$ , we have

$$\begin{aligned} &P\left(X \in \mathcal{X} \wedge \Phi(X) \geq \Phi + \frac{1}{8}\delta\varepsilon\right) \\ &\geq \sum_{x \in \mathcal{X}_1} P(X = x) \cdot P(A|B, X = x)^2 \cdot P(B|X = x) \\ &= \sum_{x \in \mathcal{X}} P(X = x) \cdot P(A|B, X = x)^2 \cdot P(B|X = x) \\ &\quad - P(X = x_0) \cdot P(A|B, X = x_0)^2 \cdot P(B|X = x_0) \\ &\quad - \sum_{x \notin \mathcal{X}_1, x \neq x_0} P(X = x) \cdot P(A|B, X = x)^2 \cdot P(B|X = x) \\ &\geq P(A|B)^2 \cdot P(B) [1 - P(X = x_0) \cdot (1 - \varepsilon/4) - P(X \neq x_0) \cdot (1 + \delta\varepsilon/4)] \\ &\geq P(A|B)^2 \cdot P(B) \cdot \frac{1}{4}\delta^2\varepsilon. \end{aligned} \quad \blacktriangleleft$$

**Proof for Lemma 6**

**Proof.** Since there are at most  $n^k$  many  $u$  with  $|u| = k$ , we have

$$\sum_{|u|=k} |\hat{f}(u)| \leq n^{k/2} \sqrt{\sum_{|u|=k} \hat{f}(u)^2}.$$

Therefore it suffices to prove that

$$\sum_{|u|=k} \hat{f}(u)^2 \leq (2e \cdot \max\{1, \ln(1/\alpha)\})^k \cdot \alpha^2.$$

When  $k \leq 2 \ln(1/\alpha)$ , it follows from the original Level- $k$  inequality (see [26, Section 9.5]). When  $k > 2 \ln(1/\alpha)$ , we also have

$$\sum_{u \in \{0,1\}^n} \widehat{f}(u)^2 = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)^2 = \alpha \leq e^k \alpha^2 \leq (2e \cdot \max\{1, \ln(1/\alpha)\})^k \cdot \alpha^2. \quad \blacktriangleleft$$

### Proof for Lemma 10

**Proof.** Notice that we can write  $Q_1 = \lambda Q_2 + (1 - \lambda)Q'$  for some distribution  $Q'$  over  $\mathcal{X}$ . Let  $Z = (Z_1, \dots, Z_n) \in \{0,1\}^n$  be i.i.d. Bernoulli random variables such that for each  $i \in [n]$ , independently,  $Z_i$  is 1 with probability  $\lambda$  and 0 with probability  $1 - \lambda$ . For each  $i \in [n]$ , we think of the  $i$ -th copy of  $Q_1$  as depending on  $Z_i$ : if  $Z_i = 1$  then  $Q_1$  is drawn from  $Q_2$ , otherwise  $Q_1$  is drawn from  $Q'$ .

In order to bound the value of the game  $\mathcal{G}_1^{\otimes n}$ , we can assume that each of the players is also given  $Z$  as input, since this can only increase the game's value. Observe that conditioned on the event  $Z = z$  for any fixed value  $z \in \{0,1\}^n$ , the value of the game is at most the value of  $\mathcal{G}_2^{\otimes |z|}$ . Thus we have

$$\begin{aligned} \text{val}(\mathcal{G}^{\otimes n}) &\leq \sum_{m=0}^n \Pr[|Z| = m] \cdot \text{val}(\mathcal{G}_2^{\otimes m}) \\ &\leq \Pr\left[|Z| \leq \frac{\lambda n}{2}\right] \cdot 1 + 1 \cdot \text{val}(\mathcal{G}_2^{\otimes \lfloor \lambda n/2 \rfloor}) \\ &\leq e^{-\lambda n/8} + \text{val}(\mathcal{G}_2^{\otimes \lfloor \lambda n/2 \rfloor}). \end{aligned} \quad \blacktriangleleft$$

### Proof for Proposition 18

**Proof.** Recalling the distribution  $Q$  in Definition 14, we have

$$\begin{aligned} P(E_1) &= \sum_{x \in \{0,1\}^n} \left(\frac{1}{2} - \frac{1}{2}n^{-1+100c}\right)^{|x|} \left(\frac{1}{2} + \frac{1}{2}n^{-1+100c}\right)^{n-|x|} b(x) \\ &\leq (1 + n^{-1+100c})^n \cdot \frac{1}{2^n} \sum_{x \in \{0,1\}^n} b(x) \\ &\leq e^{n^{100c}} \alpha. \end{aligned}$$

Since  $P(E_1) \geq e^{-n^{30c}}$ , we get  $\alpha \geq e^{-n^{130c}}$ .  $\blacktriangleleft$

### Proof for Lemma 24

**Proof.** Fix an  $x \in \{0,1\}^n$  with  $|x| \leq \frac{3}{5}n$  and  $|M(x)| < m$ . By Proposition 23, this makes for a probability of  $P(|X| > \frac{3}{5}n) \leq e^{-n/200}$ .

Since  $p = P(Y_i = 1 | X_i = 0) > n^{-1+100c}$ , by applying Chernoff Bound on the sets  $[n] \setminus \mathbf{1}(x)$  and  $M(x)$  respectively, we have

$$\begin{aligned} P\left(|Y| \leq \frac{np}{3} \mid X = x\right) &\leq e^{-np/180} < e^{-n^{100c}/180}, \\ P\left(|M(x) \cap \mathbf{1}(Y)| \geq \frac{4mp}{3} \mid X = x\right) &\leq e^{-mp/27} < e^{-n^{99c}/27}. \end{aligned}$$

Therefore by union bound,

$$\begin{aligned} P\left(|M(X)| < m \wedge |M(X) \cap \mathbf{1}(Y)| \geq \frac{4m}{n} \cdot |Y|\right) &\leq e^{-n/200} + e^{-n^{100c}/180} + e^{-n^{99c}/27} \\ &\leq e^{-n^{90c}}. \end{aligned} \quad \blacktriangleleft$$



# Local Treewidth of Random and Noisy Graphs with Applications to Stopping Contagion in Networks

Hermish Mehta ✉

Citadel Securities, Chicago, IL, USA

Daniel Reichman ✉

Department of Computer Science, Worcester Polytechnic Institute, MA, USA

---

## Abstract

We study the notion of local treewidth in sparse random graphs: the maximum treewidth over all  $k$ -vertex subgraphs of an  $n$ -vertex graph. When  $k$  is not too large, we give nearly tight bounds for this local treewidth parameter; we also derive nearly tight bounds for the local treewidth of noisy trees, trees where every non-edge is added independently with small probability. We apply our upper bounds on the local treewidth to obtain fixed parameter tractable algorithms (on random graphs and noisy trees) for edge-removal problems centered around containing a contagious process evolving over a network. In these problems, our main parameter of study is  $k$ , the number of initially “infected” vertices in the network. For the random graph models we consider and a certain range of parameters the running time of our algorithms on  $n$ -vertex graphs is  $2^{o(k)} \text{poly}(n)$ , improving upon the  $2^{\Omega(k)} \text{poly}(n)$  performance of the best-known algorithms designed for worst-case instances of these edge deletion problems.

**2012 ACM Subject Classification** Theory of computation → Dynamic graph algorithms

**Keywords and phrases** Graph Algorithms, Random Graphs, Data Structures and Algorithms, Discrete Mathematics

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.7

**Category** RANDOM

**Related Version** Full Version: <https://arxiv.org/abs/2204.07827>

**Acknowledgements** We are very grateful to Michael Krivelevich who provided numerous valuable comments and links to relevant work. Josh Erde offered useful feedback. We would like to thank the anonymous referees for helpful comments and suggestions. In particular, we thank a reviewer for noting a gap in a claimed proof of a stronger lower bound of  $\Omega(k \log d / \log n)$  of the local treewidth of  $G(n, d/n)$ . An inspiration for this paper has been the operation of the Oncology department at Haddasah Ein Karem hospital during the Covid-19 pandemic. Their professionalism and dedication are greatly acknowledged.

## 1 Introduction

Treewidth is a graph-theoretic parameter that measures the resemblance of a graph to a tree. We begin by recalling the definition of treewidth.

► **Definition 1** (Tree Decomposition). *A tree decomposition of a graph  $G = (V, E)$  is a pair  $(T, X)$ , where  $X$  is a collection of subsets of  $V$ , called bags, and  $T$  a tree on vertices  $X$  satisfying the properties below:*

1. *The union of all sets  $X_i \in X$  is  $V$ .*
2. *For all edges  $(u, v) \in E$ , there exists some bag  $X_i$  which contains both  $u$  and  $v$ .*
3. *If both  $X_i$  and  $X_j$  contain some vertex  $u \in V$ , then all bags  $X_k$  on the unique path between  $X_i$  and  $X_j$  in  $T$  also contain  $u$ .*



© Hermish Mehta and Daniel Reichman;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 7; pp. 7:1–7:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

► **Definition 2** (Treewidth). *The width of a tree decomposition  $(T, X)$  is one less than cardinality of the largest bag. More formally, we can express this as*

$$\max_i |X_i| - 1.$$

*The treewidth of a graph  $G = (V, E)$  is the minimum width among all tree decompositions of  $G$ .*

Many graph-theoretic problems that are NP-hard admit polynomial-time algorithms on graph families whose treewidth is sufficiently slowly growing as a function of the number of vertices [32]. There is vast literature concerned with finding methods to relate the treewidth of graphs to other well-studied combinatorial parameters and leveraging this to devise efficient algorithms for algorithmic problems in graphs with constant or logarithmic treewidth. An excellent introduction to the concept of treewidth as well as brief survey of the work of Robertson and Seymour in establishing this concept can be found in Chapter 12 of [16].

These treewidth-based algorithmic methods, however, have historically found limited applicability in random graphs. Sparse random graphs  $G(n, d/n)$  where every edge occurs independently with probability  $d/n$ , for some  $d > 1$ , exhibit striking contrast between their local and global properties – and this contrast is apparent when looking at treewidth. Locally, these graphs appear tree-like with high probability<sup>1</sup> (w.h.p.): the ball of radius  $O(\log_d n)$  around every vertex looks like a tree plus a constant number of additional edges. Globally, however, these graphs have w.h.p. treewidth  $\Omega(n)$ . For example, the super-critical random graph  $G(n, \frac{1+\delta}{n})$  has w.h.p. treewidth  $\Omega(n)$  [17, 46, 38]. As a result of this global property, conventional techniques used to exploit low treewidth to derive efficient algorithms do not apply directly for random graphs.

In this paper, we take advantage of the local tree-like structure of random graphs by analyzing the *local* behavior of treewidth in random graphs. Central to our approach is the following definition.

► **Definition 3** (Local Treewidth). *Let  $G$  be an undirected  $n$ -vertex graph. Given  $k \leq n$  we denote by  $t_k(G)$  the largest treewidth of a subgraph of cardinality  $k$  of  $G$ .*

In words, the local treewidth of an  $n$ -vertex graph, with locality parameter  $k$ , is the maximum possible treewidth across all subgraphs of size  $k$ . We study two models of random graphs, starting with the familiar binomial random graph  $G(n, p)$ . While the binomial random graph  $G(n, p)$  lacks many of the characteristics of empirically observed networks such as skewed degree distributions, studying algorithmic problems on random graphs can nevertheless lead to interesting algorithms.

► **Definition 4** (Noisy Trees). *Let  $T$  be an  $n$  vertex tree. The noisy tree  $T'$  obtained from  $T$  is a random graph model where every non edge of  $T$  is added to  $T$  independently, with probability  $1/n$ .*

Here we assume  $p = 1/n$  for convenience; all our results regarding noisy trees also hold when the perturbation probability  $p$  satisfies  $p = \epsilon/n$  for  $\epsilon < 1$ . Noisy trees are related to small world models of random networks [45, 44], where adding a few random edges to a graph of high diameter such as a path results with a graph of logarithmic diameter w.h.p. [36].

---

<sup>1</sup> Given a random graph model, we say an event happens with high probability if it occurs with probability tending to 1 as  $n$  tends to infinity.

Below, we give an informal description of the concepts we study and sketch our main results; we defer discussion of formal results until Section 2 and later in the paper. Our main result is a nearly tight bound holding w.h.p. for the maximum treewidth of a  $k$ -vertex subgraph of  $G(n, p)$  assuming  $k \leq n^{1-\epsilon}$  for  $\epsilon \in (0, 1)$  and  $p = d/n$  where  $d > 1$ . In the notation introduced earlier, this provides a bound for  $t_k(G)$ . Assuming  $k \leq n^\epsilon$  for a sufficiently small  $\epsilon$  we obtain nearly tight bounds for the local treewidth of noisy trees as well.

Our upper bounds on the local treewidth are motivated by algorithmic problems related to containing the spread of a contagious process over undirected graphs by deleting edges. We focus on the bootstrap percolation contagious process (Definition 8) where there is a set of initially infected vertices and noninfected vertices are infected if they have at least  $r \geq 2$  neighbors and consider two edge-removal problems: *Stopping Contagion* and *Minimizing Contagion*. Informally, in stopping contagion we are given a subset of infected nodes  $A$  and seek to remove a minimal number of edges to ensure a “protected” subset of vertices  $B$  (disjoint from  $A$ ) are not infected from  $A$ . In minimizing contagion we wish to ensure at most  $m$  additional vertices are infected from  $A$  for a target value  $m$  by deleting a minimal number of edges. Such edge removal problems might arise, among other applications [20, 21], in railways and air routes, where the goal might be to prevent spread while also minimizing interference to transportation. In this context, edge deletion may correspond to removing a transportation link altogether or introducing special requirements (such as costly checks) to people between the the endpoints. Edge removal can be also viewed as a *social distancing* measure to control an epidemic outbreak [5]. One can also study the problem of removing *vertices* to control the spread of an epidemic which is related to vaccinations: making nodes immune to infection and removing them from the network [49].

We design algorithms for stopping and minimizing contagion for random graphs and noisy trees. Note that our algorithms do not achieve polynomial time, even for  $k$  that is polylogarithmic in  $n$ ; whether there exists a polynomial time algorithm for minimizing contagion and stopping contagion in  $G(n, p)$  for every value of  $k$  is an open question. Nonetheless, the dependency of our algorithm on  $k$  is better (assuming  $k \leq n^\epsilon$  for an appropriate constant  $\epsilon > 0$ ) than the dependency of  $k$  in the running time of the best known algorithms for minimizing contagion<sup>2</sup> in the worst case [14]. Please see Subsection 2.3 for details.

Our algorithms are based on the following three observations:

1. The local treewidth of binomial random graphs and noisy trees is sublinear in  $k$ .
2. There exist fast algorithms for minimizing and stopping contagion in graphs of bounded treewidth.
3. The set of seeds  $A$  has what we call the *bounded spread* property: w.h.p. at most  $c|A|$  additional vertices are infected from  $A$  for some constant<sup>3</sup>  $c$ . Bounded spread allows us to solve minimizing contagion and stopping contagion on subgraphs that have small (sublinear in  $k$ ) treewidth.

For the sake of brevity and readability we focus on *edge* deletion problems. We note that our algorithms can be easily adapted for the analogous problems of minimizing and stopping contagion by deleting *vertices* rather than edges. The reason is that our algorithms for minimizing/stopping contagion on bounded treewidth graphs work (with the same asymptotic running time guarantees) for vertex deletion problems. Combining algorithms for bounded treewidth with the bounded spread property as well the upper bound on the local treewidth yields algorithms for the vertex deletion versions of minimizing and stopping contagion.

<sup>2</sup> We are not aware of previous algorithms for the stopping contagion problem.

<sup>3</sup> For  $G(n, d/n)$ , our constant  $c := c(d)$  is a function of  $d$ . When  $d$  is a constant independent of  $n$  so is  $c$ .

Our main contribution is studying the concept of local treewidth for random graphs and connecting it to algorithmic problems involving stopping contagion in networks. Our calculations are standard and the contribution is conceptual rather than introducing a new technique.

## 2 Our results

### 2.1 Local Treewidth Bounds

Recall we define the local treewidth of a graph  $G$ , denoted  $t_k(G)$ , to be the greatest treewidth among all subgraphs of size  $k$ . Trivially, for any graph with at least one edge and  $k \leq n$ ,  $1 \leq t_k(G) \leq k$ .

Consider as an illustrative example the random graph  $G = G(n, 1/2)$ : with high probability,  $t_k(G) = \Omega(k)$  for all values of  $k$ . For  $k \leq 1.9 \log n$  this follows as there is a clique of size  $k$  in  $G$  w.h.p. For  $k > 1.9 \log n$  this follows as a randomly chosen subset of size  $k$  has, with high probability, minimum degree  $\Omega(k)$ , and a graph with treewidth  $r$  has a vertex of degree at most  $r$ .

We can now state our bounds for  $t_k$  in the random graph models we consider. From here onward,  $\epsilon > 0$  is taken to be a positive constant in  $(0, 1)$ . We give somewhat compressed statements; reference to the full Theorems are provided throughout this section.

► **Theorem 5.** *Let  $G = G(n, p)$  with  $p = d/n$  and  $k \leq n^{1-\epsilon}$ . Then, with high probability:*

$$t_k(G) \leq 3 + O\left(\frac{k \log d}{\log n}\right).$$

Since we always know  $t_k(G) \leq k$ , the upper bound in the Theorem above becomes trivial if  $d \geq n^{\Omega(1)}$ . Also observe that the Theorem does not hold for arbitrary  $k \leq n$ , as for  $k = n$ ,  $t_k(G) = \Omega(n)$  w.h.p. In terms of lower bounds, we have the following:

► **Theorem 6.** *Suppose  $p = d/n$  and  $d > 1 + \delta$  where  $\delta > 0$  is a constant (not depending on  $n$ ). Suppose  $k \leq O(n/\log n)$ ; then, w.h.p.*

$$t_k(G) \geq \Omega\left(\frac{k}{\log n}\right).$$

More details can be found in Section 3. Our upper and lower bounds for the local treewidth of  $G(n, d/n)$  also extend to the random  $d$ -regular graph  $G(n, d)$ —details can be found in Subsection 3.3.

For noisy trees, we have the following results.

► **Theorem 7.** *Let  $T$  be an  $n$ -vertex tree with maximum degree  $\Delta$ . Let  $T'$  be a noisy tree obtained from  $T$ . Then w.h.p.*

$$t_k(T') \leq 3 + O\left(\frac{k(\log k + \log \Delta)}{\log n}\right).$$

Observe that the upper bound in the Theorem is trivial if  $k, \Delta$  are  $n^{\Omega(1)}$ . As a result, in our proofs we will assume  $k, \Delta \leq n^\epsilon$ , for sufficiently small  $\epsilon > 0$ . Our results can be extended to the case where each non-edge is added with probability  $c/n$  for  $c > 1$ . Similar ideas (which are omitted) yield the upper bound:

$$t_k(T') \leq 3 + O\left(\frac{k(\log k + \log \Delta + \log c)}{\log n}\right).$$



We also provide a lower bound, showing that up to the  $\log k, \log \Delta$  terms, the upper bound above is tight. Namely, the noisy path has w.h.p. local treewidth of order  $\Omega(k/\log n)$ . For more details on the lower and upper bounds please see Section 4.

## 2.2 Contagious Process and Edge Deletion problems

The local treewidth results outlined above prove useful in the context of two edge deletion problems we study. These problems arise when considering the evolution of a contagious processes over an undirected graph.

We focus on the  $r$ -neighbor bootstrap percolation model [11].

► **Definition 8.** *In  $r$ -neighbor bootstrap percolation we are given an undirected graph  $G = (V, E)$  and an integer threshold  $r \geq 1$ . Every vertex is either active (we also use the term infected) or inactive; a set of vertices composed entirely of active vertices is called active. Initially, a set of vertices called seeds,  $A_0$ , is activated. A contagious process evolves in discrete steps, where for integral  $i > 0$ ,*

$$A_i = A_{i-1} \cup \{v \in V : |N(v) \cap A_{i-1}| \geq r\}.$$

*Here,  $N(v)$  is the set of neighbors of  $v$ . In words, a vertex becomes active in a given step if it has at least  $r$  active neighbors. An active vertex remains active throughout the process and cannot become inactive. Set*

$$\langle A_0 \rangle = \bigcup_i A_i.$$

*The set  $\langle A_0 \rangle$  is the set of nodes that eventually get infected from  $A_0$  in  $G$ . Clearly,  $\langle A_0 \rangle$  depends on the graph  $G$ , so we sometimes write  $\langle A_0 \rangle_G$  to call attention to the underlying graph. We say a vertex  $v \in V$  gets activated or infected from a set of seeds  $A_0$  if  $v \in \langle A_0 \rangle$ .*

It is straightforward to extend this definition to the case where every vertex  $v$  has its own threshold  $t(v)$  and a vertex is infected only if it has at least  $t(v)$  active neighbors at some point. As is customary in bootstrap percolation models, we usually assume that all thresholds are larger than 1. Now, given a network with an evolving contagious process, we introduce the stopping contagion problem:

► **Definition 9 (Stopping Contagion).** *In the stopping contagion problem, we are given as input a graph  $G = (V, E)$  along with two disjoint sets of vertices,  $A, B \subseteq V$ . Given that the seed set is  $A$ , the goal is to compute the minimum number of edge deletions necessary to ensure that no vertices from  $B$  are infected. In other words, we want to make sure  $\langle A \rangle_{G'} \cap B = \emptyset$ , where  $G'$  is the graph obtained from  $G$  after edge deletions. Given an additional target parameter,  $\ell$ , the corresponding decision problem asks whether it is possible to ensure no vertices from  $B$  are infected by deleting at most  $\ell$  edges.*

Next we consider the setting where given a set of infected nodes we want to remove the minimal number of edges to ensure no more than  $k$  additional vertices are infected.

► **Definition 10.** *In the minimizing contagion problem, we are given a graph  $G = (V, E)$ , a subset of vertices  $A \subseteq V$  and a parameter  $s$ . Given that the seed set is  $A$ , we want to compute the minimum number of edge deletions required to ensure at most  $s$  vertices in  $V \setminus A$  are infected. If  $G'$  is the graph obtained from  $G$  by edge deletions, then this condition is equivalent to requiring  $|\langle A \rangle_{G'}| \leq |A| + s$ . In the decision problem, we want to decide if it is possible to ensure  $|\langle A \rangle_{G'}| \leq |A| + s$  with at most  $\ell$  edge deletions.*

Both stopping contagion and minimizing contagion are NP-complete, and stopping contagion remains NP-hard even if  $|A| = 2$  and  $|B| = 1$ . For complete proofs, please refer to the Full version of this paper [39].

### 2.3 Algorithmic Results

For minimizing contagion, current algorithmic ideas [14] can be used to prove that if  $|A|$  and the optimal solution are of size  $O(k)$  the problem can be solved in time  $2^{O(k)} \text{poly}(n)$  on  $n$ -vertex graphs. No such algorithm, parameterized by  $|A|$  and the size of the optimal solution, is known for stopping contagion. Using our upper bounds for local treewidth, however, we can prove:

► **Theorem 11.** *Let  $\epsilon$  be a constant in  $(0, 1)$ . Suppose that  $k \leq n^{1-\epsilon}$  and that every vertex has threshold greater than 1. Let  $G := G(n, p)$  where  $p = d/n$ . Assuming  $d$  is a constant, we have that w.h.p. both minimizing contagion and stopping contagion can be solved in  $G$  in time  $2^{o(k)} \text{poly}(n)$ .*

► **Theorem 12.** *Suppose that  $k \leq n^\epsilon$  for sufficiently small  $\epsilon \in (0, 1)$  and that every vertex has a threshold greater than 1. Let  $T'$  be a noisy tree where the base tree  $T$  has maximum degree  $\Delta = O(1)$ . Then w.h.p. both minimizing contagion and stopping contagion can be solved in  $T'$  in time  $2^{o(k)} \text{poly}(n)$ .*

We stress that set  $A$  of seeds can be chosen in arbitrary way. In particular, an adversary can pick  $A$  after the random edges in our graph models have been chosen.

The dependence of the running time on  $n, k, d$  and  $\Delta$  can be made explicit: for precise statements, please see Section 6. Algorithms for grids and planar graphs are presented in Section 6 as well.

For our purpose, to translate local treewidth bounds to algorithmic results, we need an algorithm for solving stopping contagion and minimizing contagion on graphs of low treewidth. We provide such an algorithm that runs in exponential time in the treewidth, assuming the maximum degree is constant, using ideas from [14]. More details can be found in Section 5.

### 2.4 Our Techniques

Our upper bounds for the local treewidth build on a simple “edge excess principle”: A  $k$ -vertex connected graph with  $k + r$  edges has treewidth at most  $r + 1$ . As the treewidth of a set of connected components is the maximum treewidth of a component, it suffices to analyze the number of edges in connected subgraphs of the random graphs we study. For  $G(n, p)$  this is straightforward, but for noisy trees it is somewhat more involved. We find it easier to first analyze the edge excess of connected subgraphs, before considering connecting edges that allow us bound the excess of arbitrary subgraphs.

A key component in our lower bound is the simple fact that if  $H$  is a minor of  $G$  then  $\text{tw}(G) \geq \text{tw}(H)$ . Therefore it suffices to prove the existence of large treewidth subgraphs that are minors w.h.p. of random graphs and noisy trees. Recall that an  $n$ -vertex graph is called an  $\alpha$ -expander if there exists  $\alpha \in (0, 1)$  such that every subset  $S$  of vertices with at most  $n/2$  vertices has at least  $\alpha|S|$  neighbors not in  $S$ . We use the fact [35] that for any graph  $H$  with  $k$  vertices and edges, assuming  $k = O(n/\log n)$  an  $n$ -vertex expander has an embedding<sup>4</sup> of

<sup>4</sup> See Subsection 2.7 for further details on minor-theoretic concepts we use.

$H$  as a minor in  $G$ . Furthermore, every connected subgraph of  $G$  corresponding to a vertex in  $H$  is of size  $O(\log n)$ . The lower bound then follows as it is known [34, 35] that  $G(n, \frac{1+\delta}{n})$  contains with high probability a subgraph with  $\Omega(n)$  vertices that is an  $\alpha$ -expander for an appropriate choice  $\alpha$ . Similar ideas are used to prove the existence of large minors with linear treewidth in the noisy trees (e.g., the noisy path).

Our algorithms for minimizing contagion and stopping contagion in graphs of bounded treewidth build on techniques designed to exploit the tree-like nature of low treewidth graphs, sharing similarities to algorithms for target set selection in [8], where target set selection is the problem of finding a minimal set that infects an entire graph under the bootstrap percolation model. More directly, our problem resembles the Influence Diffusion Minimization (IDM) studied in [14], where the goal is to minimize the spread of the  $r$ -neighbor bootstrap percolation process by preventing spread through vertices. After subdividing edges, minimizing contagion essentially reduces to IDM, albeit with additional restrictions on the vertices we can immunize (only vertices that belong to the “middle” of a subdivided edge can be deleted); we therefore solve a generalization of the IDM problem and use this to provide efficient algorithms for the minimizing and stopping contagion.

At a high-level, our algorithm works by solving the stopping contagion recursively on subgraphs and then combining these solutions via dynamic-programming until we have a solution for the whole graph. To combine subproblems successfully, at each step we explicitly compute solutions for all possible states of vertices in a bag. While this could take exponential time in general, this approach provides an efficient algorithm in graphs with bounded treewidth.

Our proof of bounded spread in noisy trees builds works by proving that small subsets of such trees contain few edges [13, 25]. Since every non seed vertex needs at least two vertices to get infected, small contagious sets require small subsets that contain too many edges. Therefore, one can prove that small sets of seeds cannot infect too many vertices; the proof of small trees’ local sparsity is similar to the proof that w.h.p. such noisy trees have small local treewidth.

## 2.5 Related Work

While the idea to remove edges or vertices to contain an epidemic has been studied before [47, 10, 3], most of these works focus using edge or vertex deletions that break the graph to connected components of sublinear (or even constant) size [20, 47, 10]. Recently approximation algorithms for edge deletion problems that arise in controlling epidemics has been studied in [5] for the SIR epidemic model. In particular, [5] studies the problem of deleting a set of edges of weight at most  $B$  that minimizes the set of infected nodes after edges deletions. All these works consider a different contagion model from the  $r \geq 2$  bootstrap percolation model studied here.

Bootstrap percolation was first introduced by statistical physicists [11] and has been studied on a variety of graphs [6, 41, 25, 2, 50, 19, 48].

The fixed parameter tractability of minimizing contagion with respect to *vertex* deletions, as opposed to edge deletions, has been thoroughly investigated with respect to various parameters such as the maximum degree, treewidth, and the size of the seed set  $k$  in [14]. The authors of [14] present efficient algorithms for minimizing contagion for graphs of bounded maximum degree and treewidth. With respect to  $k$ , using ideas from FPT algorithms for cut problems [26], they give a  $2^{k+\ell} \text{poly}(n)$  algorithm for the case where the set of seeds is of size  $k$  and there is a solution of size  $\ell$  to the problem. Their algorithm can be easily adapted to the case of edge deletions: see Theorem 24. We are not aware of the stopping contagion

problem studied before, nor are we aware of previous studies of the minimizing contagion problem in random graphs. In order to deal with both stopping contagion and minimizing contagion for graphs of bounded treewidth, we build on algorithmic ideas from [8]. The NP-hardness of minimizing contagion with respect to vertex deletion is proved in [14] – our proof for the NP-hardness of the edge deletion version of minimizing contagion was found concurrently and independently; the proof is different from the proof appearing in [14].

There are two regimes of interest for the study of treewidth in sparse random graphs. For the subcritical regime  $p \leq d/n$  with  $d < 1$ ,  $G(n, p)$  has w.h.p. unicyclic connected components of size  $O(\log n)$  [23] and hence has treewidth at most 2. For the supercritical regime with  $p \geq d/n$  and  $d > 1$ ,  $G(n, p)$  has w.h.p. a giant component of size  $\Omega(n)$  [23] and determining the treewidth is more complicated. Klops [32] proved that the treewidth of  $G(n, d/n)$  is  $\Omega(n)$  w.h.p. for  $d \geq 2.36$ . His result was improved by Gao [28] who showed that for  $d \geq 2.16$ , the treewidth of  $G(n, d/n)$  is  $\Omega(n)$  with high probability. Gao asked if his result can be strengthened to prove that  $G(n, d/n)$  has treewidth linear in  $n$  w.h.p. for any  $d > 1$ ; this was later shown in [38]. A different and somewhat simplified proof establishing that the treewidth of  $G(n, d/n)$  is  $\Omega(n)$  w.h.p. was given in [46]. Finally, the fine-grained behavior of treewidth of  $G(n, (1 + \epsilon)/n)$  was studied in [17] where it was shown that for sufficiently small  $\epsilon$ , the treewidth of  $G(n, (1 + \epsilon)/n)$  is w.h.p.

$$\Omega\left(\frac{\epsilon^3}{\log 1/\epsilon}\right)n.$$

The first lower bound for the treewidth of random regular graphs appears to be from [46]: the authors prove that for every constant  $d > d_0$  where  $d_0$  is a sufficiently large constant, the treewidth of the random regular graph  $G(n, d)$  is  $\Omega(n)$  w.h.p. In [24] it was also shown that random graphs with a given degree sequence (with bounded maximum degree) that ensure the existence of a giant component w.h.p. (namely a degree sequence satisfying the Molloy-Reed criterion [40]) have linear treewidth as well, which implies, using a different argument than in [46], that  $G(n, d)$  for  $d > 2$  has linear treewidth w.h.p. A different proof for the linear lower bound of the treewidth of  $G(n, d)$  for  $d > 2$  is given in [17].

Several papers have examined notions of local treewidth in devising algorithms for algorithmic problems such as subgraph isomorphism [22, 30, 29, 27]. For example, Grohe [29] defines a graph family  $\mathcal{C}$  of having bounded local treewidth if there exists a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for every graph  $G = (V, E)$  in  $\mathcal{C}$  and every integer  $r$ , for every vertex  $v \in V$  the treewidth of the subgraph of  $G$  induced on all vertices of distance at most  $r$  from  $v$  is at most  $f(r)$ . These works primarily focus on planar graphs and graphs avoiding a fixed minor. The only work we are aware of that has examined the local treewidth of random graphs is that of [18]. Their main goal is to demonstrate that the treewidth of balls of radius  $r$  around a given vertex depends only on  $r$ , as opposed to analyzing the local treewidth as function of  $n, d$  and  $k$  as we do here. We employ a similar edge excess argument to the one in [18] although there are some differences in the analysis and the results: please see Section 3 for more details. We are not aware of previous work lower bounding the local treewidth of random graphs.

Embedding minors in expanders has received attention in combinatorics [37] and theoretical computer science, finding applications in proof complexity [4]. Kleinberg and Rubinfeld [31] proved that if  $G = (V, E)$  is a  $\alpha$ -vertex expander with maximum degree  $\Delta$ , then every graph with  $n/\log^\kappa n$  vertices and edges is a minor of  $G$  for a constant  $\kappa > 1$  depending on  $\Delta$  and  $\alpha$ . Later it was stated [12] that  $\kappa(\Delta, \alpha) = \Omega(\log^2(d)/\log^2(1/\alpha))$ . Krivelevich [35] together with Nandov proved that if  $G$  is an  $\alpha$ -vertex expander then it contains every graph with  $cn/\log n$  edges and vertices for some universal constant  $c > 0$ .

The sparsity of random graphs as well as randomly perturbed trees was used in showing that these families have w.h.p. bounded expansion<sup>5</sup> [43, 15]. These results are incomparable with our treewidth results: it is known that graphs with bounded maximum degree have bounded expansion and that  $G(n, d/n)$  has bounded expansion w.h.p. [43, 42]. In contrast, there exist 3-regular graphs with linear treewidth and as previously mentioned the treewidth of  $G(n, d/n)$  is  $\Omega(n)$ .

## 2.6 Future Directions

Our work raises several questions. We consider undirected unweighted graphs. However directed edges can be more accurate in modeling epidemic spread [1] and some edges might be more costly to move than others. Extending our algorithms to directed weighted graphs is an interesting direction for future research.

Our upper and lower bounds for the local treewidth of  $G(n, p)$  (with  $p = d/n$ ) currently differ by a multiplicative factor of order  $\log d$ . We believe that for  $k \leq n^{1-\epsilon}$  the local treewidth of  $G(n, p)$  is w.h.p.  $\Omega(k \log d / \log n)$ . Whether this is indeed the case remains for future work. Our upper bounds on the local treewidth of noisy trees can be made independent of the maximum degree of the tree; namely, for arbitrary trees, the local treewidth should be upper bounded w.h.p. by  $O(k / \log n)$  assuming  $k$  is not too large. Proving or disproving this however remains open. Understanding how well one can approximate minimizing contagion and stopping contagion in general graphs, as well as graphs with certain structural properties (e.g. planar graphs) is a potential direction for future research as well. Finally, it could be of interest to study if our bounds for local treewidth coupled with sophisticated algorithms for graphs with bounded local treewidth [27, 29, 22] could lead to improved running time for additional algorithmic problems in random graphs.

## 2.7 Preliminaries

Throughout the paper  $\log$  denotes the logarithm function with base 2; we omit floor and ceiling signs to improve readability. All graphs considered are undirected and have no parallel edges. Given a graph  $G = (V, E)$  and two disjoint sets of vertices  $A, B$  we denote by  $E(A, B)$  the set of edges connecting a vertex in  $A$  to a vertex in  $B$ . For  $A, B$  as above we denote by  $N_G(A, B)$  the set of vertices in  $B$  with a neighbor in  $A$ . For a subset of vertices  $A \subseteq V$  and an edge  $e$  we say that  $A$  *touches*  $e$  if at least one of the endpoints of  $e$  belongs to  $A$ . If both endpoints of  $e$  belong to  $A$  then we say that  $A$  *spans*  $e$ .

A graph  $H$  is a *minor* of  $G$  if  $H$  can be obtained from  $G$  by repeatedly doing one of three operations: deleting an edge, contracting an edge or deleting a vertex. We keep our graphs simple and remove any parallel edges that may form during contractions. It can be verified [42] that a graph  $H$  with  $k$  vertices is a minor of  $G$  if and only if there are  $k$  vertex disjoint connected subgraphs of  $G, C_1 \dots C_k$  such that for every edge  $(v_i, v_j)$  of  $H$ , there is an edge connecting a vertex in  $C_i$  to a vertex of  $C_j$ . We refer to the map mapping every vertex of  $H, v_j$  to  $C_j$  as an *embedding* of  $H$  in  $G$ ; the maximum vertex cardinality of  $C_i, 1 \leq i \leq k$  is called the *width* of the embedding. We shall be relying on the well-known fact [42, 16] that if  $H$  is a minor of  $G$  then the treewidth of  $G$  is lower bounded by the treewidth of  $H$ .

We will also need the following definition of an edge expander:

<sup>5</sup> Bounded expansion should not be confused with the edge expansion of a graph. For a precise definition please see [43, 42].

► **Definition 13.** Let  $\alpha \in (0, 1)$ . A graph  $G$  is an  $\alpha$ -expander if every subset of vertices  $S$  with  $|S| \leq n/2$  satisfies

$$|N_G(S, V \setminus S)| \geq \alpha|S|.$$

### 3 Local Treewidth of Random Graphs

In this section we prove both an upper and lower bound for  $t_k(G(n, p))$  that with high probability. We assume  $k \leq n^{1-\epsilon}$  for a constant  $\epsilon > 0$ .

#### 3.1 Upper Bound

Our main idea in upper bounding  $t_k(G)$  is to leverage the fact that  $G(n, p)$  is locally sparse and that if a few edges are added on top of a tree, the treewidth of the resulting graph cannot grow too much.

► **Lemma 14.** Let  $G$  be a connected graph with  $n$  vertices and  $n - 2 + \ell$  edges. Then  $\text{tw}(G) \leq \ell$ .

**Proof.** Since  $G$  is connected, it must have a spanning tree  $T$  with  $n$  vertices and  $n - 1$  edges. The graph  $G$  has exactly  $\ell - 1$  additional edges; since adding an edge can increase a graph's treewidth by at most 1, we immediately get the desired bound. ◀

$$\text{tw}(G) \leq \text{tw}(T) + \ell - 1 = \ell$$

We can now prove:

► **Theorem 15.** Suppose that  $k \leq n^{1-\epsilon}$ . Then for  $G = G(n, p)$  we have that w.h.p. for every  $m \leq k$ :

$$t_m(G) \leq 3 + O\left(\frac{m \log d}{\log n}\right).$$

**Proof.** Since the Theorem is obvious for  $d = n^{\Omega(1)}$  we assume that  $d \leq n^{\epsilon/2}$ . We first prove the statement for  $m = k$ . Given a graph  $G$  with treewidth  $t$ , it is always possible to find a connected subgraph of  $G$  with identical treewidth to  $G$ . In that spirit, rather than bounding the probability there exists some  $k$ -vertex subgraph of  $G$  with treewidth exceeding some  $r$ , we bound the probability some subgraph on  $s \leq k$  vertices is connected and has treewidth greater than  $r$  in  $G$ .

Fix some  $S \subseteq V$  with exactly  $s$  vertices. Note there are  $s^{s-2}$  possible spanning trees which could connect the vertices in  $S$ , each requiring  $s - 1$  edges. While the resulting subgraph would be connected, its treewidth is only 1. Therefore,  $r$  additional edges would also be required to produce a subgraph with treewidth at least  $r + 1$ . Accounting for the ways to choose these edges, the probability the subgraph induced on  $S$  is connected and has treewidth greater than  $r$  is at most

$$s^{s-2} \binom{\binom{s}{2}}{r} \left(\frac{d}{n}\right)^{r+s-1}.$$

This follows since each edge occurs independently with probability  $p = d/n$ . Now, we bound the probability that any such subset  $S$  with at most  $k$  vertices exists. To that end, we take a union bound over all  $\binom{n}{s}$  possible subsets of  $s$  vertices, letting  $s$  range from 1 to  $k$ . Putting this together and using the inequality  $\binom{a}{b} \leq (ea/b)^b$  yields

$$\begin{aligned} \sum_{s=1}^k \binom{n}{s} \times s^{s-2} \binom{\binom{s}{2}}{r} \left(\frac{d}{n}\right)^{r+s-1} &\leq \frac{d^r}{n^{r-1}} \sum_{s=1}^k e^s \left(\frac{es^2}{2r}\right)^r d^s \\ &\leq \frac{d^r}{n^{r-1}} k e^k \left(\frac{ek^2}{r}\right)^r d^k \end{aligned}$$

To complete the proof, notice this probability can be made to be at most  $n^{-1}$  (using  $k \leq n^{1-\epsilon}$  and  $d \leq n^{\epsilon/2}$ ) when  $r$  is taken to be

$$2 + O\left(\frac{k \log d}{\log n}\right).$$

The Theorem now follows for  $m = k$  from Lemma 14. Using the above proof along with a simple union bound over all  $m \leq k \leq n^{1-\epsilon}$  implies the statement for all  $m \leq k$ . ◀

Notice the approach above yields a sharper bound than if we solely attempted to bound the treewidth by counting the number of excess edges above  $k - 1$ . To explain, notice a  $k$ -vertex subgraph can have treewidth  $r$  only if it has at least  $r + k - 1$  edges. A simple union bound over all possible subsets of  $k$  vertices, upper bounds the probability we are interested in.

$$\binom{n}{k} \binom{\binom{k}{2}}{r+k-1} \left(\frac{d}{n}\right)^{r+k-1} \leq \frac{k^{2k} k^{2r} d^k d^r}{n^{r-1}}$$

This is implicitly used in [18] to bound the treewidth of balls of radius  $r$  in  $G(n, p)$ ; as mentioned above, our method improves on this result. More concretely, since the upper bound now has a additional  $k^k$  factor in the numerator, using this in our application would yield the weaker upper bound

$$t_k(G) = 3 + O\left(\frac{k(\log k + \log d)}{\log n}\right).$$

### 3.2 Lower Bound

Throughout this section we assume that  $d > 1 + \delta$  where  $\delta > 0$ .

First we need the following result from [35]:

► **Proposition 16.** *Consider the random graph  $G := G(n, \frac{1+\delta}{n})$ . Then there is a constant  $c > 0$  depending on  $\delta$  such that for every graph  $H$  with at most  $k$  vertices and edges,  $G$  contains an embedding  $H'$  of  $H$ . Furthermore the width of the embedding is  $O(\log n)$ .*

► **Proposition 17.** *There exist graphs with  $m$  vertices and  $m$  edges of treewidth  $\Omega(m)$ .*

**Proof.** As random 3-regular graphs have with high probability linear treewidth [17, 24] there are  $m$ -vertex graphs with  $m$  vertices and  $3m/2$  edges and treewidth  $\Omega(m)$ . Adding to such a graph  $m/2$  isolated vertices results with a graph with the desired property. ◀

Using our results we can lower bound the local treewidth of a random graph:

► **Theorem 18.** *Let  $G := G(n, d/n)$  be a random graph with  $d > 1 + \delta$ . Assume  $k \leq O(n/\log n)$ . Then w.h.p.  $G$  contains a subgraph with  $O(k)$  vertices whose treewidth is  $\Omega(\frac{k}{\log n})$ .*



**Proof.** We may assume that  $k = \Omega(\log n)$ , otherwise the lower bound in the Theorem is immediate. Let  $H$  be a graph with  $s$  vertices and edges of treewidth  $\Omega(s)$ . Let  $s \leq \left(\frac{n}{\log n}\right)$ . By Proposition 16  $G$  contains an embedding of  $H, H'$  of width  $O(\log n)$ . It follows that  $H'$  has at most  $O(s \log n)$  vertices and treewidth at least  $\Omega(s)$  (as  $H$  is a minor of  $H'$ ) which is what we wanted to prove.  $\blacktriangleleft$

### 3.3 Local Treewidth of Random Regular Graphs

Similar bounds on the local treewidth of random regular graphs  $G(n, d)$  can be established via similar arguments to those used for  $G(n, d/n)$ . For the upper bound, one can use the fact [13] that for every  $k < nd/4$  distinct unordered pairs of vertices, the probability they all occur simultaneously in  $G(n, d)$  is at most  $(2d/n)^k$  and then nearly identical arguments to those in Theorem 15. The lower bound follows easily from embedding results for expanders:

► **Theorem 19.** *Let  $d > 2$  be a constant. Then with high probability a random  $d$ -regular graph  $G$  is minor universal: any graph  $H$  with at most  $O(n/\log n)$  vertices and edges can be embedded into  $G$ . Furthermore, the width of the embedding is  $O(\log n)$ .*

**Proof.** By a result of [35] if  $G$  is an  $\alpha$ -expander with  $\alpha > 0$  bounded away from zero then the claim in the Proposition hold. The result now follows as it is well known [9, 33] that with high probability the random  $d$ -regular graph is an  $\alpha$ -expander for  $\alpha > 0$ .  $\blacktriangleleft$

We summarize this with the following Theorem:

► **Theorem 20.** *Suppose that  $2 < d$  is a constant and  $k \leq n^{1-\epsilon}$  for some constant  $\epsilon \in (0, 1)$ . Then for  $G = G(n, d)$  we have that w.h.p.:*

$$\Omega\left(\frac{k}{\log n}\right) \leq t_k(G) \leq 3 + O\left(\frac{k \log d}{\log n}\right).$$

## 4 Local treewidth of Noisy Graphs

We study the local treewidth of noisy graphs: Recall that in this model there is a base  $n$ -vertex graph  $G$  with maximum degree  $\Delta$ . On top of this base graph every non edge of  $G$  is added independently with probability  $1/n$ . All proofs missing from this section can be found in [39]. Our main result is:

► **Theorem 21.** *Let  $G$  be an  $n$ -vertex connected graph of maximum degree  $\Delta$ . Suppose that we add every non-edge of  $G$  to  $G$  with probability  $1/n$  independently of all other random edges. Call the resulting graph  $G'$ . With high probability, then,  $t_k(G') \leq O(t_k(G) + r)$ , where*

$$r = 3 + O\left(\frac{k(\log k + \log \Delta)}{\log n}\right).$$

For a proof please see the full version [39].

The upper bound in Theorem 21 is nearly tight for certain noisy trees.

► **Theorem 22.** *Consider the  $n$  vertex path,  $P_n$ . Suppose we add every nonedge to  $P_n$  with probability  $\epsilon/n$  where  $\epsilon > 0$  is an arbitrary constant. Call the perturbed graph  $P'$ . Then with high probability for any  $\Omega(\log n) \leq k \leq O(n/\log n)$ , there exists a subgraph of  $P'$  with  $O(k)$  vertices with treewidth  $\Omega(k/\log n)$ .*



**Proof.** Fix  $B$  to be a large enough constant. Chop  $P_n$  to  $n/B$  disjoint paths<sup>6</sup>  $A_1 \dots A_{n/B}$  each of length  $B$ . Consider now the graph  $G$  whose vertex set is  $A_1 \dots A_{n/B}$  and two vertices  $A_i$  and  $A_j$  are connected if there is an edge (in  $P'$ ) connecting  $A_i$  to  $A_j$ . The probability two vertices in  $G$  are connected is at least

$$1 - (1 - \epsilon/n)^{B^2} \geq \epsilon B^2/2n.$$

For a fixed graph  $H$  with  $s$  vertices and edges, it is known [35] that the supercritical random graph  $G(m, \frac{1+\epsilon}{m})$  contains an embedding of  $H$  into  $G$  as long as  $s = O(m/\log m)$ . Furthermore the width of the embedding is  $O(\log m)$ . The probability that two vertices in  $G$  are connected is larger than  $\frac{1+\epsilon}{n/B}$ . Therefore we can embed  $H$  into a subgraph  $H'$  of  $G$  whose size is at most  $s \log n$  such that  $H$  is a minor of  $H'$ . Furthermore as the vertices of  $G$  are paths of length  $B$  (in  $P_n$ ), the embedding of  $H$  into  $G$  directly translates to an embedding of  $H$  into  $P'$  whose width is  $O(B \log n) = O(\log n)$ . Choosing  $H$  with  $s$  vertices and edges and treewidth  $\Omega(s)$  concludes the proof.  $\blacktriangleleft$

## 5 Algorithms for Graphs of Bounded Treewidth

In this section, we build on the results of [14] to provide polynomial time algorithms for bounded treewidth instances of minimizing contagion and stopping contagion. As we sketched in our introduction, we generalize the influence diffusion minimization problem introduced by the authors and use a similar dynamic-programming algorithm. Our main result is the following algorithm for graphs of bounded treewidth  $\tau$ :

► **Theorem 23.** *Let  $G$  be an  $n$  vertex graph with maximum degree  $\Delta$ , maximum threshold  $r$  and treewidth  $\tau$ . Then both minimizing and stopping contagion can be solved in time  $O(\tau 1296^\tau \min\{r, \max\{\Delta, 2\}\}^{4\tau} \text{poly}(n))$ .*

For a proof, including a description of our algorithm and runtime analysis, please see the full version [39]. Note that to combine subproblems, we must effectively account for the effect of infected vertices elsewhere on each subgraph we consider. We therefore essentially solve minimizing contagion and stopping contagion in a more flexible infection model, where thresholds are allowed to differ between vertices but remain at most  $r$ ; as a result, our theorem cleanly translates to this setting as well.

## 6 Algorithms for Minimizing and Stopping Contagion in Grids, Random Graphs and Noisy Trees

In this section we study how to solve minimizing contagion and stopping contagion when the set of seeds  $A$  is not too large and does not spread by too much. We use this along with local treewidth upper bounds to devise algorithms for minimizing and stopping contagion in random graphs. We also consider algorithms for grids and planar graphs. As usual all missing proofs appear in [39].

Using similar ideas to [14] (who consider vertex deletions problems) we have the following result for the minimizing contagion problem whose proof can be found in [39].

► **Theorem 24.** *Let  $G = (V, E)$  be an  $n$ -vertex graph. Suppose there are  $t$  edges whose removal ensures no more than  $r$  vertices are infected in  $G$  from the seed set  $A \subseteq V$ . Then minimizing contagion can be solved optimally in (randomized)  $2^{r+t} \text{poly}(n)$  time where  $n$  is the number of vertices.*

<sup>6</sup> To simplify the presentation we assume  $B$  divides  $n$ . Similar ideas work otherwise.

The algorithm above can become slow if  $r$  or  $t$  are very large. Additionally, we do not know how to get similar results (e.g., algorithms of running time  $2^{|A|} \text{poly}(n)$ ) for stopping contagion. Below we show that we can improve upon this algorithm for graphs that have some local sparsity conditions. A key property we use is that for both minimizing contagion and stopping contagion with a seed set  $A$ , we restrict our attention to the subgraph of  $G$  induced on  $\langle A \rangle$ .

## 6.1 Grids and Planar Graphs

Consider the  $n \times n$  grid where all vertices have threshold at least 2 we have the following “bounded spread” result:

► **Lemma 25.** *In the  $n \times n$  grid every set of size  $k$  infects no more than  $O(k^2)$  vertices.*

**Proof.** Embed the  $n \times n$  grid  $G = \{1, \dots, n\} \times \{1, \dots, n\}$  in  $H = \{0, \dots, n+1\} \times \{0, \dots, n+1\}$  in the natural way. Given a subset  $A$  of  $G$ , the *perimeter* of  $A$  is the set of all vertices not belonging to  $A$  having a neighbor in  $A$ . The crucial observation is that if  $A$  is a set of infected seeds, the perimeter of  $A$  can never increase during the contagion process [7]. As the perimeter of  $A$  is at most  $4k$  the infected set has perimeter at most  $4k$  as well. The result follows as every set  $A \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$  of size  $m$  has perimeter  $\Omega(\sqrt{m})$ . ◀

Using Theorem 24 we have that minimizing contagion on the  $n$  by  $n$  grid with  $k = |A|$  can be solved in time  $2^{O(k^2)} \text{poly}(n)$ . We simply apply the algorithm in Theorem 24 to  $\langle A \rangle$ . Alternatively we can use exhaustive search over all subsets of edges in the graph induced on  $\langle A \rangle$  to solve<sup>7</sup> both minimizing or stopping contagion. We can do better using the following fact:

► **Lemma 26.** *Let  $G$  be a subgraph of an  $n$  by  $n$  grid with  $r$  vertices. Then  $G$  has treewidth  $O(\sqrt{r})$ .*

**Proof.** Every  $m$ -vertex planar graph has treewidth  $O(\sqrt{m})$ . ◀

► **Corollary 27.** *Let  $G = (V, E)$  be the  $n$  by  $n$  grid. Suppose  $H = (V, E')$  where  $E' \subseteq E$  and every vertex has a threshold of at least 2. Let  $A$  be the seed set with  $k = |A|$ . Then stopping contagion and minimizing contagion can be solved in time  $2^{O(k)} \text{poly}(n)$ .*

**Proof.** For solving either problems we only need to consider the subgraph of  $G$ ,  $\langle A \rangle$ . The result now follows from Theorem 23. ◀

Similarly, for a planar graph where every vertex has threshold at least 2 and at most  $b$  and every subset  $A$  of size  $k$  infects at most  $f(k)$  vertices, stopping contagion can be solved in time  $b^{O(\sqrt{f(k)})} \text{poly}(n)$ .

## 6.2 Sparse Random Graphs

Consider the random graph  $G(n, d/n)$  assuming all vertices have threshold larger than 1. Assuming  $d \leq n^{1/2-\delta}$  for  $\delta \in (0, 1/2)$ , it is known [25] that with high probability every set of size  $O(\frac{n}{d^2 \log d})$  does not infect more than  $O(|A| \log d)$  vertices. Furthermore, it is known [25] that any set of size  $O(n/d^2)$  has with high probability constant average degree. It follows that assuming  $|A| = O(\frac{n}{d^2 \log d})$  the optimal solution to minimizing contagion is of size

<sup>7</sup> For minimizing contagion using the FPT algorithm may be preferable as it may run significantly faster if the optimal solution has cardinality  $o(k^2)$ .

$O(|A| \log d)$ . Therefore in random graphs with  $|A| \leq O(\frac{n}{d^2 \log d})$ , minimizing contagion can be solved using Theorem 24 in time  $O(2^{|A| \log d} \text{poly}(n))$ . As before, exhaustive search over all edges on the graph induced on  $\langle A \rangle$  can solve both minimizing and stopping contagion in time  $O(2^{|A| \log d} \text{poly}(n))$  as well.

Using our local treewidth estimates, Theorem 23, the bounded spread property and the fact that w.h.p the maximum degree of  $G$  is  $O(\log n / \log \log n)$  we have the following improvement for the running time:

► **Theorem 28.** *Let  $G := G(n, d/n)$ ,  $\epsilon \in (0, 1)$  and  $\delta \in (0, 1/2)$ . Denote by  $k$  to be the size of the seed set  $A$ . Suppose that  $k \leq O(\min(n^{1-\epsilon}, \frac{n}{d^2 \log d}))$  and  $d \leq n^{1/2-\delta}$ , and that every vertex has threshold larger than 1. Then w.h.p both minimizing contagion and stopping contagion can be solved in time*

$$\exp \left( O \left( \frac{k \log^2 d \log \log n}{\log n} \right) \right) \text{poly}(n).$$

**Proof.** As before we can solve either problem on  $\langle A \rangle$  using the upper bound on the treewidth from Theorem 15, the fact that with high probability  $|\langle A \rangle| \leq O(\log d |A|)$  and the algorithm for graphs of bounded treewidth for stopping or minimizing contagion. ◀

### 6.3 Noisy trees

We now devise an algorithm for stopping contagion and minimizing contagion for noisy trees. To achieve this we first prove that for forests every sets of seeds does not spread by much and furthermore this property is maintained after adding a “small” number of edges on top of the edges belonging to the forest. Then we use similar ideas to Theorem 21 and prove that noisy trees are locally sparse in the sense that every subsets of vertices of cardinality  $k$  spans w.h.p  $k + o(k)$  edges assuming  $k$  is not too large. We use this property to prove that any subset  $A$  of  $k$  seeds infects w.h.p  $O(k)$  vertices. Thereafter we can use the algorithms for bounded treewidth to solve either minimizing contagion or stopping contagion on  $\langle A \rangle$ . We assume throughout this section that  $\epsilon \in (0, 1)$  is a sufficiently small constant ( $\epsilon < 1/100$  would suffice for our proofs to go through). Using these ideas we can prove the following Theorem whose complete proof can be found in [39].

► **Theorem 29.** *Let  $T$  be a tree and let  $T'$  be the noisy tree obtained from  $T$ . Assume  $|A| = k$ ,  $\Delta \leq n^\epsilon$  and that every vertex has threshold larger than 1. Let  $m := \max(\log \log n, \log \Delta)$ . Then both minimizing contagion and stopping contagion can be solved in  $T'$  in time*

$$\exp \left( O \left( \frac{k(\log k + \log \Delta)m}{\log n} \right) \right) \text{poly}(n).$$

---

#### References

- 1 Antoine Allard, Cristopher Moore, Samuel V Scarpino, Benjamin M Althouse, and Laurent Hébert-Dufresne. The role of directionality, heterogeneity and correlations in epidemic risk and spread. *arXiv preprint*, 2020. [arXiv:2005.11283](#).
- 2 Hamed Amini and Nikolaos Fountoulakis. Bootstrap percolation in power-law random graphs. *Journal of Statistical Physics*, 155(1):72–92, 2014.
- 3 James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *Journal of Computer and System Sciences*, 72(6):1077–1093, 2006.
- 4 Per Austrin and Kilian Risse. Perfect matching in random graphs is as hard as Tseitin\*. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 979–1012. SIAM, 2022.

- 5 Amy Babay, Michael Dinitz, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti. Controlling epidemic spread using probabilistic diffusion models on networks. *arXiv preprint*, 2022. [arXiv:2202.08296](#).
- 6 József Balogh and Béla Bollobás. Bootstrap percolation on the hypercube. *Probability Theory and Related Fields*, 134(4):624–648, 2006.
- 7 József Balogh and Gábor Pete. Random disease on the square grid. *Random Structures & Algorithms*, 13(3-4):409–422, 1998.
- 8 Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, 2011.
- 9 Béla Bollobás. The isoperimetric number of random regular graphs. *European Journal of combinatorics*, 9(3):241–244, 1988.
- 10 Alfredo Braunstein, Luca Dall’Asta, Guilhem Semerjian, and Lenka Zdeborová. Network dismantling. *Proceedings of the National Academy of Sciences*, 113(44):12368–12373, 2016.
- 11 John Chalupa, Paul L Leath, and Gary R Reich. Bootstrap percolation on a bethe lattice. *Journal of Physics C: Solid State Physics*, 12(1):L31, 1979.
- 12 Julia Chuzhoy and Rachit Nimavat. Large minors in expanders. *arXiv preprint*, 2019. [arXiv:1901.09349](#).
- 13 Amin Coja-Oghlan, Uriel Feige, Michael Krivelevich, and Daniel Reichman. Contagious sets in expanders. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms*, pages 1953–1987. SIAM, 2014.
- 14 Gennaro Cordasco, Luisa Gargano, and Adele Anna Rescigno. Parameterized complexity of immunization in the threshold model. *arXiv preprint*, 2021. [arXiv:2102.03537](#).
- 15 Erik D Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *arXiv preprint*, 2014. [arXiv:1406.2587](#).
- 16 Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173, 2005.
- 17 Tuan Anh Do, Joshua Erde, and Mihyun Kang. A note on the width of sparse random graphs. *arXiv preprint*, 2022. [arXiv:2202.06087](#).
- 18 Jan Dreier, Philipp Kuinke, Ba Le Xuan, and Peter Rossmanith. Local structure theorems for erdős-rényi graphs and their algorithmic applications. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 125–136. Springer, 2018.
- 19 Roozbeh Ebrahimi, Jie Gao, Golnaz Ghasemiesfeh, and Grant Schoenebeck. Complex contagions in kleinberg’s small world model. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 63–72, 2015.
- 20 Jessica Enright and Kitty Meeks. Deleting edges to restrict the size of an epidemic: a new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018.
- 21 Jessica Enright, Kitty Meeks, George B Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021.
- 22 David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Graph Algorithms and Applications I*, pages 283–309. World Scientific, 2002.
- 23 Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5(1):17–60, 1960.
- 24 Uriel Feige, Jonathan Hermon, and Daniel Reichman. On giant components and treewidth in the layers model. *Random Structures & Algorithms*, 48(3):524–545, 2016.
- 25 Uriel Feige, Michael Krivelevich, Daniel Reichman, et al. Contagious sets in random graphs. *The Annals of Applied Probability*, 27(5):2675–2697, 2017.
- 26 Fedor V Fomin, Petr A Golovach, and Janne H Korhonen. On the parameterized complexity of cutting a few vertices from a graph. In *International Symposium on Mathematical Foundations of Computer Science*, pages 421–432. Springer, 2013.
- 27 Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM (JACM)*, 48(6):1184–1206, 2001.

- 28 Yong Gao. Treewidth of erdős-rényi random graphs, random intersection graphs, and scale-free random graphs. *Discrete Applied Mathematics*, 160(4-5):566–578, 2012.
- 29 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 4(23):613–632, 2003.
- 30 Mohammad Taghi Hajiaghayi. *Algorithms for graphs of (locally) bounded treewidth*. PhD thesis, Citeseer, 2001.
- 31 Jon Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 86–95. IEEE, 1996.
- 32 Ton Kloks. *Treewidth: computations and approximations*. Springer, 1994.
- 33 Brett Kolesnik and Nick Wormald. Lower bounds for the isoperimetric numbers of random regular graphs. *SIAM Journal on Discrete Mathematics*, 28(1):553–575, 2014.
- 34 Michael Krivelevich. Finding and using expanders in locally sparse graphs. *SIAM Journal on Discrete Mathematics*, 32(1):611–623, 2018.
- 35 Michael Krivelevich. Expanders – how to find them, and what to find in them. *Surveys in Combinatorics*, 456:115–142, 2019.
- 36 Michael Krivelevich, Daniel Reichman, and Wojciech Samotij. Smoothed analysis on connected graphs. *SIAM Journal on Discrete Mathematics*, 29(3):1654–1669, 2015.
- 37 Michael Krivelevich and Benjamin Sudakov. Minors in expanding graphs. *Geometric and Functional Analysis*, 19(1):294–331, 2009.
- 38 Choongbum Lee, Joonkyung Lee, and Sang-il Oum. Rank-width of random graphs. *Journal of Graph Theory*, 70(3):339–347, 2012.
- 39 Hermish Mehta and Daniel Reichman. Local treewidth of random and noisy graphs with applications to stopping contagion in networks. *arXiv preprint*, 2022. [arXiv:2204.07827](https://arxiv.org/abs/2204.07827).
- 40 Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, 6(2-3):161–180, 1995.
- 41 Natasha Morrison and Jonathan A Noel. Extremal bounds for bootstrap percolation in the hypercube. *Journal of Combinatorial Theory, Series A*, 156:61–84, 2018.
- 42 Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012.
- 43 Jaroslav Nešetřil, Patrice Ossona de Mendez, and David R Wood. Characterisations and examples of graph classes with bounded expansion. *European Journal of Combinatorics*, 33(3):350–373, 2012.
- 44 Mark EJ Newman, DJ Walls, Mark Newman, Albert-László Barabási, and Duncan J Watts. Scaling and percolation in the small-world network model. In *The Structure and Dynamics of Networks*, pages 310–320. Princeton University Press, 2011.
- 45 Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.
- 46 Guillem Perarnau and Oriol Serra. On the tree-depth of random graphs. *Discrete Applied Mathematics*, 168:119–126, 2014.
- 47 Xiao-Long Ren, Niels Gleinig, Dirk Helbing, and Nino Antulov-Fantulin. Generalized network dismantling. *Proceedings of the national academy of sciences*, 116(14):6554–6559, 2019.
- 48 Eric Riedl. Largest and smallest minimal percolating sets in trees. *The electronic journal of combinatorics*, pages P64–P64, 2012.
- 49 Prathyush Sambaturu, Bijaya Adhikari, B Aditya Prakash, Srinivasan Venkatramanan, and Anil Vullikanti. Designing effective and practical interventions to contain epidemics. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1187–1195, 2020.
- 50 Grant Schoenebeck and Fang-Yi Yu. Complex contagions on configuration model graphs with a power-law degree distribution. In *International Conference on Web and Internet Economics*, pages 459–472. Springer, 2016.



# Beyond Single-Deletion Correcting Codes: Substitutions and Transpositions

Ryan Gabrys   




ECE Department, University of California, San Diego, CA, USA

Venkatesan Guruswami   

EECS Department, University of California, Berkeley, CA, USA

João Ribeiro   

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Ke Wu   

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

We consider the problem of designing low-redundancy codes in settings where one must correct deletions in conjunction with substitutions or adjacent transpositions; a combination of errors that is usually observed in DNA-based data storage. One of the most basic versions of this problem was settled more than 50 years ago by Levenshtein, who proved that binary Varshamov-Tenengolts codes correct one arbitrary edit error, i.e., one deletion *or* one substitution, with nearly optimal redundancy. However, this approach fails to extend to many simple and natural variations of the binary single-edit error setting. In this work, we make progress on the code design problem above in three such variations:

- We construct linear-time encodable and decodable length- $n$  non-binary codes correcting a single edit error with nearly optimal redundancy  $\log n + O(\log \log n)$ , providing an alternative simpler proof of a result by Cai, Chee, Gabrys, Kiah, and Nguyen (IEEE Trans. Inf. Theory 2021). This is achieved by employing what we call *weighted VT sketches*, a new notion that may be of independent interest.
- We show the existence of a binary code correcting one deletion *or* one adjacent transposition with nearly optimal redundancy  $\log n + O(\log \log n)$ .
- We construct linear-time encodable and list-decodable binary codes with list-size 2 for one deletion *and* one substitution with redundancy  $4 \log n + O(\log \log n)$ . This matches the existential bound up to an  $O(\log \log n)$  additive term.

**2012 ACM Subject Classification** Theory of computation → Error-correcting codes

**Keywords and phrases** Synchronization errors, Optimal redundancy, Explicit codes

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.8

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2112.09971>

**Funding** *Venkatesan Guruswami:* Research supported in part by the NSF grants CCF-1814603 and CCF-2107347. Part of the work was done while at Carnegie Mellon University.

*João Ribeiro:* Research supported in part by the NSF grants CCF-1814603 and CCF-2107347 and by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

*Ke Wu:* Research supported in part by a DARPA SIEVE award, SRI Subcontract Number 53978, and DARPA Prime Contract Number HR00110C0086.



© Ryan Gabrys, Venkatesan Guruswami, João Ribeiro, and Ke Wu;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 8; pp. 8:1–8:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

Deletions, substitutions, and transpositions are some of the most common types of errors jointly affecting information encoded in DNA-based data storage systems [27, 14]. Therefore, it is natural to consider models capturing the interplay between these types of errors, along with the best possible codes for these settings. More concretely, one usually seeks to pin down the optimal redundancy required to correct such errors, and also to design fast encoding and decoding procedures for low-redundancy codes. It is well-known that deletions are challenging to handle even in isolation, since they cause a loss of synchronization between sender and receiver. The situation where one aims to correct deletions in conjunction with other reasonable types of errors is even more difficult. Our understanding of this interplay remains scarce even in basic settings where only one or two such worst-case errors may occur.

One of the most fundamental settings where deletions interact with the other types of errors mentioned above is that of correcting a single *edit* error (i.e., a deletion, insertion, or substitution) over a *binary* alphabet. In this case, linear-time encodable and decodable binary codes correcting a single edit error with nearly optimal redundancy have been known for more than 50 years. Levenshtein [13] showed that the binary Varshamov-Tenengolts (VT) code [24] defined as

$$\mathcal{C} = \left\{ x \in \{0, 1\}^n : \sum_{i=1}^n i \cdot x_i = a \pmod{(2n+1)} \right\} \quad (1)$$

corrects one arbitrary edit error. For an appropriate choice of  $a$ , this code has redundancy at most  $\log n + 2$ , and it is not hard to see that at least  $\log n$  bits of redundancy are required to correct one edit error. Remarkably, a greedy Gilbert-Varshamov-type argument only guarantees the existence of single-edit correcting codes with redundancy  $2 \log n$  – much higher than what can be achieved with the VT code. We recommend Sloane’s excellent survey [18] for a more in-depth overview of binary VT codes and their connections to combinatorics.

Although the questions of determining the optimal redundancy and giving nearly-optimal explicit constructions of codes in the binary single-edit setting have been settled long ago, the underlying approach fails to extend to many simple, natural variations of this setting combining deletions with substitutions and transpositions. In this work, we make progress on these questions in three such fundamental variations, which we proceed to describe next.

### 1.1 Non-binary single-edit correcting codes

We begin by considering the problem of correcting a single arbitrary edit error over a non-binary alphabet. This setting is especially relevant due to its connection to DNA-based data storage, which requires coding over a 4-ary alphabet. In this case, the standard *VT sketch*

$$f(x) = \sum_{i=1}^n i \cdot x_i \pmod{N}, \quad (2)$$

which allows us to correct one binary edit error in (1) with an appropriate choice of  $N$ , is no longer enough. Instead, we present a natural extension of the binary VT code to a non-binary alphabet via a new notion of *weighted VT sketches*, which yields an order-optimal result.

► **Theorem 1.** *There exists a 4-ary<sup>1</sup> single-edit correcting code  $\mathcal{C} \subseteq \{0, 1, 2, 3\}^n$  with  $\log n + \log \log n + 7 + o(1)$  bits of redundancy, where  $o(1) \rightarrow 0$  when  $n \rightarrow \infty$ . Moreover, there exists a single edit-correcting code  $\mathcal{C} \subseteq \{0, 1, 2, 3\}^n$  with  $\log n + O(\log \log n)$  redundant bits that supports linear-time encoding and decoding.*

<sup>1</sup> A 4-ary alphabet is relevant for DNA-based data storage.



This problem was previously considered by Cai, Chee, Gabrys, Kiah, and Nguyen [2], who proved an analogous result. Our existential result requires 6 fewer bits of redundancy than the corresponding result from [2], and our explicit code supports linear time encoding and decoding procedures, while the explicit code from [2] requires  $\Theta(n \log n)$  time encoding [22]. However, we believe that our more significant contribution in this setting is the simpler approach we employ to prove Theorem 1 via weighted VT sketches. The technique of weighted VT sketches seems quite natural and powerful and may be of independent interest.

We note that the existential result in Theorem 1 extends to arbitrary alphabet size  $q$  with  $\log n + O_q(\log \log n)$  redundant bits, but we focus on  $q = 4$  since it is the most interesting setting and provides the clearest exposition of our techniques. More details can be found in Section 3, where we also present a more in-depth discussion on why the standard VT sketch (2) does not suffice in the non-binary case.

## 1.2 Binary codes correcting one deletion *or* one adjacent transposition

As our second contribution, we consider the interplay between deletions and adjacent transpositions, which map 01 to 10 and vice-versa. An adjacent transposition may be seen as a special case of a burst of two substitutions. Besides its relevance to DNA-based storage, the interplay between deletions and transpositions is an interesting follow-up to the single-edit setting discussed above because the VT sketch is highly ineffective when dealing with transpositions, while it is the staple technique for correcting deletions and substitutions. The issue is that, if  $y, y' \in \{0, 1\}^n$  are obtained from  $x \in \{0, 1\}^n$  via any two adjacent transpositions of the form  $01 \mapsto 10$ , then  $f(y) = f(y') = f(x) - 1$ , where we recall  $f(z) = \sum_{i=1}^n i \cdot z_i \bmod N$  is the VT sketch. This implies that knowing the VT sketch  $f(x)$  reveals almost no information about the adjacent transposition, since correcting an adjacent transposition is equivalent to finding its location.

In this setting, the best known redundancy lower bound is  $\log n$  (the same as for single-deletion correcting codes), while the best known existential upper bound is  $2 \log n$ , obtained by naively intersecting a single-deletion correcting code and a single-transposition correcting code. A code with redundancy  $\log n + O(1)$  was claimed in [7, Section III], but the argument there is flawed. In this work, we determine the optimal redundancy of codes in this setting up to an  $O(\log \log n)$  additive term via a novel marker-based approach. More precisely, we prove the following result, more details of which can be found in Section 4.

► **Theorem 2.** *There exists a binary code  $\mathcal{C} \subseteq \{0, 1\}^n$  correcting one deletion or one transposition with redundancy  $\log n + O(\log \log n)$ .*

Since we know that every code that corrects one deletion also corrects one insertion [13], we also conclude from Theorem 2 that there exists a binary code correcting one deletion, one insertion, or one transposition with nearly optimal redundancy  $\log n + O(\log \log n)$ .

## 1.3 Binary codes for one deletion *and* one substitution

To conclude, we make progress on the study of single-deletion single-substitution correcting codes. Recent work by Smagloy, Welter, Wachter-Zeh, and Yaakobi [19] constructed efficiently encodable and decodable binary single-deletion single-substitution correcting codes with redundancy close to  $6 \log n$ . On the other hand, it is known that  $2 \log n$  redundant bits are required, and a greedy approach shows the *existence* of a single-deletion single-substitution correcting code with redundancy  $4 \log n + O(1)$ .

In this setting, we ask what improvements are possible if we relax the unique decoding requirement slightly and instead require that the code be *list-decodable with list-size 2*. There, our goal is to design a low-redundancy code  $\mathcal{C} \subseteq \{0, 1\}^n$  such that for any corrupted string  $y \in \{0, 1\}^{n-1} \cup \{0, 1\}^n$  there are at most two codewords  $x, x' \in \mathcal{C}$  that can be transformed into  $y$  via some combination of at most one deletion and one substitution. This is the strongest possible requirement after unique decoding, which corresponds to lists of size 1.

The best known *existential* upper bound on the optimal redundancy in the list-decoding setting is still  $4 \log n + O(1)$  via the Gilbert-Varshamov-type greedy algorithm. We give an explicit list-decodable code with list-size 2 correcting one deletion and one substitution with redundancy matching the existential bound up to an  $O(\log \log n)$  additive term. At a high level, this code is obtained by combining the standard VT sketch (2) with *run-based sketches*, which have been recently used in the design of two-deletion correcting codes [9]. More precisely, we have the following result, details of which can be found in Section 5.

► **Theorem 3.** *There exists a linear-time encodable and decodable binary list-size 2 single-deletion single-substitution correcting code  $\mathcal{C} \subseteq \{0, 1\}^n$  with  $4 \log n + O(\log \log n)$  bits of redundancy.*

Subsequently to the appearance of our work online, Song, Cai, and Nguyen [20] constructed a list-decodable code with list-size 2 for one deletion and one substitution with redundancy  $3 \log n + O(\log \log n)$ .

## 1.4 Related work

Recently, there has been a flurry of works making progress in coding-theoretic questions analogous to the ones we consider here in other extensions of the binary single-edit error setting. A line of work culminating in [1, 9, 17] has succeeded in constructing explicit low-redundancy codes correcting a constant number of worst-case deletions. Constructions focused on the two-deletion case have also been given, e.g., in [17, 6, 9]. Explicit binary codes correcting a sublinear number of edit errors with redundancy optimal up to a constant factor have also been constructed recently [3, 10]. Other works have considered the related setting where one wishes to correct a burst of deletions or insertions [15, 12, 26], or a combination of duplications and edit errors [23]. Following up on [19], codes correcting a combination of more than one deletion and one substitution were given in [21] with sub-optimal redundancy. List-decodable codes in settings with indel errors have also been considered before. For example, Wachter-Zeh [25] and Guruswami, Haeupler, and Shahrasbi [8] study list-decodability from a linear fraction of deletions and insertions.

Most relevant to our result in Section 1.3, Guruswami and Håstad [9] constructed an explicit list-size two code correcting two deletions with redundancy  $3 \log n + O(\log \log n)$ , thus beating the greedy existential bound in this setting.

With respect to the interplay between deletions and transpositions, Gabrys, Yaakobi, and Milenkovic [7] constructed codes correcting a single deletion *and* many adjacent transpositions. In an incomparable regime, Schulman and Zuckerman [16], Cheng, Jin, Li, and Wu [4], and Haeupler and Shahrasbi [11] constructed explicit codes with good redundancy correcting a linear fraction of deletions and insertions and a nearly-linear fraction of transpositions.

## 2 Preliminaries

### 2.1 Notation and conventions

We denote sets by uppercase letters such as  $S$  and  $T$  or uppercase calligraphic letters such as  $\mathcal{C}$ , and define  $[n] = \{0, 1, \dots, n-1\}$ ,  $S^{\leq k} = \bigcup_{i=0}^k S^i$ , and  $S^* = \bigcup_{i=0}^{\infty} S^i$  for any set  $S$ . The symmetric difference between two sets  $S$  and  $T$  is denoted by  $S \Delta T$ . We use the notation

$\{\{a, a, b\}\}$  for multisets, which may contain several copies of each element. Given two strings  $x$  and  $y$  over a common alphabet  $\Sigma$ , we denote their concatenation by  $x||y$  and write  $x[i : j] = (x_i, x_{i+1}, \dots, x_j)$ . We say  $y \in \Sigma^k$  is a  $k$ -subsequence of  $x \in \Sigma^n$  if there are  $k$  indices  $1 \leq i_1 < i_2 < \dots < i_k \leq n$  such that  $x_{i_j} = y_j$  for  $j = 1, \dots, k$ , in which case we also call  $x$  an  $n$ -supersequence of  $y$ . Moreover, we say  $x[i : j]$  is an  $a$ -run of  $x$  if  $x[i : j] = a^{j-i+1}$  for a symbol  $a \in \Sigma$ . We denote the base-2 logarithm by  $\log$ . A length- $n$  code  $\mathcal{C}$  is a subset of  $\Sigma^n$  for some alphabet  $\Sigma$  which will be clear from context. In this work, we are interested in the *redundancy* of certain codes (measured in bits), which we define as  $n \log |\Sigma| - \log |\mathcal{C}|$ .

## 2.2 Error models and codes

Since we will be dealing with three distinct but related models of worst-case errors, we begin by defining the relevant standard concepts in a more general way. We may define a worst-case error model over some alphabet  $\Sigma$  by specifying a family of *error balls*  $\mathcal{B} = \{B(y) \subseteq \Sigma^* : y \in \Sigma^*\}$ , where the  $B(y)$  can be arbitrary sets. Usually,  $B(y)$  contains all strings that can be corrupted into  $y$  by applying an allowed error pattern. We proceed to define unique decodability of a code  $\mathcal{C} \subseteq \Sigma^n$  with respect to an error model.

► **Definition 4** (Uniquely decodable code). *We say a code  $\mathcal{C} \subseteq \Sigma^n$  is uniquely decodable (with respect to  $\mathcal{B}$ ) if  $|B(y) \cap \mathcal{C}| \leq 1$  for all  $y \in \Sigma^*$ .*

Throughout this work the underlying error model will always be clear from context, so we do not mention it explicitly. We will also consider *list-decodable* codes with small list size in Section 5, and so we require the following more general definition.

► **Definition 5** (List-size  $t$  decodable code). *We say a code  $\mathcal{C} \subseteq \Sigma^n$  is list-size  $t$  decodable (with respect to  $\mathcal{B}$ ) if  $|B(y) \cap \mathcal{C}| \leq t$  for all  $y \in \Sigma^*$ .*

Note that uniquely decodable codes correspond exactly to list-size 1 codes. Moreover, we remark that for the error models considered in this work and constant  $t$ , the best existential bound for list-size  $t$  codes coincides with the best existential bound for uniquely decodable codes up to a constant additive term.

We proceed to describe the type of errors we consider. A deletion transforms a string  $x \in \Sigma^n$  into one of its  $(n - 1)$ -subsequences. An insertion transforms a string  $x \in \Sigma^n$  into one of its  $(n + 1)$ -supersequences. A substitution transforms  $x \in \Sigma^n$  into a string  $x' \in \Sigma^n$  that differs from  $x$  in exactly one coordinate. An adjacent transposition transforms strings of the form  $ab$  into  $ba$ . More formally, a string  $x \in \Sigma^n$  is transformed into a string  $x' \in \Sigma^n$  with the property that  $x'_k = x_{k+1}$  and  $x'_{k+1} = x_k$  for some  $k$ , and  $x'_i = x_i$  for  $i \neq k, k + 1$ .

We can now instantiate the above general definitions under the specific error models considered in this paper. In the case of a single edit,  $B(y)$  contains all strings which can be transformed into  $y$  via at most one deletion, one insertion, or one substitution. In the case of one deletion *and* one substitution,  $B(y)$  contains all strings that can be transformed into  $y$  by applying at most one deletion and at most one substitution. Finally, in the case of one deletion or one adjacent transposition,  $B(y)$  contains all strings that can be transformed into  $y$  by applying either at most one deletion or at most one transposition.

## 3 Non-binary single-edit correcting codes

In this section, we describe and analyze the code construction used to prove Theorem 1. Before we do so, we provide some intuition behind our approach.

### 3.1 The binary alphabet case as a motivating example

It is instructive to start off with the binary alphabet case and the VT code described in (1), which motivates our approach for non-binary alphabets. More concretely, we may wonder whether a direct generalization of  $\mathcal{C}$  to larger alphabets also corrects a single edit error, say

$$\mathcal{C}' = \left\{ x \in [q]^n \mid \sum_{i=1}^n ix_i = s \pmod{(1+2qn)}, \quad \forall c \in [q] : |\{i : x_i = c\}| = s_c \pmod{2} \right\},$$

where  $[q] = \{0, 1, \dots, q-1\}$  and  $s, s_0, \dots, s_{q-1}$  are appropriately chosen integers. However, this approach fails already over a ternary alphabet  $\{0, 1, 2\}$ . In fact,  $\mathcal{C}'$  cannot correct worst-case deletions of 1's because it does not allow us to distinguish between  $\dots \underline{1}02\dots$  and  $\dots 02\underline{1}\dots$ , which can be obtained one from the other by deleting and inserting a 1 in the underlined positions. More generally, there exist codewords  $x \in \mathcal{C}'$  with substrings  $(x_j = 1, x_{j+1}, \dots, x_k)$  not consisting solely of 1's satisfying

$$\sum_{i=j+1}^k (x_i - 1) = 0. \quad (3)$$

This is problematic since the string  $x'$  obtained by deleting  $x_j = 1$  from  $x$  and inserting a 1 between  $x_k$  and  $x_{k+1}$  is also in  $\mathcal{C}'$ . In order to avoid the problem encountered by  $\mathcal{C}'$ , we instead consider a *weighted VT sketch* of the form

$$f_w(x) = \sum_{i=1}^n i \cdot w(x_i) \pmod{N} \quad (4)$$

for some weight function  $w : [q] \rightarrow \mathbb{Z}$  and an appropriate modulus  $N$ . Using  $f_w$  instead of the standard VT sketch  $f(x) = \sum_{i=1}^n ix_i \pmod{N}$  in the argument above causes the condition (3) for an uncorrectable 1-deletion to be replaced by  $\sum_{i=j+1}^k (w(x_i) - w(1)) = 0$ . Then, choosing  $0 \leq w(0) < w(1) < w(2) < \dots < w(q-1)$  appropriately allows us to correct the deletion of a 1 in  $x$  given knowledge of  $f_w(x)$  provided that  $x$  satisfies a simple runlength constraint. In turn, encoding an arbitrary message  $z$  into a string  $x$  satisfying this constraint can be done very efficiently via a direct application of the simple *runlength replacement* technique from [15] using few redundant bits. Theorem 1 is then obtained by instantiating the weighted VT sketch (4) with an appropriate weight function and modulus.

### 3.2 Code construction

In this section, we present our construction of a 4-ary single-edit correcting code which leads to Theorem 1. As discussed in Section 3.1, given an arbitrary string  $x \in \{0, 1, 2, 3\}^n$  we consider a weighted VT sketch

$$f(x) = \sum_{i=1}^n i \cdot w(x_i) \pmod{[1 + 2n \cdot (2 \log n + 12)]},$$

where  $w(0) = 0$ ,  $w(1) = 1$ ,  $w(2) = 2 \log n + 11$ , and  $w(3) = 2 \log n + 12$ , along with the count sketches  $h_c(x) = |\{i : x_i = c\}| \pmod{2}$  for  $c \in \{0, 1, 2\}$ . Intuitively, the count sketches allow us to cheaply narrow down exactly what type of deletion or substitution occurred (but not its position). As we shall prove later on, successfully correcting the deletion of an  $a$  boils down to ensuring that

$$\sum_{i=j}^k (w(x_i) - w(a)) \neq 0 \quad (5)$$

for all  $1 \leq j \leq k \leq n$  such that there is  $i \in [j, k]$  with  $x_i \neq a$ . We call strings  $x$  that satisfy this property for every  $a$  *regular*, and proceed to show that enforcing a simple runlength constraint on  $x$  is sufficient to guarantee that it is regular.

► **Lemma 6.** *Suppose  $x \in \{0, 1, 2, 3\}^n$  satisfies the following property: If  $x'$  denotes the subsequence of  $x$  obtained by deleting all 1's and 3's and  $x''$  denotes the subsequence obtained by deleting all 0's and 2's, it holds that all 0-runs of  $x'$  and all 3-runs of  $x''$  have length at most  $\log n + 3$ . Then,  $x$  is regular.*

**Proof.** See the full version [5]. ◀

Let  $\mathcal{G} \subseteq \{0, 1, 2, 3\}^n$  denote the set of regular strings. Given the above definitions, we set our code to be

$$\mathcal{C} = \mathcal{G} \cap \{x \in \{0, 1, 2, 3\}^n : f(x) = s, h_c(x) = s_c, c \in \{0, 1, 2\}\} \quad (6)$$

for appropriate choices of  $s \in \{0, \dots, 1 + 2n \cdot (2 \log n + 12)\}$  and  $s_c \in \{0, 1\}$  for  $c = 0, 1, 2$ . A straightforward application of the probabilistic method shows that most strings are regular.

► **Lemma 7.** *Let  $X$  be sampled uniformly at random from  $\{0, 1, 2, 3\}^n$ . Then, we have  $\Pr[X \text{ is regular}] \geq 7/8$ .*

As a result, by the pigeonhole principle there exist choices of  $s, s_0, s_1, s_2$  such that

$$|\mathcal{C}| \geq \frac{7 \cdot 4^n}{8 \cdot 2^3 \cdot (1 + 2n \cdot (2 \log n + 12))}.$$

This implies that we can make it so that  $\mathcal{C}$  has  $\log n + \log \log n + 6 + o(1)$  bits of redundancy, where  $o(1) \rightarrow 0$  when  $n \rightarrow \infty$ , as desired. If  $n$  is not a power of two, then taking ceilings yields at most one extra bit of redundancy for a total of  $\log n + \log \log n + 7 + o(1)$  bits, as claimed.

It remains to show that  $\mathcal{C}$  corrects a single edit in linear time and that a standard modification of  $\mathcal{C}$  admits a linear time encoder. Observe that if a codeword  $x \in \mathcal{C}$  is corrupted into a string  $y$  by a single edit error, we can tell whether it was a deletion, insertion, or substitution by computing  $|y|$ . Therefore, we treat each such case separately. Since correcting one substitution in our code is analogous to correcting one substitution in the original binary VT code, and since correcting one insertion is similar to correcting one deletion, we consider only the case of one deletion here and leave the remaining cases to the full version [5].

### 3.3 Correcting one deletion

Suppose that  $y$  is obtained from  $x \in \mathcal{C}$  by deleting an  $a$  at position  $i$ . First, note that we can find  $a$  by computing  $h_c(y) - h_c(x)$  for  $c = 0, 1, 2$ . Now, let  $y^{(j)}$  denote the string obtained by inserting an  $a$  to the left of  $y_j$  (when  $j = n$  this means we insert an  $a$  at the end of  $y$ ). We have  $x = y^{(i)}$  and our goal is to find  $i$ . Consider  $n \geq j \geq i$  and observe that

$$f(x) - f(y^{(j)}) = f(y^{(i)}) - f(y^{(j)}) = \sum_{\ell=i+1}^j (w(x_\ell) - w(a)),$$

because  $y_{\ell-1} = x_\ell$  for  $\ell > i$ . Since  $x$  is regular, it follows that  $\sum_{\ell=i+1}^j (w(x_\ell) - w(a)) \neq 0$  unless  $x_{i+1} = \dots = x_j = a$ . This suggests the following decoding algorithm: Successively compute  $f(x) - f(y^{(j)})$  for  $j = n, n-1, \dots, 1$  until  $f(x) - f(y^{(j)}) = 0$ , in which case the above argument ensures that  $y^{(j)} = x$  since we must be inserting  $a$  into the same  $a$ -run of  $x$  from which an  $a$  was deleted. This procedure runs in overall time  $O(n)$ , since we can compute  $f(x) - f(y^{(j-1)})$  given  $f(x) - f(y^{(j)})$  with  $O(1)$  operations.

### 3.4 A linear-time encoder

We have described a linear-time decoder that corrects a single edit error in regular strings  $x$  assuming knowledge of the weighted VT sketch  $f(x)$  and the count sketches  $h_c(x)$  for  $c = 0, 1, 2$ . It remains to describe a low-redundancy linear-time encoding procedure for a slightly modified version of our code  $\mathcal{C}$  defined in (6). Fix an arbitrary message  $z \in \{0, 1, 2, 3\}^m$ . We proceed in two steps:

1. We encode  $z$  into a regular string  $x \in \{0, 1, 2, 3\}^{m+4}$  in linear time by exploiting the runlength replacement technique from [15];
2. We append an appropriate encoding of the sketches (which we now see as binary strings) to  $x$  that can be recovered even if the final string is corrupted by an edit error. This adds only  $O(\log \log n)$  bits of redundancy, and allows  $x$  (and thus  $z$ ) to be recovered in linear time.

The complete analysis can be found in the full version [5].

## 4 Binary codes correcting one deletion or one transposition

In this section, we describe and analyze the code construction used to prove Theorem 2. As discussed in Section 1.2, the adjacent transposition precludes the use of the standard VT sketch. Therefore, we undertake a radically different approach.

### 4.1 Code construction and high-level overview of our approach

Our starting point is a marker-based segmentation approach considered by Lenz and Polyanskii [12] to correct bursts of deletions. We then introduce several new ideas. Roughly speaking, our idea is to partition a string  $x \in \{0, 1\}^n$  into consecutive short substrings  $z_1^x, \dots, z_\ell^x$  for some  $\ell$  according to the occurrences of a special marker string in  $x$ . Then, by carefully embedding hashes of each segment  $z_i^x$  into a VT-type sketch, adding information about the *multiset* of hashes, and exploiting specific structural properties of deletions and adjacent transpositions, we are able to determine a short interval containing the position where the error occurred. Once this is done, a standard technique allows us to recover the true position of the error by slightly increasing the redundancy.

We now describe the code construction in detail. For a given integer  $n > 0$ , let  $\Delta = 50 + 1000 \log n$  and  $m = 1000\Delta^2 = O(\log^2 n)$ . For the sake of readability, we have made no efforts to optimize constants, and assume  $n$  is a power of two to avoid using ceilings and floors. Given a string  $x \in \{0, 1\}^n$ , we divide it into substrings split according to occurrences of the marker 0011. To avoid edge cases, assume that  $x$  ends in 0011 – this will only add 4 bits to the overall redundancy. Then, this marker-based segmentation induces a vector  $z^x = (z_1^x, \dots, z_{\ell_x}^x)$ , where  $1 \leq \ell_x \leq n$ , and each string  $z_i^x$  has length at least 4, ends with 0011, and 0011 only occurs once in each such string. We may assume that  $|z_i^x| \leq \Delta$  for all  $i$ . This will only add 1 bit to the overall redundancy, as captured in the following simple lemma.

► **Lemma 8.** *Suppose  $X$  is uniformly random over  $\{0, 1\}^n$ . Then,  $\Pr[|z_i^X| \leq \Delta, i = 1, \dots, \ell_X] \geq \frac{1}{2}$ .*

Our goal now will be to impose constraints on  $z^x$  so that (i) We only introduce  $\log n + O(\log \log n)$  bits of redundancy, and (ii) If  $x$  is corrupted by a deletion or transposition in  $z_i^x$ , we can then locate a window  $W \subseteq [n]$  of size  $|W| = O(\log^4 n)$  such that  $z_i^x \subseteq W$ . This will then allow us to correct the error later on by adding  $O(\log \log n)$  bits of redundancy.

Since each  $z_i^x$  has length at most  $\Delta = O(\log n)$ , we will exploit the fact that there exists a hash function  $h$  with short output that allows us to correct a deletion, substitution, or transposition in all strings of length at most  $3\Delta$ . This is guaranteed by the following lemma.

► **Lemma 9.** *There exists a hash function  $h : \{0, 1\}^{\leq 3\Delta} \rightarrow [m]$  with the following property: If  $z'$  is obtained from  $z$  by at most two transpositions, two substitutions, or at most a deletion and an insertion, then  $h(z) \neq h(z')$ .*

**Proof.** We can construct such a hash function  $h$  greedily. Let  $A(z)$  denote the set of such strings obtained from  $z \in \{0, 1\}^{\leq 3\Delta}$ . Since  $|A(z)| < m$ , we can set  $h(z)$  so that  $h(z) \neq h(z')$  for all  $z' \in A(z) \setminus \{z\}$ . ◀

With the intuition above and the hash function  $h$  guaranteed by Lemma 9 in mind, we consider the VT-type sketch

$$f(x) = \sum_{j=1}^{\ell_x} j(|z_j^x| \cdot m + h(z_j^x)) \mod (L = 10n \cdot \Delta \cdot m + 1)$$

along with the count sketches  $g_1(x) = \ell_x \mod 5$  and  $g_2(x) = \sum_{i=1}^n \bar{x}_i \mod 3$ , where  $\bar{x}_i = \sum_{j=1}^i x_j \mod 2$ . At a high level, the sketch  $f(x)$  is the main tool we use to approximately locate the error in  $x$ . The count sketches  $g_1(x)$  and  $g_2(x)$  are added to allow us to detect how many markers are created or destroyed by the error, and to distinguish between the cases where there is no error or a transposition occurs. Thus, we define the preliminary code

$$\mathcal{C}' = \left\{ x \in \{0, 1\}^n \mid \begin{array}{l} x[n-3, n] = (0, 0, 1, 1), f(x) = s_0, g_1(x) = s_1, g_2(x) = s_2, \\ \forall i \in [\ell_x] : |z_i^x| \leq \Delta \end{array} \right\}$$

for appropriate choices of  $s_0, s_1, s_2$ . Taking into account all constraints, the choice of  $\Delta$  and  $m$ , and Lemma 8, the pigeonhole principle implies that we can choose  $s_0, s_1, s_2$  so that this code has at most  $4 + \log(10n \cdot \Delta \cdot m + 1) + 1 + 2 + 2 + 1 = \log n + O(\log \log n)$  bits of redundancy.

However, it turns out that the constraints imposed in  $\mathcal{C}'$  are not enough to handle a deletion or a transposition. Intuitively, the reason for this is that, in order to make use of the sketch  $f(x)$  when decoding, we will need additional information both about the hashes of the segments of  $x$  that were affected by the error and the hashes of the corresponding corrupted segments in the corrupted string  $y$ . Therefore, given a vector  $z^x$  and the hash function  $h$  guaranteed by Lemma 9, we will be interested in the associated *hash multiset*  $H_x = \{\{h(z_1^x), \dots, h(z_{\ell_x}^x)\}\}$  over  $[m]$ . As we shall see, a deletion or transposition will change this multiset by at most 4 elements. Therefore, we will expurgate  $\mathcal{C}'$  so that any pair of remaining codewords  $x$  and  $x'$  satisfy either  $H_x = H_{x'}$  or  $|H_x \triangle H_{x'}| \geq 10$ . This will allow us to recover the true hash multiset of  $x$  from the hash multiset of the corrupted string. The following lemma shows that this expurgation adds only an extra  $O(\log m) = O(\log \log n)$  bits of redundancy.

► **Lemma 10.** *There exists a code  $\mathcal{C} \subseteq \mathcal{C}'$  of size  $|\mathcal{C}| \geq \frac{|\mathcal{C}'|}{m^{10}}$  such that for any  $x, x' \in \mathcal{C}$  we either have  $H_x = H_{x'}$  or  $|H_x \triangle H_{x'}| \geq 10$ .*

We will take our *error-locating* code to be the expurgated code  $\mathcal{C}$  guaranteed by Lemma 10. By the redundancy of  $\mathcal{C}'$  above and the choice of  $m$ , it follows that there exists a choice of  $s_0$  and  $s_1$  such that  $\mathcal{C}$  has  $\log n + O(\log \log n)$  bits of redundancy. We prove the following result, which states that, given a corrupted version of  $x \in \mathcal{C}$ , we can identify a small interval containing the position where the error occurred.



► **Theorem 11.** *If  $x \in \mathcal{C}$  is corrupted into  $y$  via one deletion or transposition, we can recover from  $y$  a window  $W \subseteq [n]$  of size  $|W| \leq 10^{10} \log^4 n$  that contains the position where the error occurred (in the case of a transposition, we take the error location to be the smallest of the two affected indices).*

We can use Theorem 11 to prove our main Theorem 2 via standard methods (see the full version [5]).

Fix  $x \in \mathcal{C}$  and suppose  $y$  is obtained from  $x$  via one deletion or one transposition. To prove Theorem 11, we consider several independent cases based on the fact that a marker cannot overlap with itself, that we can identify whether a deletion occurred by computing  $|y|$ , and that we can identify whether a transposition occurred by comparing  $g_2(x)$  and  $g_2(y)$ . Since the arguments are similar, we show how to locate one deletion and leave the case of one adjacent transposition to the full version [5].

## 4.2 Locating one deletion

In this section, we show how we can locate one deletion appropriately. Fix  $x \in \mathcal{C}$  and suppose that a deletion is applied to  $z_i^x$ . The following lemma holds due to the marker structure.

► **Lemma 12.** *A deletion either (i) Creates a new marker and does not delete any existing markers, in which case  $\ell_y = \ell_x + 1$ , (ii) Deletes an existing marker and does not create any new markers, in which case  $\ell_y = \ell_x - 1$ , or (iii) Neither deletes existing markers nor creates new markers, in which case  $\ell_y = \ell_x$ .*

Note that we can distinguish between the cases detailed in Lemma 12 by comparing  $g_1(x)$  and  $g_1(y)$ . Thus, we analyze each case separately:

1.  $\ell_y = \ell_x$ : In this case, we have  $z^y = (z_1^x, \dots, z_{i-1}^x, z'_i, z_{i+1}^x, \dots, z_{\ell_x}^x)$ , where  $z'_i$  is obtained from  $z_i^x$  by a deletion (in particular,  $|z'_i| = |z_i^x| - 1$ ). Therefore, it holds that

$$\begin{aligned} f(x) - f(y) &= \sum_{j=1}^{\ell_x} j(|z_j^x| \cdot m + h(z_j^x)) - \sum_{j=1}^{\ell_y} j(|z_j^y| \cdot m + h(z_j^y)) \pmod L \\ &= i(|z_i^x| \cdot m + h(z_i^x) - |z'_i| \cdot m - h(z'_i)) \\ &= i(m + h(z_i^x) - h(z'_i)), \end{aligned}$$

where the second equality uses  $\ell_y = \ell_x$ . Let  $H_y$  denote the hash multiset of  $y$ . Then, we know that  $|H_x \triangle H_y| \leq 2$ . Therefore, we can recover  $H_x$  from  $H_y$ , which means that we can recover  $h(z_i^x) - h(z'_i)$ . Indeed, if  $h(z_i^x) - h(z'_i) = 0$  then  $H_x = H_y$ . On the other hand, if  $h(z_i^x) - h(z'_i) \neq 0$  then  $|H_x \triangle H_y| = 2$  and we recover both  $h(z_i^x)$  (the element in  $H_x$  but not in  $H_y$ ) and  $h(z'_i)$  (the element in  $H_y$  but not in  $H_x$ ). As a result, we know  $m + h(z_i^x) - h(z'_i)$ . Since it also holds that  $m + h(z_i^x) - h(z'_i) \neq 0$  (because  $|h(z_i^x) - h(z'_i)| < m$ ), we can recover  $i$  from  $f(x) - f(y)$ . This gives a window  $W$  of length at most  $\Delta = O(\log n)$ .

2.  $\ell_y = \ell_x - 1$ : In this case, the marker at the end of  $z_i^x$  is destroyed, merging  $z_i^x$  and  $z_{i+1}^x$ . Observe that if  $i = \ell_x$  then we can simply detect that the last marker in  $x$  was destroyed. Therefore, we assume that  $i < \ell_x$ , in which case we have  $z^y = (z_1^x, \dots, z_{i-1}^x, z'_i, z_{i+2}^x, \dots, z_{\ell_x}^x)$ , where  $|z'_i| = |z_i^x| + |z_{i+1}^x| - 1$ . Consequently, it holds that



$$\begin{aligned}
f(x) - f(y) &= \sum_{j=1}^{\ell_x} j(|z_j^x| \cdot m + h(z_j^x)) - \sum_{j=1}^{\ell_y} j(|z_j^y| \cdot m + h(z_j^y)) \pmod L \\
&= i(|z_i^x| \cdot m + h(z_i^x)) + (i+1)(|z_{i+1}^x| \cdot m + h(z_{i+1}^x)) - i(|z_i^y| \cdot m + h(z_i^y)) \\
&\quad + \sum_{j=i+2}^{\ell_x} (|z_j^x| \cdot m + h(z_j^x)) \\
&= \sum_{j=i+2}^{\ell_x} (|z_j^x| \cdot m + h(z_j^x)) + i(m + h(z_i^x) + h(z_{i+1}^x) - h(z_i^y)) \\
&\quad + (|z_{i+1}^x| \cdot m + h(z_{i+1}^x)).
\end{aligned}$$

Note that, since  $|H_x \triangle H(y)| \leq 3$ , we can recover  $H_x$  from  $H_y$ . In particular, this means that we know  $h(z_i^x) + h(z_{i+1}^x) - h(z_i^y)$ . Therefore, for  $i' = \ell_y - 1, \ell_y - 2, \dots, i$  we can compute the *potential function*

$$\begin{aligned}
\Phi(i') &= \sum_{j=i'+1}^{\ell_y} (|z_j^y| \cdot m + h(z_j^y)) + i'(m + h(z_i^x) + h(z_{i+1}^x) - h(z_i^y)) \\
&= \sum_{j=i'+2}^{\ell_x} (|z_j^x| \cdot m + h(z_j^x)) + i'(m + h(z_i^x) + h(z_{i+1}^x) - h(z_i^y)).
\end{aligned}$$

Note that

$$|\Phi(i) - (f(x) - f(y))| = ||z_{i+1}^x| \cdot m + h(z_{i+1}^x)| \leq \Delta \cdot m + m \leq 10^7 \log^2 n. \quad (7)$$

Moreover, we also have

$$\begin{aligned}
\Phi(i' - 1) - \Phi(i') &= |z_{i'+1}^x| \cdot m + h(z_{i'+1}^x) - (m + h(z_i^x) + h(z_{i+1}^x) - h(z_i^y)) \\
&\geq 4m - 3m = m. \quad (8)
\end{aligned}$$

This suggests the following procedure for recovering the window  $W$ . Sequentially compute  $\Phi(i')$  for  $i'$  starting at  $\ell_y - 1$  until we find  $i^* \geq i$  such that  $|\Phi(i') - (f(x) - f(y))| \leq 10^6 \log^2 n$ . This is guaranteed to exist since  $i' = i$  satisfies this property. We claim that  $i^* - i \leq 10^7 \log n$ . In fact, if this is not the case then the monotonicity property in (8) implies that  $|\Phi(i) - (f(x) - f(y))| > m \cdot 10^7 \log n > 10^7 \log^2 n$ , contradicting (7). Since  $|z_j^x| \leq \Delta$  for every  $j$ , recovering  $i^*$  also yields a window  $W \subseteq [n]$  of size  $|W| = 10^6 \log n \cdot \Delta = 10^9 \log^2 n$  containing the error position, as desired.

3.  $\ell_y = \ell_x + 1$ : This case is very similar to the previous one (see the full version [5]).

## 5 Binary list-size two code for one deletion and one substitution

In this section, we describe and analyze a binary list-size two decodable code for one deletion and one substitution, which yields Theorem 3. Departing from the approach of [19], our construction makes use of *run-based sketches* combined with the standard VT sketch. Run-based sketches have thus far been exploited in the construction of multiple-deletion correcting codes, including list-decodable codes with small list size [9].

### 5.1 Code construction

We begin by describing some required concepts: Given a string  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , we define its *run string*  $r^x$  by first setting  $r_0^x = 0$  along with  $x_0 = 0$  and  $x_{n+1} = 1$ , and then iteratively computing  $r_i^x = r_{i-1}^x$  if  $x_i = x_{i-1}$  and  $r_i^x = r_{i-1}^x + 1$  otherwise for  $i = 1, \dots, n, n+1$ . Note that every string  $x$  is uniquely determined by its run string  $r^x$  and vice-versa. Moreover, it holds that  $r^x$  defines a non-decreasing sequence and  $0 \leq r_i^x \leq i$  for every  $i = 1, \dots, n, n+1$ . As an example, the run string corresponding to  $x = 011101000$  is  $r^x = 0111234445$ . We call  $r_i^x$  the *rank* of index  $i$  in  $x$ . We will denote the total number of runs in  $x$  by  $r(x)$ .

The main component of our code is a combination of the standard VT sketch

$$f(x) = \sum_{i=1}^n ix_i \mod (3n+1) \quad (9)$$

with the run-based sketches

$$f_1^r(x) = \sum_{i=1}^n r_i^x \mod (12n+1), \quad (10)$$

$$f_2^r(x) = \sum_{i=1}^n r_i^x(r_i^x - 1) \mod (16n^2 + 1) \quad (11)$$

originally considered in [9]. Additionally, we also consider the count sketches

$$h(x) = \sum_{i=1}^n x_i \mod 5 \quad \text{and} \quad h_r(x) = r(x) \mod 13. \quad (12)$$

Intuitively, the count sketches are used to distinguish different error patterns. The sketch  $h(x)$  is used to determine the value of the bit deleted and the value of the bit flipped, while  $h_r(x)$  is used to identify how the number of runs was affected by the errors. For each possible error pattern, we use the standard VT-sketch and the run-based sketches to decode. Given the above, our code is defined to be

$$\mathcal{C} = \{x \in \{0, 1\}^n : f(x) = s, f_1^r(x) = s_1^r, f_2^r(x) = s_2^r, h(x) = u, h_r(x) = u_r\}, \quad (13)$$

for an appropriate choice of  $s \in [3n+1]$ ,  $s_1^r \in [12n+1]$ ,  $s_2^r \in [16n^2+1]$ ,  $u \in [5]$ , and  $u_r \in [13]$ . By the pigeonhole principle, there is such a choice which ensures  $\mathcal{C}$  has redundancy  $4 \log n + O(1)$ .

In the remainder of this section, we first provide a high-level overview of our approach towards showing that  $\mathcal{C}$  admits linear-time list-decoding from one deletion and one substitution with list-size 2. Then, we analyze a special case which exemplifies our more general approach. The remainder of our argument appears in the full version [5]. We remark that linear-time decoding and encoding of a slightly modified version of  $\mathcal{C}$  (which has redundancy  $4 \log n + O(\log \log n)$  instead) follow without difficulty from this analysis via standard methods. These algorithms are presented and analyzed in the full version [5].

### 5.2 High-level overview of our approach

Fix  $x \in \mathcal{C}$ , and let  $y$  be the string obtained from  $x$  after one deletion at index  $d$  and one substitution at index  $e$ . We use  $x_e$  to denote the bit flipped, and  $x_d$  to denote the bit deleted in  $x$ . When  $d = e$ , we have one deletion and no substitution. Our goal is to recover  $x$  from  $y$ .

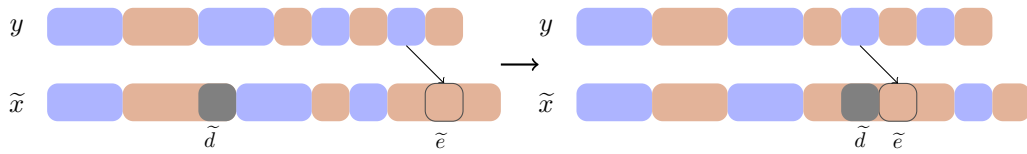
We begin with some simple but useful remarks. First, we observe that one deletion and no substitution can be equivalently transformed to one deletion and one substitution. Thus, we will only consider the case in which we have one deletion and one substitution, i.e.,  $d \neq e$ . We present a proof of this fact in the full version [5]. Second, the following structural lemma about the number of runs in a corrupted string will prove useful in our case analysis.

► **Lemma 13.** *If  $x'$  is obtained from  $x$  via one deletion, then either  $r(x') = r(x)$  or  $r(x') = r(x) - 2$ . On the other hand, if  $x'$  is obtained from  $x$  via one substitution, then either  $r(x') = r(x)$ ,  $r(x') = r(x) - 2$ , or  $r(x') = r(x) + 2$ .*

Combining Lemma 13 with the count sketches  $h(x)$  and  $h_r(x)$  and knowledge of  $y$  ensures that we can identify not only the values of  $x_d$  and  $x_e$ , but also  $r(x)$ . As a result, this allows us to split our analysis into several independent cases.

The process of decoding can be thought of as inserting a bit  $x_d$  before the  $d$ -th bit in  $y$  and flipping the  $(e - \delta)$ -th bit in  $y$ , where  $\delta \in \{0, 1\}$  is the indicator variable of whether  $e > d$ . Our goal is to find  $d$  and  $e$ . We will begin with a candidate position pair  $(\tilde{d}, \tilde{e})$  with  $\tilde{d}$  as small as possible with the property that, if  $\tilde{x}$  denotes the string obtained from  $y$  by inserting  $x_d$  before the  $\tilde{d}$ -th bit in  $y$  and flipping the bit at position  $\tilde{e} - \delta$  in  $y$ , where  $\delta$  indicates whether  $\tilde{d} < \tilde{e}$ , then  $f(\tilde{x}) = f(x)$ ,  $h_r(\tilde{x}) = h_r(x)$ , and  $h_r(x') = h_r(\tilde{x}')$ , where  $x'$  (resp.  $\tilde{x}'$ ) denotes the string obtained from  $x$  (resp.  $\tilde{x}$ ) by deleting  $x_d$  (resp.  $\tilde{x}_{\tilde{d}}$ ). We call such pairs *valid*. Intuitively, valid pairs are indistinguishable from the true error pattern  $(d, e)$  from the perspective of the VT sketch and the count sketches, and there may be several of them. However, crucially, many are ruled out via the run-based sketches. Note that the true error pattern  $(d, e)$  is a valid pair, so such pairs always exist.

Roughly speaking, our strategy is to start with some valid pair  $(\tilde{d}, \tilde{e})$  and sequentially move to the *next* valid pair. This is done by moving  $\tilde{d}$  one index to the right and checking whether the *unique* index  $\tilde{e}$  that ensures  $f(\tilde{x}) = f(x)$  forms a valid pair  $(\tilde{d}, \tilde{e})$ . If this does not hold, then we move  $\tilde{d}$  one more index to the right, and repeat the process. We call this an *elementary move*. Note that since inserting a bit  $b$  into a  $b$ -run at any position gives the same output, we may always move  $\tilde{d}$  to the end of the next  $x_d$ -run in  $y$  (which may be empty). Figure 1 shows an example of an elementary move.



■ **Figure 1** Example of an elementary move. Suppose that the error pattern indicates that  $x_d = 1$ ,  $x_e = 1$ , and the deletion does not reduce the number of runs while the substitution increases the number of runs by two. The process starts with the left figure in which a bit 1 is inserted at position  $\tilde{d}$ , the end of a 1-run and the bit 1 at position  $\tilde{e} - 1$  is flipped. After an elementary move,  $\tilde{d}$  moves to the end of the next 1-run, and  $\tilde{e}$  moves to the next position that matches the error pattern  $y_{\tilde{e}-\delta+1} = y_{\tilde{e}-\delta-1} = 0$ .

Considering this step-by-step process with elementary moves proves useful because it turns out to be feasible to track how the different sketches change in each such move. In particular, the following equations will be useful to determine how  $\tilde{d}$  and  $\tilde{e}$  change in each elementary move. Recall that we regard  $y$  as a string obtained via one substitution at index  $e - \delta$  from  $x' \in \{0, 1\}^{n-1}$ , where  $x'$  is obtained via one deletion from  $x$  at index  $d$ . Note that

$$f(x) - f(x') = dx_d + \sum_{i=d}^{n-1} x'_i \quad \text{and} \quad f(x') - f(y) = (e - \delta)[x_e - (1 - x_e)].$$

Moreover, we have  $\sum_{i=d}^{n-1} x'_i = \sum_{i=d}^{n-1} y_i + \delta(2x_e - 1)$ . Combining these three observations yields

$$f(x) - f(y) = dx_d + \sum_{i=d}^{n-1} y_i + e(2x_e - 1). \quad (14)$$

We prove that, during this sequential process, either  $f_1^r$  is monotonic and hence rules out all but one valid pair  $(\tilde{d}, \tilde{e})$ , or a convexity-type property of  $f_2^r$ , which implies that it takes on each value at most twice, rules out all but at most two valid pairs. The convexity of  $f_2^r(x)$  is a consequence of the following lemma.

► **Lemma 14** ([9, Lemma 4.1]). *Let  $a_i$  and  $a'_i$  be two sequences of non-negative integers such that  $\sum_{i=1}^n a_i = \sum_{i=1}^n a'_i$  and there is a value  $t$  such that for all  $i$  satisfying  $a_i < a'_i$  it holds that  $a'_i \leq t$ , and for all  $i$  satisfying  $a_i > a'_i$  it holds that  $a'_i \geq t$ . Then, either  $a_i = a'_i$  for all  $i$ , or  $\sum_{i=1}^n a_i(a_i - 1) > \sum_{i=1}^n a'_i(a'_i - 1)$ .*

Finally, we note that, in the high level overview above, we ignored the fact that we do not have access to the intermediate string  $x'$ , but we need to know  $h_r(x')$ . For example, if  $r(y) = r(x)$ , then there is uncertainty about  $h_r(x')$ . In fact, it could be that both errors do not change the number of runs, or that both errors do change the number of runs but these cancel each other out. Since we are aiming for list-size 2 decoding, this is not problematic, and we handle it in the final decoding procedure.

Below, we consider one special case which exemplifies how our high level approach above can be realized. The remaining cases are analyzed in the full version [5].

### 5.3 Special case – Unique decoding when the number of runs increases by two

If  $r(y) = r(x) + 2$ , then it must be that  $r(x) = r(x')$  and  $r(y) = r(x') + 2$ . This means that the deletion does not change the number of runs (and thus occurred in a run of length at least 2 in  $x$ ), while the substitution affects a bit in the middle of a run of length at least 3. In particular, we have  $y_{e-\delta-1} = y_{e-\delta+1} = 1 - y_{e-\delta}$ . In this case, it follows that

$$f_1^r(x) - f_1^r(x') = r_d^x, \quad f_1^r(x') - f_1^r(y) = -(1 + 2(n - e + \delta)).$$

Therefore, for the run-based sketch  $f_1^r(x)$  it holds that

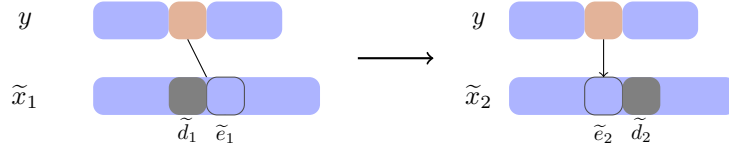
$$f_1^r(x) - f_1^r(y) = r_d^x - (1 + 2(n - e + \delta)). \quad (15)$$

We now proceed by case analysis on the value of  $x_d$  and  $x_e$ .

#### 5.3.1 If $x_e = x_d = b$

In this case, when  $\tilde{d}$  makes an elementary move to the right, it must pass across a  $(1 - b)$ -run of some length  $\ell \geq 1$ . According to (14), position  $\tilde{e}$  has to move to the left by  $\ell$  so that  $f(\tilde{x}) = f(x)$ . If we have  $\tilde{d} < \tilde{e}$  before one elementary move but  $\tilde{d} > \tilde{e}$  after that move, we call it a *take over step*. For each elementary move:

- If the move is not a take over step: Then,  $r_d^{\tilde{x}}$  increases by 2 while  $2(n - \tilde{d} + \tilde{\delta}) + 1$  increases by  $2\ell$ . Therefore, (15) implies that  $f_1^r(\tilde{x})$  strictly decreases after such a move whenever  $\ell > 1$ . If  $\ell = 1$ , then  $\tilde{e}$  moves by 1 to the left and  $f_1^r(\tilde{x})$  remains unchanged. However, since we need  $1 - b = y_{e-\tilde{\delta}} = 1 - y_{e-\tilde{\delta}}$  it follows that  $\tilde{e}$  cannot move only 1 position to the left, and so  $\ell > 1$  necessarily.
- If the move is a take over step: Before the move,  $\tilde{d}$  is on the left of a  $(1 - b)$ -run of length  $\ell \geq 1$  while  $\tilde{e} > \tilde{d}$  satisfies  $y_{e-1} = 1 - b$  and  $y_{e-2} = y_e = b$ . After the move,  $\tilde{d}$  moves to the right of the  $(1 - b)$ -run of length  $\ell$ , while  $\tilde{e}$  is to the left of  $\tilde{d}$ . Moreover, it must be that  $y_e = 1 - b$  and  $y_{e-1} = y_{e+1} = b$ . To match the error pattern, the only possible case is that  $\ell = 1$ . To see why this is the case, note that when  $\ell \geq 2$  the index  $\tilde{e}$  has to move to the left by at least  $\ell + 2$  to match the error pattern  $y_{e-\tilde{\delta}-1} = y_{e-\tilde{\delta}+1} = 1 - y_{e-\tilde{\delta}}$ . However, this move leads to  $f(\tilde{x}) \neq f(x)$ , and thus does not yield a valid pair  $(\tilde{d}, \tilde{e})$ . When  $\ell = 1$ , let  $(\tilde{d}_1, \tilde{e}_1)$  and  $(\tilde{d}_2, \tilde{e}_2)$  denote the position pair before and after the move, respectively. Then, these two pairs yield the same candidate solution  $\tilde{x}_1 = \tilde{x}_2$ . See Figure 2 for an example.



■ **Figure 2** An example of a take over step. If the take over happens, it must be that  $\ell = 1$ . The resulting  $\tilde{x}_1$  and  $\tilde{x}_2$  are the same.

Taking into account both cases above, we see that  $f_1^r(\tilde{x})$  decreases during each elementary move, and decreases by at most  $2n$  during the whole process. Since the value of  $f_1^r(x)$  is taken modulo  $12n + 1$ , there is only a unique pair  $(\tilde{d}, \tilde{e})$  that yields a solution such that  $f_1^r(\tilde{x}) = f_1^r(x)$ . Hence,  $f(x)$  and  $f_1^r(x)$  together with  $y$  uniquely determine one valid pair  $(\tilde{d}, \tilde{e})$ , which in turn yields a unique candidate solution  $\tilde{x} = x$ .

### 5.3.2 If $x_d = 1 - x_e = b$

In this case, when  $\tilde{d}$  makes an elementary move to the right, it must pass across a  $(1 - b)$ -run of some length  $\ell \geq 1$ . Then,  $\tilde{e}$  has to move to the right by  $\ell$  so that  $f(\tilde{x}) = f(x)$ . During each such move  $f_1^r(\tilde{x})$  strictly increases. For the whole process,  $f_1^r(\tilde{x})$  increases by at most  $2n$ . By a similar argument as above, we have that  $f(x)$  and  $f_1^r(x)$  together with  $y$  uniquely determine one valid pair  $(\tilde{d}, \tilde{e})$  which yields the correct solution  $\tilde{x} = x$ .

## 6 Open problems

Our work leaves open several natural avenues for future research. We highlight a few of them here:

- Given the effectiveness of weighted VT sketches in the construction of nearly optimal non-binary single-edit correcting codes in Section 3 with fast encoding and decoding, it would be interesting to find further applications of this notion.
- We believe that the code we introduce and analyze in Section 5 is actually uniquely decodable under one deletion and one substitution. Proving this would be quite interesting, since then we would also have explicit uniquely decodable single-deletion single-substitution correcting codes with redundancy matching the existential bound, analogous to what is known for two-deletion correcting codes [9].

- The code we designed in Section 4 fails to correct an arbitrary substitution. Roughly speaking, the reason behind this is that one substitution may simultaneously destroy and create a marker with a different starting point. As the clear next step, it would be interesting to show the existence of a binary code correcting one *edit* error or one transposition with redundancy  $\log n + O(\log \log n)$ .

---

## References

- 1 Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. *IEEE Transactions on Information Theory*, 64(5):3403–3410, 2018. doi:10.1109/TIT.2017.2746566.
- 2 Kui Cai, Yeow Meng Chee, Ryan Gabrys, Han Mao Kiah, and Tuan Thanh Nguyen. Correcting a single indel/edit for DNA-based data storage: Linear-time encoders and order-optimality. *IEEE Transactions on Information Theory*, 67(6):3438–3451, 2021. doi:10.1109/TIT.2021.3049627.
- 3 Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 200–211, 2018. doi:10.1109/FOCS.2018.00028.
- 4 Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Block edit errors with transpositions: Deterministic document exchange protocols and almost optimal binary codes. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 37:1–37:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.37.
- 5 Ryan Gabrys, Venkatesan Guruswami, João Ribeiro, and Ke Wu. Beyond single-deletion correcting codes: Substitutions and transpositions. *arXiv e-prints*, December 2021. doi:10.48550/arXiv.2112.09971.
- 6 Ryan Gabrys and Frederic Sala. Codes correcting two deletions. *IEEE Transactions on Information Theory*, 65(2):965–974, 2019. doi:10.1109/TIT.2018.2876281.
- 7 Ryan Gabrys, Eitan Yaakobi, and Olgica Milenkovic. Codes in the Damerau distance for deletion and adjacent transposition correction. *IEEE Transactions on Information Theory*, 64(4):2550–2570, 2018. doi:10.1109/TIT.2017.2778143.
- 8 Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrabi. Optimally resilient codes for list-decoding from insertions and deletions. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 524–537, 2020. doi:10.1145/3357713.3384262.
- 9 Venkatesan Guruswami and Johan Håstad. Explicit two-deletion codes with redundancy matching the existential bound. *IEEE Transactions on Information Theory*, 67(10):6384–6394, 2021. doi:10.1109/TIT.2021.3069446.
- 10 Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 334–347, 2019. doi:10.1109/FOCS.2019.00029.
- 11 Bernhard Haeupler and Amirbehshad Shahrabi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 841–854, 2018. doi:10.1145/3188745.3188940.
- 12 Andreas Lenz and Nikita Polyanskii. Optimal codes correcting a burst of deletions of variable length. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 757–762, 2020. doi:10.1109/ISIT44484.2020.9174288.
- 13 Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk*, 163(4):845–848, 1965.

- 14 Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. Random access in large-scale DNA data storage. *Nature biotechnology*, 36(3):242, 2018. doi:10.1038/nbt.4079.
- 15 Clayton Schoeny, Antonia Wachter-Zeh, Ryan Gabrys, and Eitan Yaakobi. Codes correcting a burst of deletions or insertions. *IEEE Transactions on Information Theory*, 63(4):1971–1985, 2017. doi:10.1109/TIT.2017.2661747.
- 16 Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, 1999. doi:10.1109/18.796406.
- 17 Jin Sima and Jehoshua Bruck. On optimal  $k$ -deletion correcting codes. *IEEE Transactions on Information Theory*, 67(6):3360–3375, 2021. doi:10.1109/TIT.2020.3028702.
- 18 Neil J. A. Sloane. On single-deletion-correcting codes. *arXiv*, 2002. doi:10.48550/arXiv.math/0207197.
- 19 Ilia Smagloy, Lorenz Welter, Antonia Wachter-Zeh, and Eitan Yaakobi. Single-deletion single-substitution correcting codes. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 775–780, 2020. doi:10.1109/ISIT44484.2020.9174213.
- 20 Wentu Song, Kui Cai, and Tuan Thanh Nguyen. List-decodable codes for single-deletion single-substitution with list-size two, January 2022. doi:10.48550/arXiv.2201.02013.
- 21 Wentu Song, Nikita Polyanskii, Kui Cai, and Xuan He. On multiple-deletion multiple-substitution correcting codes. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2655–2660, 2021. doi:10.1109/ISIT45174.2021.9517878.
- 22 Daniel Tan. Implementation of single-edit correcting code, 2020. URL: <https://github.com/dtch1997/single-edit-correcting-code>.
- 23 Yuanyuan Tang and Farzad Farnoud. Error-correcting codes for short tandem duplication and edit errors. *IEEE Transactions on Information Theory*, 68(2):871–880, 2022. doi:10.1109/TIT.2021.3125724.
- 24 Rom R. Varshamov and Grigory M. Tenengolts. Codes which correct single asymmetric errors. *Autom. Remote Control*, 26(2):286–290, 1965.
- 25 Antonia Wachter-Zeh. List decoding of insertions and deletions. *IEEE Transactions on Information Theory*, 64(9):6297–6304, 2018. doi:10.1109/TIT.2017.2777471.
- 26 Shuche Wang, Jin Sima, and Farzad Farnoud. Non-binary codes for correcting a burst of at most 2 deletions. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2804–2809, 2021. doi:10.1109/ISIT45174.2021.9517917.
- 27 S. M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and error-free DNA-based data storage. *Scientific reports*, 7(1):5011, 2017. doi:10.1038/s41598-017-05188-1.





# Affine Extractors and AC0-Parity

Xuangui Huang ✉

Northeastern University, Boston, MA, USA

Peter Ivanov ✉

Northeastern University, Boston, MA, USA

Emanuele Viola ✉

Northeastern University, Boston, MA, USA

---

## Abstract

We study a simple and general template for constructing affine extractors by composing a linear transformation with resilient functions. Using this we show that good affine extractors can be computed by non-explicit circuits of various types, including AC0-Xor circuits: AC0 circuits with a layer of parity gates at the input. We also show that one-sided extractors can be computed by small DNF-Xor circuits, and separate these circuits from other well-studied classes. As a further motivation for studying DNF-Xor circuits we show that if they can approximate inner product then small AC0-Xor circuits can compute it exactly – a long-standing open problem.

**2012 ACM Subject Classification** Theory of computation → Circuit complexity

**Keywords and phrases** affine extractor, resilient function, constant-depth circuit, parity gate, inner product

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.9

**Category** RANDOM

**Funding** Supported by NSF grants CCF-1813930 and CCF-2114116.

**Acknowledgements** We are grateful to the anonymous reviewers for helpful feedback.

AC0 with parity gates is a frontier class in circuit complexity, essentially the strongest class for which we can prove strong lower bounds for explicit functions. These lower bounds however have been stuck since the classic results from the 80's [32, 36]. In particular, unlike the case of AC0, we do not have (1) strong average-case lower bounds, (2) pseudorandom generators, or (3) hierarchy results for this class.

Remarkably, (1), (2), and (3) are not known even for the subclass AC0-Xor of AC0 circuits with a layer of parity gates (or their negations) next to the input level. (On the other hand, (1) and (2) are known for Xor-AC0 [39].) In fact, (1) and (2) are not known even for Or-And-Xor circuits, a.k.a. DNF-Xor circuits. Hence these classes (AC0-Xor and DNF-Xor) have gained importance as prominent special cases of AC0 with parity gates which require new proof techniques.

A natural candidate for providing (1) is the inner product function, and the following question has been highlighted and studied in several works, including [35, 14, 5, 13, 18].

► **Problem 1.** Is the Inner Product function  $\text{IP}(x, y) := \sum_i x_i y_i \bmod 2$  computable by polynomial-size AC0-Xor circuits?

A number of works have solved special cases of the problem, proving lower bounds for computing IP when the circuit class is further restricted: [27, 14] proved exponential lower bounds for Or-And-Xor. [5] proved a lower bound for small AC0-Xor circuits when the parity layer is “typical.” [13] (cf. [28]) proved an  $n^{2-o(1)}$  lower bound for And-Or-And-Xor circuits. For depth- $d$  AC0-Xor circuits they proved an  $n^{1+\Omega(1/4^d)}$  lower bound. The latter result was improved on in [10] which obtained an  $\Omega(n^{1+1/2^d})$  lower bound that holds even if the circuit computes IP on a  $1/2 + n^{-\log n}$  fraction of inputs.



© Xuangui Huang, Peter Ivanov, and Emanuele Viola;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 9; pp. 9:1–9:14



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To summarize, to compute IP there are quadratic lower bounds for And-Or-And-Xor. These lower bounds hold in the worst case while average-case lower bounds are not known. Average-case lower bounds are not even known for polynomial-size DNF-Xor. For higher depth we have lower bounds for size which approaches linear exponentially fast with the depth, and these lower bounds hold even in the average case.

**Extractors.** An extractor for a class of distributions (a.k.a. source) is a function that is nearly unbiased when the input is chosen according to any distribution in the class. For various classes of distributions, extractors have been studied with remarkable intensity in the theoretical computer science literature for decades. A class of distributions which is important in many works including the present one is that of distributions which are uniform over *linear* or *affine vector subspaces* of  $\{0, 1\}^n$ , which we simply call *affine*.

► **Definition 2** (Affine extractors). *A function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is an affine extractor for dimension (a.k.a. entropy)  $k$  with error (a.k.a. bias)  $\epsilon$  if for every  $k$ -dimensional affine space  $A \subseteq \{0, 1\}^n$  and for  $U_A$  the uniform distribution over  $A$  we have  $|\mathbb{P}[f(U_A) = 1] - \mathbb{P}[f(U_A) = 0]| \leq \epsilon$ .*

*We say that the extractor is one-sided if the conclusion is relaxed to  $\mathbb{P}[f(U_A) = 1] \geq 1/2 - \epsilon$ , and  $f$  is nearly balanced:  $|\mathbb{P}[f(U) = 1] - 1/2| \leq \epsilon$ , where  $U := U_{\{0, 1\}^n}$ .*

Many papers have been devoted to constructing affine extractors. The latest [11] works for nearly logarithmic dimension.

One motivation for studying affine extractors is that they arise naturally in the study of *circuit lower bounds*. For example, the method of *restrictions* partitions the input in affine spaces, and so any function that becomes constant via a suitable restriction cannot be a good affine extractor. In particular, switching lemmas [20, 1, 23, 26, 24, 25] imply that small AC0 circuits cannot compute affine extractors. The same holds for models which shrink under restrictions, such as De Morgan formulas, see [37] for the latest shrinkage bound and history. And the first numerical progress in more than 30 years on lower bounds for general circuits – [19] – holds for computing affine extractors. Finally, affine extractors also give *sampling lower bounds* [41, 42, 44].

This also means that showing that a circuit class can compute good affine extractors indicates some of the difficulties that may arise when trying to prove lower bounds against that class. This direction has been pursued in a number of works, in fact going back to [31] (cf. [34]). More recently, the paper [15] (Theorem A.6) shows that affine extractors for dimension  $k = O(\log n)$  can be computed by

1. polynomials mod 2 of degree  $O(\log n)$ ,
2. Xor-And-Xor circuits of size  $n^{2+o(1)}$ ,
3. De Morgan's formulas of size  $n^{5+o(1)}$ .

Their results also give good dependence on  $\epsilon$ , which we omit for simplicity.

It is a folklore result that IP is an affine extractor for dimension larger than  $n/2$  (a proof can be found in [42]). Moreover, some of the previous lower bounds hold for computing affine extractors. The worst-case  $n^{1+c^{-d}}$  lower bound for depth  $d$  in [5] holds for computing affine extractors, even with very weak parameters. The quadratic lower bound for And-Or-And-Xor [13] and the average-case lower bound for depth  $d$  [10] hold for computing extractors if the error is exponentially small. We do not know if they can be generalized to affine extractors with constant error, but jumping ahead we give a simple proof of an  $n^{1.5-o(1)}$  lower bound for computing constant-error extractors by And-Or-And-Xor circuits (Section 4).

Our first main result is that small (non-explicit) AC0-Xor circuits can compute very good affine extractors. In fact, And-Or-And-Xor circuits of size  $n^2 \log^{O(1)} n$  suffice, matching the depth and – up to logarithmic factors – the size lower bounds in [13].

► **Theorem 3.** *There exists an And-Or-And-Xor circuit  $C$  of size  $n^2 \log^{O(1)} n$  that computes an affine extractor for dimension  $k \geq \log^c n$  with error  $1/\Omega(\log n)$ , where  $c > 0$  is an absolute constant.*

The proof is in Section 1. We actually give a general template for constructing affine extractors, and obtain constructions in other models as well. In particular, we show that De Morgan formulas of size  $n^{4+o(1)}$  can compute affine extractors (see Theorem 15), improving the  $n^{5+o(1)}$  bound from [15] (Item 3 above).

It is natural to ask if the depth of the circuit in Theorem 3 is tight, that is if Or-And-Xor (a.k.a. DNF-Xor) circuits can compute good affine extractors. We note that an And-Xor circuit computes (the characteristic function of) an affine space, and so a DNF-Xor circuit of size  $s$  computes an union of  $s$  affine spaces. Understanding the power of unions of affine spaces seems interesting from a mathematical perspective as well, and it is a natural next step towards more general models after affine spaces.

It is easy to show that DNF-Xor circuits require exponential size to compute good affine extractors, and a proof can be found in [14]. However, we show next that they can compute *one-sided* extractors. (Note that the DNF-Xor sub-circuits in the construction in Theorem 3 are not balanced and so do not compute one-sided affine extractors.)

► **Theorem 4.** *There exists an  $O(n \log^2 n)$  size DNF-Xor circuit that computes a one-sided affine extractor for dimension  $k \geq c \log^3 n$  with error  $1/\log^{1.9} n$ , where  $c > 0$  is an absolute constant.*

We apply this theorem to separate DNF-Xor circuits from other classes such as parity decision trees (PDTs) and AC0-Xor circuits with  $n$  parity gates. The separation from PDTs is obtained by showing the more general separation from *disjoint* unions of affine subspaces. These separations hold in the average case too, and we show tightness with respect to several parameters. These results point to the strength of the model and to the techniques we can (not) use for lower bounds.

Let us elaborate on the separation from PDTs. For comparison, recall that any polynomial-size DNF on  $n$  bits can be approximated by a decision tree (DT) of depth  $n - \Omega(n/\log n)$ . (Proof sketch: We can ignore terms of size  $\omega(\log n)$ . Then a switching lemma shows that we can fix all but  $\Omega(n/\log n)$  variables and the DNF collapses to a decision tree of depth  $O(\log n)$ .) It is natural to ask if a corresponding switching lemma or simulation exists for DNF-Xor in terms of PDT. We show that the answer is negative:

► **Corollary 5.** *There exists a DNF-Xor circuit  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  of size  $n \cdot \text{poly} \log n$  such that for any depth  $n - \log^{2+o(1)} n$  PDT  $T : \{0, 1\}^n \rightarrow \{0, 1\}$  we have  $\mathbb{P}[f(U) = T(U)] \leq 1/2 + 1/\Omega(\log n)$ .*

Note that the “depth deficiency” (i.e.,  $n$  minus the depth of the tree) decreases exponentially from the  $\Omega(n/\log n)$  in the simulation of DNFs by DTs to  $\log^{2+o(1)} n$  in the simulation of DNF-Xors by PDTs. We summarize this finding informally as follows:

– PDTs are *not* to DNF-Xor what DTs are to DNFs –

This finding stands in contrast with our extensions of other simulations of AC0 circuits by DTs to the setting of AC0-Xor circuits and PDTs. This includes simulations given by the switching lemma, and simulations that exploit various restrictions on fan-in, see Section 2.

The study of DNF-Xor circuits is also motivated by our next result, which shows that if IP can be approximated by small such circuits, then in fact IP can be computed (exactly) by small AC0-Xor circuits.

► **Theorem 6.** *Suppose there is  $c > 0$  and an AC0-Xor circuit of size  $n^c$  that computes IP correctly on a  $1/2 + 1/\log^c n$  fraction of the inputs. Then there are polynomial-size, constant-depth AC0-Xor circuits that compute IP.*

The proof is in Section 3.

A concurrent work [18] shows that if small DNF-Xor circuits compute IP on a  $5/6 + \epsilon$  fraction of the inputs, then there are efficient data-streaming and communication protocols for low-degree polynomials. The conclusions in [18] and the present work thus concern different models. The hypotheses are also different. Whereas [18] requires a DNF-Xor circuit to compute IP on a  $5/6 + \epsilon$  fraction of the inputs, in our application an AC0-Xor circuit computing it on a  $1/2 + 1/\text{poly log}$  fraction suffices. Also, a partial converse to Theorem 6 is given by the so-called discriminator lemma [22]: if a size- $s$  And-Or-And-Xor circuit computes IP, then a size- $s$  DNF-Xor circuit computes IP on  $1/2 + 1/s$  fraction of the inputs.

We note that the hypothesis in Theorem 6 is related to extractors. Indeed, let  $C$  be a DNF-Xor circuit of size  $s$ . Let  $S := \{x : C(x) = 1\}$  and let  $|S|/2^n =: p$ . Suppose that IP is biased on  $S$ , that is  $|\mathbb{P}[IP(U_S) = 1] - \mathbb{P}[IP(U_S) = 0]| \geq \epsilon$ . Then either  $C$  or the negation of  $C$  computes IP correctly on a  $1/2 + p\epsilon$  fraction of inputs. In other words, if IP is not an extractor for  $U_S$ , then we can approximate IP, and by Theorem 6 we can compute it with small AC0-Xor circuits. To avoid the latter, IP should have bias  $\leq 1/\log^c n$  on any set  $S$  as above of size  $\geq 1/\log^c n$ , for any  $c$ .

► **Problem 7.** Does IP extract randomness from unions of polynomially many affine spaces?

This work raises several other questions. Besides the question of explicitness, an obvious question is matching lower bounds and affine-extractor constructions. In particular, it would be interesting to know if one can compute affine extractors by depth- $d$  AC0-Xor circuits of size  $n^{1+c^{-d}}$ . This would follow if one can show a *size-depth tradeoff* for  $r$ -wise resilient functions (defined later), which we also raise as a question. In general, we raise the question of understanding the complexity of computing  $r$ -wise resilient functions in various models of computation. For example, can they be computed by linear-size circuits? From the side of lower bounds, it would be interesting to strengthen our  $n^{1.5-o(1)}$  lower bound for computing affine extractors (in Section 4) to quadratic.

## 1 Constructing affine extractors

The proof of Theorem 3 builds on ideas developed in the literature on extractors. At the high level, we use an approach from [21], Section 5.3, of combining a suitable linear transformation with a *resilient function* (defined below). [21] aims to construct extractors for *bit-fixing* sources (a special case of affine sources) of *large* entropy ( $n/\text{poly log}$ ) and computable in AC0. They pick a sparse linear transformation, which guarantees that the extractor is computable in AC0, and which is sufficient because the entropy is close to  $n$ . By contrast, we aim to extract from the more general affine sources, and even with polylogarithmic entropy. On the other hand, we can pick a non-sparse linear transformation thanks to the layer of parity gates. [21] shows that the output of the linear transformation is uniform except for few bits; instead we can only guarantee that it is  $r$ -wise independent except for few bits.

► **Definition 8** ([41]). A distribution  $D$  over  $\{0, 1\}^m$  is  $r$ -wise uniform but for  $b$  bits if there is a set  $S \subseteq [m]$  of size  $m - b$  such that for any  $r$  elements in  $S$  the projection of  $D$  onto the corresponding bits is uniform over  $\{0, 1\}^r$ . If  $b = 0$  we simply say  $r$ -wise uniform.

A main and simple result in this paper is that applying a suitable linear transformation one can turn an affine source into a distribution of the type above. The corresponding linear transformations seem interesting to study, so we give a definition.

► **Definition 9.** An  $m \times n$  matrix  $T$  is  $k$ -affine to  $r$ -wise uniform but for  $b$ -bits if for any distribution  $U_A$  uniform over an affine space  $A \subseteq \{0, 1\}^n$  of dimension  $\geq k$  the distribution  $TU_A$  is  $r$ -wise uniform but for  $b$  bits.

We raise the question of understanding the complexity of computing such matrices efficiently. For example, in particular we ask if these transformations (with good parameters as below) can be computed by linear-size circuits, local maps, etc. For starters, we prove that such matrices exist via the probabilistic method.

► **Lemma 10.** A matrix  $T$  as in Definition 9 exists for any  $b > 3n$  and  $k \geq 2r \log m$ .

**Proof.** It suffices to prove the lemma for any linear space (rather than affine). To verify this, write the uniform distribution over an affine space  $A$  as  $SX + s$  where  $S$  is a full-rank  $n \times k$  matrix,  $X \in \{0, 1\}^k$  is uniform, and  $s \in \{0, 1\}^n$  is a fixed shift. Consider  $T(SX + s) = TSX + Ts$ . Since  $SX$  is a linear space,  $TSX$  is  $r$ -wise uniform but for  $b$  bits. This property is unaffected by adding the fixed shift  $Ts$ .

Let  $U$  be uniform over  $\{0, 1\}^k$ . Recall that for an  $r \times k$  matrix  $M$  the distribution  $MU$  is uniform if and only if the rows of  $M$  are linearly independent. Hence, for our goal it suffices to construct matrices such that if we exclude  $b$  rows, any  $r$  of the remaining rows are linearly independent. Pick  $T$  uniformly at random. Fix a full-rank  $n \times k$  matrix  $S$  and note that  $TS$  is a uniform  $m \times k$  matrix  $M$ . We say a set  $B \subseteq [m]$  of size  $b$  is *bad* if each row (with index) in  $B$  is a linear combination of  $\leq r$  rows not in  $B$ . If such bad sets of size  $b$  do not exist then the proof is completed as follows. Greedily pick rows of  $TS$  that are not linear combinations of  $\leq r$  rows already picked. One can pick  $\geq m - b$  rows, otherwise a bad set of size  $b$  exists.

Now we want to bound the probability that there exists a bad set  $B$  of size  $b$ . Fix  $B$ , and fix arbitrarily the rows of  $M$  not in  $B$ . Let  $H$  be the set of vectors that can be obtained as a linear combination of  $\leq r$  rows not in  $B$ . We have

$$|H| \leq \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r} \leq 2^{r \log m}.$$

The probability that each row in  $B$  falls in  $H$  is then

$$\left(\frac{|H|}{2^k}\right)^b = 2^{b(r \log m - k)}.$$

When  $k \geq 2r \log m$  this probability is

$$\leq 2^{-kb/2}.$$

Hence the probability that there exists a bad set of size  $b$  is

$$\leq \binom{m}{b} 2^{-kb/2} \leq 2^{b(\log m - k/2)} \leq 2^{-bk/3}.$$

Finally, there are at most  $2^{kn}$  linear spaces of dimension  $k$ . Therefore the probability that there exists such a space with a bad set as above is  $\leq 2^{k(n-b/3)}$ . Setting  $b > 3n$  this probability is less than one and the desired matrix  $T$  exists. ◀

Given this lemma, it remains to extract from distributions over  $\{0,1\}^m$  which are  $r$ -wise uniform but for  $b$  bits.

► **Definition 11.** *A function  $f : \{0,1\}^m \rightarrow \{0,1\}$  is  $r$ -wise  $(b, \epsilon)$ -resilient if for any  $r$ -wise uniform distribution  $X$  we have  $|\mathbb{P}[f(X) = 1] - 1/2| \leq \epsilon$ , and for any set  $B$  of size  $\leq b$  the probability over  $X$  that changing the bits in  $B$  changes the value of  $f$  is  $\leq \epsilon$  (and in particular the bias that one can obtain changing those bits is  $\leq 1/2 + 2\epsilon$ ). Note that this is equivalent to saying that  $f$  is an extractor with error  $2\epsilon$  for distributions which are  $r$ -wise uniform but for  $b$  bits.*

The paper [41] showed that the majority function is resilient over  $r$ -wise uniform distributions, relying on the Central Limit Theorem for  $r$ -wise uniform distributions from [16].

► **Lemma 12** ([41]). *The Majority function is  $r$ -wise  $(m^{0.499}, 1/100)$ -resilient for all sufficiently large  $r$ .*

Using this, we can show that Maj-Xor circuits can compute affine extractors with optimal dependence on dimension, up to constant factors.

► **Theorem 13.** *There is a non-explicit Maj-Xor circuit of polynomial size that computes an affine extractor for dimension  $O(\log n)$  with error  $1/100$ .*

**Proof.** Apply Lemma 10 with  $m = n^{2.1}$  and  $b = 4n$ . Let  $V$  be an affine space of dimension  $k \geq 2r \log m = O(\log n)$ . Let  $U_V$  be the uniform distribution over  $V$ . By Lemma 10,  $TU_V$  is  $r$ -wise independent except for  $b$  bits. We conclude by Lemma 12. ◀

For Theorem 3 we need extractors computable in AC0 however. The seminal work [3] (cf. [46] for a streamlined exposition of a slightly weaker result) showed through the probabilistic method the existence of functions on  $m$  bits that are  $m$ -wise  $(\Omega(\alpha m / \log^2 m), O(\alpha))$ -resilient for any  $\alpha$ . Moreover, their functions are computable by polynomial-size AC0 circuits. We observe that their construction is also resilient over poly log  $m$ -wise distributions. This can be shown using the fact that polylog-wise uniformity fools AC0 circuits [6, 33, 9], and for completeness we include a proof in Section A.

► **Lemma 14.** *There exist  $c > 0$  and a function  $f : \{0,1\}^m \rightarrow \{0,1\}$  that is  $\log^c m$ -wise  $(\Omega(\alpha m / \log^2 m), O(\alpha))$ -resilient, for any  $\alpha \geq 1/m$ . Moreover,  $f$  is computable by depth-3 circuits of size  $O(m^2 / \log m)$ .*

We note that explicit constructions of poly log-wise resilient functions appear in [12] and [29]. One can use either [12] or [29] to obtain affine extractors with our approach. However, some of the parameters would be a little worse than what we claimed. For example, the circuit size would be  $n^c$  for  $c > 2$ . Using these constructions we see that the only bottleneck to an explicit construction is the layer of parity gates. Should an explicit construction for that be found, the affine extractor would be simpler than previous explicit constructions for comparable entropy (see [11] and references therein).

**Proof of Theorem 3.** The parity gates compute the linear transformation  $T$  in Lemma 10 with the parameters  $m = O(n \log^3 n)$  and  $b = m / \log^3 m > 3n$ . By the assumption on  $k$  we have  $r = \log^{c'} m$  for a constant  $c'$  as large as desired. The distribution  $TU$  is  $r$ -wise uniform but for  $b$  bits. We feed its output to the function  $f$  in Lemma 14 for  $\alpha = \Theta(1/\log m)$ . Then  $f$  is  $(m/\log^3 m, O(1/\log m))$ -resilient, and the result follows. ◀



Finally, we obtain a construction for De Morgan formulas.

► **Theorem 15.** *The affine extractor in Theorem 3 can be computed by De Morgan formulas of size  $n^4 \log^{O(1)} n$ , or by formulas over the full binary base  $B_2$  of size  $n^3 \log^{O(1)} n$ .*

**Proof.** From the proof of Theorem 3 we know that the fan-ins of the And-Or-And-Xor circuit, starting from the output And, are  $n \log^{O(1)} n, n \log^{O(1)} n, \log^{O(1)} n, n$ . Note that an And or Or on  $t$  bits can be computed by De Morgan formulas of size  $O(t)$ , while Parity on  $t$  bits can be computed by such formulas of size  $O(t^2)$  and  $B_2$  formulas of size  $O(t)$ . The result follows. ◀

## 2 DNF-Xor

Our construction proving Theorem 4 is similar to our affine-extractor construction. We show that the so-called Tribes function is “one-sided resilient,” so composing it with the layer of Xor gates from Lemma 10 yields a one-sided extractor.

► **Definition 16** ([7]; cf. [30], Proposition 4.12).  *$\text{Tribes}_w : \{0, 1\}^m \rightarrow \{0, 1\}$  is the read-once DNF where every term has size  $w$  and  $|\mathbb{P}[\text{Tribes}(U) = 1] - 1/2| = O(\log m)/m$ . This makes  $w = \log m - \log \log m + O(1)$ .*

We need the following lemma.

► **Lemma 17.** *Let  $D$  be a  $w \log(1/\epsilon)$ -wise uniform distribution on  $n$  bits. Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a read-once DNF with terms of size  $\leq w$ . Let  $U$  be the uniform distribution over  $\{0, 1\}^n$ . Then  $|\mathbb{P}[f(D) = 1] - \mathbb{P}[f(U) = 1]| \leq \epsilon$ .*

This claim follows by noting that the input distribution to the Or in the DNF is  $\log(1/\epsilon)$ -wise *independent* (and not necessarily uniform). We can then apply the corresponding fundamental result in pseudorandomness [17]; see [43], Lecture 1, for an exposition. The latter result states that the Or of  $\log(1/\epsilon)$ -wise independent random variables has the same probability of being one as the Or of independent random variables with the same marginals.

**Proof of Theorem 4.** We need a slight extension of Lemma 10. We claim that the matrix  $T$  constructed there has the additional property  $(\star)$  that any linear combination of  $\leq r$  rows of  $T$  is nonzero. This can be established with essentially the same proof, because the probability that a uniform  $m \times k$  matrix does not satisfy this is  $\leq 2^{O(r \log m) - k}$  which is less than  $1/2$  by our choice of parameter (and the proof of the lemma shows that a uniformly selected  $T$  satisfies the lemma with probability  $> 1/2$ ).

Hence, consider the matrix  $T$  from Lemma 10 with  $m = O(n \log^2 n)$  and  $b = 4n$ , and further take it to satisfy  $(\star)$ . Feed this distribution into the Tribes function on  $m$  bits. First, note that by  $(\star)$  we have that  $TU$  is  $r$ -wise uniform where  $r = k/2 \log m$ . Hence, the output distribution of the And gates is  $r/\log m$ -wise independent. By our assumption that  $k \geq c \log^3 n$ , it will be  $c \log m$ -wise independent for a  $c$  large enough so that by Lemma 17 the probability that Tribes outputs 1 on  $TU$  is within  $1/m$  of the probability it outputs 1 over the uniform distribution, and so still within  $O(\log m)/m$  of  $1/2$ .

This proves that our function is indeed nearly balanced. There remains to prove that it is 1 with high probability over any large affine space  $S$ . We have that  $TSU$  is  $k/4 \log m$ -wise uniform but for  $b = 4n$  bits. Now we basically show that the good bits suffice to make the function 1 with probability about  $1/2$ . The bad bits touch  $\leq b$  terms. Hence there are  $m/w - b$  good terms, defined as those terms that do not take any bad bit as input. By Lemma 17 as above, the probability that the Or of the good terms is 0 over  $TSU$  is within  $1/m$  of the probability that it is 0 over  $U$ . The latter probability is

$$(1 - 2^{-w})^{m/w-b} \leq (1/2 + O(\log m)/m)(1 - 2^{-w})^{-b},$$

where the first factor in the right-hand side is from the definition of Tribes. For the second term note that

$$(1 - 2^{-w})^{-b} \leq e^{b/2^w}.$$

We have  $2^w = \Theta(m/\log m) = \Theta(n \log^2 n / \log \log n)$  and so  $b/2^w \leq 1/\log^{1.9} n$  and  $e^{b/2^w} \leq 1 + O(1)/\log^{1.9} n$ , and the result follows.  $\blacktriangleleft$

We now use the above result to give separations.

**► Definition 18.** We say  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  is a  $k$ -affine-partition if  $g$  can be expressed as  $g(x) = \sum_{i=1}^t \alpha_i \mathbf{1}_{V_i}$  where  $V_1, V_2, \dots, V_t$  are disjoint affine subspaces of dimension  $\geq k$  that form a partition of  $\mathbb{F}_2^n$  and for each  $i$ ,  $\alpha_i \in \{0, 1\}$ .

We next show one-sided extractors for dimension  $k$  cannot even be approximated by  $k$ -affine partitions.

**▷ Claim 19.** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a one-sided extractor for dimension  $k$  with error  $\epsilon$ , and let  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  be a  $k$ -affine partition. Then

$$\mathbb{P}[f(U) = g(U)] \leq \frac{1}{2} + 3\epsilon.$$

*Proof.* Let  $G_0 := \{x : g(x) = 0\}$ ,  $p := |G_0|/2^n$  and  $G_1 := \{x : g(x) = 1\}$ . Let  $\alpha := \mathbb{P}_{x \in G_0}[f(x) = 0]$  and  $\beta := \mathbb{P}_{x \in G_1}[f(x) = 0]$ . We have

$$\mathbb{P}[f(U) \neq g(U)] = p(1 - \alpha) + (1 - p)\beta.$$

Because  $f$  is nearly balanced, we have  $p\alpha + (1 - p)\beta \geq 1/2 - \epsilon$ , and so  $(1 - p)\beta \geq 1/2 - \epsilon - p\alpha$ . Plugging this above we get

$$\mathbb{P}[f(U) \neq g(U)] \geq 1/2 - \epsilon - p\alpha + p(1 - \alpha) = 1/2 - \epsilon + p(1 - 2\alpha).$$

Also by the extractor property we have  $\alpha \leq 1/2 + \epsilon$ . (Since  $G_0$  is the disjoint union of spaces on which  $f$  outputs 0 on at most a  $1/2 + \epsilon$  fraction of the elements.) Hence  $1 - 2\alpha \geq -2\epsilon$ . Combining with above yields

$$\mathbb{P}[f(U) \neq g(U)] \geq 1/2 - \epsilon - 2p\epsilon \geq 1/2 - 3\epsilon. \quad \blacktriangleleft$$

We showed in Theorem 4 that small DNF-Xor circuits can compute such extractors. In fact, the circuits are of the type  $\text{Or}_{n \log^{O(1)} n} \text{-And}_{O(\log n)} \text{-Xor}$ ; subscripts indicate fan-ins. Again, this is equivalent to a nearly-linear collection of spaces of very large dimension ( $n - O(\log n)$ ) that cannot be approximated by *disjoint* spaces, even if the dimensions of the latter spaces are as small as polylogarithmic. Note that a parity decision tree (PDT) on  $n$  bits with depth  $n - k$  gives a  $k$ -affine partition, and this proves Corollary 5.

It is natural to ask if this separation (between DNF-Xor and PDT) is tight. We show that indeed it is, in three different settings.



## 2.1 Setting 1: The number of parity gates

We consider AC0-Xor circuits where the Xor gates correspond to a basis.

► **Definition 20.** *AC0-Xor-B circuits on  $n$  bits are AC0-Xor circuits where the number of Xor gates is  $n$  and the corresponding vectors form a basis.*

We show that small AC0-Xor-B circuits *can* be approximated by moderate-depth PDTs, showing that in Corollary 5 it is essential that the number of Xor gates is larger than  $n$ , even if we allow general AC0 post-process (instead of DNF).

The proof amounts to observing that switching lemmas for AC0 apply as stated to AC0-Xor-B, except that they yield PDTs rather than DTs. Specifically, it follows for example from the switching lemma in [24] (see Corollary 11 in [45] for an explicit statement about AC0) that for  $h := 2^{o(n/2^d \log^{d-1} n)}$  an AC0 circuit of size  $\leq h$  and depth  $d$  can be approximated by a DT of depth  $n - \Omega(n/\log^{d-1} n)$  except with error  $1/h$ . The corresponding statement applies to AC0-Xor-B.

▷ **Claim 21.** For  $h := 2^{o(n/2^d \log^{d-1} n)}$ , an AC0-Xor-B circuit of size  $\leq h$  and depth  $d$  can be approximated by a PDT of depth  $n - \Omega(n/\log^{d-1} n)$  except with error  $1/h$ .

To prove this, apply the result for AC0 mentioned above to the AC0 part of the circuit. Querying one input bit to the AC0 part can be simulated by querying a parity of the input bits to the AC0-Xor-B circuit, resulting into a PDT. A straightforward combination of the above results also yields a separation between small DNF-Xor and AC0-Xor-B circuits.

## 2.2 Setting 2: The fan-in of the And gates

Next we show that the fan-in of the And gates in the separation (between DNF-Xor and PDT) is tight up to an  $O(\log \log n)$  factor: We show that any Or-And-Xor circuit where the And fan-in is at most  $\log n - 2 \log \log n$  can be approximated by a depth  $O(n/\log n)$  PDT with at most constant error. This follows from the following lemma, which is a “PDT version” of the corresponding result for DNF and DT, see [4, 38]. We follow the exposition in [45].

► **Lemma 22.** *For every Or-And<sub>w</sub>-Xor circuit  $C : \{0, 1\}^n \rightarrow \{0, 1\}$ , there exists a PDT  $T$  of depth  $\leq 2w2^w \log(1/\epsilon)$  with range  $\{0, 1, ?\}$  such that:*

1.  $\Pr_{x \in \{0, 1\}^n} [T(x) = ?] \leq \epsilon$ .
2. For all  $x \in \{0, 1\}^n$ ,  $T(x) \neq ? \Rightarrow T(x) = C(x)$ .

**Proof.** We are going to define  $T : \{0, 1\}^n \rightarrow \{0, 1, ?\}$  recursively. If  $C$  is a constant then  $T$  is a constant. Otherwise, let  $C = \bigvee_{i=1}^m C_i$  where each subcircuit  $C_i$  is an And of a set of at most  $w$  parities, denoted by  $P_i$ . We can assume w.l.o.g. that for each  $i$ ,  $P_i$  is linearly independent. We greedily construct an index set  $I \subseteq [m]$  as follows: we look at each  $P_i$  one-by-one, and add  $i$  into  $I$  if  $P_i \cup \bigcup_{j \in I} P_j$  is linearly independent. There are two cases:

1. If  $|I| \geq 2^w \log(1/\epsilon)$ , we let  $T$  query all the parities in  $P_i$  for the first  $2^w \log(1/\epsilon)$  indices in  $I$ , which decide the values for the corresponding subcircuits  $C_i$ . If any of the subcircuits is True, then  $T$  outputs 1, otherwise it outputs ?.
2. Otherwise  $|I| < 2^w \log(1/\epsilon)$ , then the size of  $\bigcup_{i \in I} P_i$  is at most  $w2^w \log(1/\epsilon)$ . Moreover, for any  $P_j$  with  $j \notin I$ , there must exist a parity  $p_j \in P_j$  such that  $p_j \in \text{span}(\bigcup_{i \in I} P_i \cup (P_j \setminus \{p_j\}))$ . The tree  $T$  first queries every parity in  $\bigcup_{i \in I} P_i$ . After that, we know that for each  $j \notin I$ ,  $p_j \in \text{span}(P_j \setminus \{p_j\})$ . As the subcircuit  $C_j$  is an And of the parities in  $P_j$ , if setting all the parities in  $P_j \setminus \{p_j\}$  to be 1 forces  $p_j$  to be 0, we can just ignore this subcircuit. If it forces  $p_j$  to be 1, we can safely remove  $p_j$  to get an And of  $\leq w - 1$

parities. Therefore what we get is an Or-And-Xor circuit  $C'$  where the fan-in of each And is  $\leq w - 1$  (which might depend on the results of the queries), and we recurse on  $C'$  to get a parity decision tree  $T'$ .

The depth of  $t_C$  is  $\leq w2^w \log(1/\epsilon) + (w - 1)2^{w-1} \log(1/\epsilon) + \dots \leq 2w2^w \log(1/\epsilon)$ . Item 2 is evident by definition. For Item 1, note that  $T$  outputs ? only if none of the first  $2^w \log(1/\epsilon)$  subcircuits in  $I$  is True. Each  $P_i$  is linearly independent, and by construction of  $I$  the outputs of these subcircuits are independent, so the probability can be bounded by  $(1 - 2^{-w})^{2^w \log(1/\epsilon)} \leq \epsilon$ . ◀

### 2.3 Setting 3: The fan-in of the Or gate

We show that  $\text{Or}_{o(n/\log n)}$ -And-Xor circuits can be approximated by moderate-depth PDTs.

▷ **Claim 23.** Let  $C : \{0, 1\}^n \rightarrow \{0, 1\}$  be an  $\text{Or}_{o(n/\log n)}$ -And-Xor circuit. There exists a PDT  $T$  of depth  $(1 - \Omega(1))n$  such that  $\mathbb{P}[C(U) = T(U)] \geq 1/2 + \Omega(1)$ .

*Proof.* Let  $C'$  denote the circuit obtained from  $C$  by deleting all the And gates with fan-in greater than  $\log n - \log \log n - O(1)$ . By Lemma 22 a PDT with depth  $(1 - \Omega(1))n$  can approximate  $C'$  with constant error. Now we argue that the removal of And gates does not introduce too much error. Any removed And gate evaluates to True under uniform inputs with probability  $\leq 2^{-(\log n - \log \log n - O(1))} = O(\log n/n)$ . Since the number of removed And gates is  $o(n/\log n)$ , by a union bound the total error introduced by the removal is  $o(1)$ . ◀

## 3 Proof of Theorem 6

We begin by observing that IP can be “randomly self-reduced” very efficiently: the overhead is just computing a parity. More formally:

$$IP(x + a) = IP(x) + L_a(x)$$

for any  $x, a \in \{0, 1\}^n$ , and where  $L_a : \{0, 1\}^n \rightarrow \{0, 1\}$  is an affine transformation that depends on  $a$  only. To verify this just consider a monomial and note that  $(x_1 + a_1)(x_2 + a_2) = x_1x_2 + a_1x_2 + a_2x_1 + a_1a_2 = x_1x_2 + L_a(x)$ . The same fact is used for example in pseudorandom generators for low-degree polynomials [8], and in [18]. In general the proof is also similar to the simplified average-case lower bounds for parity [40].

Let  $C : \{0, 1\}^n \rightarrow \{0, 1\}$  be a circuit that computes IP on  $1/2 + \epsilon$  fraction of the inputs. Consider the random circuit  $C_A$  for uniform  $A$  which on input  $x$  outputs

$$C_A(x) := C(x + A) + L_A(x).$$

Note that for every  $x$ ,  $\mathbb{P}_A[C_A(x) = IP(x)] \geq \mathbb{P}_A[C(x + A) = IP(x + A)] \geq 1/2 + \epsilon$ . Moreover, for any fixed  $A$  the circuit  $C_A$  is an AC0-Xor circuit of polynomial size. To verify this, note that we can compute  $L_A(x)$  using parities at the input level, and the output Xor is on two bits and can be computed in AC0. Also, adding the “shift”  $A$  to  $x$  can be absorbed in the parity gates at the input level.

There remains to boost the probability. Consider the random circuit  $D$  which computes  $t = O(n/\epsilon^2)$  copies of  $C_A$  with independent  $A$ , and then computes approximate majority. Specifically, it outputs 1 if at least  $(1/2 + \epsilon/2)t$  copies output 1, and it outputs 0 if at least  $(1/2 + \epsilon/2)t$  copies output 0. By a Chernoff bound, on any input this circuit has error probability  $< 2^{-n}$ . Hence we can fix the randomness so that it computes correctly every input. Moreover, the approximate majority computation can be done by polynomial-size AC0 circuits for  $\epsilon = 1/\log^{O(1)} n$  [1, 2].

#### 4 A lower bound for computing affine extractors

In this section we prove the following almost  $n^{1.5}$  lower bound for computing affine extractors, even if the error is constant (when the error is exponentially small, a quadratic lower bound can be inferred from the techniques in [13]). While the bound is weaker, the proof appears more elementary than the one in [13].

► **Theorem 24.** *Let  $C : \{0, 1\}^n \rightarrow \{0, 1\}$  be an And-Or-And-Xor circuit that computes an affine extractor for dimension  $n/2$  with error  $1/4$ . Then  $C$  has size  $\Omega(n^{1.5}/\log n)$ .*

► **Lemma 25.** *Let  $t \leq n$  and  $C$  be as in Theorem 24 but with the additional restriction that the fan-in of the middle And gates is  $t$ . Then  $C$  has size  $\Omega(n^2/(t \log n))$ .*

**Proof of Lemma 25.** We assume that a circuit of size  $o(n^2/t \log n)$  exists, and reach a contradiction. Let  $R$  denote the set of Or gates in  $C$ , and let  $A$  denote the set of And gates in  $C$ , excluding the output And gate. Draw a bipartite graph  $G = (R \cup A, E)$  between  $R$  and  $A$ . Each Or gate must have at least  $n/16$  edges, otherwise we can set  $C$  to 0 using  $n/16$  linear restrictions (corresponding to a vector space of dimension  $n - n/16$ ).

Hence there exists some And gate that is connected to at least a  $\frac{n}{16|A|}$  fraction of Or gates in  $R$ . We set it to 1 using at most  $t$  restrictions, eliminate the adjacent Or gates, and consider  $G$  on the resulting affine subspace. We repeat this process  $k$  times for  $k = n/16t$ . Note that we can always find an Or gate with fan-in  $\geq n/16$ , for else we can set the circuit to 0 by setting  $kt + n/16 \leq n/8$  parities.

At the end of the process, the number of Or gates is

$$|R| \left(1 - \frac{n}{16|A|}\right)^k \leq |R| \left(1 - \frac{100t \log n}{n}\right)^{n/16t} \leq n^2/n^3 < 1.$$

This means that the circuit is fixed, which is a contradiction. ◀

► **Lemma 26.** *Let  $t \leq n$  and  $C : \{0, 1\}^n \rightarrow \{0, 1\}$  be an And-Or-And-Xor circuit that computes an affine-extractor for dimension  $n/2$  with error  $1/4$ . Either the size of  $C$  is  $\Omega(nt/\log n)$  or there exists an affine subspace  $H$  of dimension  $\geq 7n/8$  such that  $C|_H$  is an And-Or-And $_t$ -Xor circuit.*

**Proof of Lemma 26.** Let  $A_t$  denote the set of And gates of fan-in greater than  $t$ , excluding the output, and let  $X_t$  denote the set of Xor gates connected to  $A_t$ . Draw the bipartite graph  $G = (A_t \cup X_t, E)$  connecting  $A_t, X_t$ . There is some gate  $x \in X_t$  connected to at least a  $\frac{t}{|X_t|}$  fraction of nodes in  $A_t$  as long as  $|A_t| \geq 1$ . After  $k = n/16$  iterations of setting the XOR gate with the highest degree in  $G$  to 0, if  $|A_t| \geq 1$  we have at most

$$|A_t| \left(1 - \frac{t}{|X_t|}\right)^k \leq |A_t| e^{-\frac{nt}{16|X_t|}}$$

And gates left in  $G$ . If  $|X_t| \geq nt/16 \log n$  we are done, since obviously  $C$  has size  $\geq |X_t|$ . Similarly, if  $|A_t| \geq n^2/16$  we are also done. Otherwise,

$$|A_t| e^{-\frac{nt}{16|X_t|}} \leq |A_t| \frac{1}{n} \leq \frac{n}{16}.$$

So after making  $n/16$  restrictions, we are left with at most  $n/16$  And gates in  $A_t$ . We can make at most  $n/16$  additional restrictions setting them to 0, so that there are no more And gates in  $A_t$  (we might set some to 1 during this process, but that only helps us). We have made at most  $n/16 + n/16 = n/8$  restrictions to reach a subspace  $H$  where  $C|_H$  has no And gates of fan-in  $\geq t$ . ◀

**Proof of Theorem 24.** We combine Lemmas 25 and 26 with the threshold  $t = \sqrt{n}$ . By Lemma 26, either  $C = \Omega(n^{3/2}/\log n)$  or there is some affine subspace  $H$  of dimension  $7n/8$  on which  $C|_H$  has middle And gates of fan-in  $\leq \sqrt{n}$ . In the first case we are done. In the second case, let  $n' = 7n/8$ . Then we can think of  $C|_H$  as a  $(4n'/7, 1/4)$  affine extractor on  $n'$  variables and apply Lemma 25.  $\blacktriangleleft$

---

## References

- 1 Miklós Ajtai.  $\Sigma_1^1$ -formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.
- 2 Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in computational complexity theory*, pages 1–20. Amer. Math. Soc., Providence, RI, 1993.
- 3 Miklos Ajtai and Nathan Linial. The influence of large coalitions. *Combinatorica*, 13:129–145, 1993.
- 4 Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant-depth circuits. *Advances in Computing Research - Randomness and Computation*, 5:199–223, 1989.
- 5 Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC0 MOD 2. In Moni Naor, editor, *ACM Innovations in Theoretical Computer Science conf. (ITCS)*, pages 251–260. ACM, 2014. doi:10.1145/2554797.2554821.
- 6 Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009.
- 7 Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *26th Symposium on Foundations of Computer Science*, pages 408–416, Portland, Oregon, 21–23 October 1985. IEEE.
- 8 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. on Computing*, 39(6):2464–2486, 2010.
- 9 Mark Braverman. Polylogarithmic independence fools AC<sup>0</sup> circuits. *J. of the ACM*, 57(5), 2010.
- 10 Mark Bun, Robin Kothari, and Justin Thaler. Quantum algorithms and approximating polynomials for composed functions with shared inputs. In Timothy M. Chan, editor, *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 662–678. SIAM, 2019. doi:10.1137/1.9781611975482.42.
- 11 Eshan Chattopadhyay, Jesse Goodman, and Jyun-Jie Liao. Affine extractors for almost logarithmic entropy. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 622–633. IEEE, 2021. doi:10.1109/FOCS52979.2021.00067.
- 12 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *ACM Symp. on the Theory of Computing (STOC)*, pages 670–683, 2016.
- 13 Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. AC<sup>0</sup> mod<sub>2</sub> lower bounds for the boolean inner product. *J. Comput. Syst. Sci.*, 97:45–59, 2018.
- 14 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *ACM Innovations in Theoretical Computer Science conf. (ITCS)*, pages 47–58, 2016.
- 15 Gil Cohen and Avishay Tal. Two structural results for low degree polynomials and applications. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 680–709. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.680.
- 16 Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. Bounded independence fools halfspaces. *SIAM J. on Computing*, 39(8):3441–3462, 2010.

- 17 Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Approximations of general independent distributions. In *ACM Symp. on the Theory of Computing (STOC)*, pages 10–16, 1992.
- 18 Michael Ezra and Ron D. Rothblum. Small circuits imply efficient arthur-merlin protocols. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 67:1–67:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.67.
- 19 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In Irit Dinur, editor, *IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 89–98. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.19.
- 20 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
- 21 Oded Goldreich, Emanuele Viola, and Avi Wigderson. On randomness extraction in AC0. In *IEEE Conf. on Computational Complexity (CCC)*, 2015.
- 22 András Hajnal, Wolfgang Maass, Pavel Pudlák, Máriaó Szegedy, and György Turán. Threshold circuits of bounded depth. *J. of Computer and System Sciences*, 46(2):129–154, 1993.
- 23 Johan Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.
- 24 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. on Computing*, 43(5):1699–1708, 2014.
- 25 Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *J. of the ACM*, 64(5):35:1–35:27, 2017. doi:10.1145/3095799.
- 26 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC<sup>0</sup>. In *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 961–972, 2012.
- 27 Stasys Jukna. On graph complexity. *Comb. Probab. Comput.*, 15(6):855–876, 2006. doi:10.1017/S0963548306007620.
- 28 Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.
- 29 Raghu Meka. Explicit resilient functions matching ajtai-linial. In *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1132–1148, 2017.
- 30 Ryan O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
- 31 A. A. Razborov. Formulas of bounded depth in the basis  $\{\&, \oplus\}$  and some combinatorial problems. *Voprosy Kibernet. (Moscow)*, 134:149–166, 1988.
- 32 Alexander Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Akademiya Nauk SSSR. Matematicheskie Zametki*, 41(4):598–607, 1987. English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, 41(4):333–338, 1987.
- 33 Alexander A. Razborov. A simple proof of Bazzi’s theorem. *ACM Transactions on Computation Theory (TOCT)*, 1(1), 2009.
- 34 Petr Savický. Improved boolean formulas for the ramsey graphs. *Random Struct. Algorithms*, 6(4):407–416, 1995. doi:10.1002/rsa.3240060404.
- 35 Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. Available at <http://www.ccs.neu.edu/home/viola/>, 2012.
- 36 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *19th ACM Symp. on the Theory of Computing (STOC)*, pages 77–82. ACM, 1987.
- 37 Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 551–560. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.65.

- 38 Luca Trevisan. Some applications of coding theory in computational complexity. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 347–424. Dept. Math., Seconda Univ. Napoli, Caserta, 2004.
- 39 Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. on Computing*, 36(5):1387–1403, 2007.
- 40 Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.
- 41 Emanuele Viola. Extractors for circuit sources. *SIAM J. on Computing*, 43(2):355–972, 2014.
- 42 Emanuele Viola. Quadratic maps are hard to sample. *ACM Trans. Computation Theory*, 8(4), 2016.
- 43 Emanuele Viola. Special topics in complexity theory. ECCC lecture notes. Also available at <http://www.ccs.neu.edu/home/viola/classes/spepf17.html>, 2017.
- 44 Emanuele Viola. Sampling lower bounds: boolean average-case and permutations. *SIAM J. on Computing*, 49(1), 2020. Available at <http://www.ccs.neu.edu/home/viola/>.
- 45 Emanuele Viola. AC0 unpredictability. *ACM Trans. Computation Theory*, 13(1), 2021.
- 46 Jake Wellens. *Assorted results in boolean function complexity, uniform sampling and clique partitions of graphs*. PhD thesis, Massachusetts Institute of Technology, 2020.

## A Proof of Lemma 14

Let  $f : \{0, 1\}^m \rightarrow \{0, 1\}$  be the function in [3, Theorem 5.1] which is  $m$ -wise  $(\Omega(\alpha m / \log^2 m), O(\alpha))$ -resilient.  $f$  is the And of  $m$  read-once DNFs, so it is a depth-3 circuit of size  $O(m^2 / \log m)$ . We need to show that over any  $r$ -wise distribution  $D$ :

1. The bias of  $f$  is  $\leq O(\alpha)$ .
2. The probability of changing the value of  $f$  by changing at most  $\alpha m / \log^2 m$  bits of  $D$  is  $O(\alpha)$ .

[3, Theorem 5.1] proves 1. and 2. for  $r = m$ . The fact that 1. holds for  $r = \text{poly log } m$  then follows by [9], using that  $\alpha \geq 1/m$ . For the second point we reason as follows. Fix some set  $Q$  of  $\alpha m / \log^2 m$  bad bits. Let  $e(y) : \{0, 1\}^{\overline{Q}} \rightarrow \{0, 1\}$  denote the indicator function of  $f$  not being fixed after assigning  $y$  to the good bits  $\overline{Q}$ . Now we show that  $e(y)$  is computable by an AC0 circuit so we can again apply [9] and reduce to the known resilience under the uniform distribution from [3, Theorem 5.1]. For some partial assignment  $y$  to  $\overline{Q}$ ,  $f$  is not fixed if and only if at least one DNF function is not fixed, and no DNF outputs 0. What remains to show is that for each DNF function, the corresponding indicator  $e'(y)$  can be expressed as an AC0 function. To verify this, note that the DNF is not fixed by  $y$  if every And term that does not intersect with  $Q$  has a bit set to 0, and there is at least one And term intersecting with  $Q$  such that all possible bits set by  $y$  are 1. This computation can be written as a polynomial-size AC0 circuit.



# Hyperbolic Concentration, Anti-Concentration, and Discrepancy

Zhao Song ✉

Adobe Research, Seattle, WA, USA

Ruizhe Zhang ✉

The University of Texas at Austin, TX, USA

---

## Abstract

Chernoff bound is a fundamental tool in theoretical computer science. It has been extensively used in randomized algorithm design and stochastic type analysis. Discrepancy theory, which deals with finding a bi-coloring of a set system such that the coloring of each set is balanced, has a huge number of applications in approximation algorithms design. Chernoff bound [Che52] implies that a random bi-coloring of any set system with  $n$  sets and  $n$  elements will have discrepancy  $O(\sqrt{n \log n})$  with high probability, while the famous result by Spencer [Spe85] shows that there exists an  $O(\sqrt{n})$  discrepancy solution.

The study of hyperbolic polynomials dates back to the early 20th century when used to solve PDEs by Gårding [Går59]. In recent years, more applications are found in control theory, optimization, real algebraic geometry, and so on. In particular, the breakthrough result by Marcus, Spielman, and Srivastava [MSS15] uses the theory of hyperbolic polynomials to prove the Kadison-Singer conjecture [KS59], which is closely related to discrepancy theory.

In this paper, we present a list of new results for hyperbolic polynomials:

- We show two nearly optimal hyperbolic Chernoff bounds: one for Rademacher sum of arbitrary vectors and another for random vectors in the hyperbolic cone.
- We show a hyperbolic anti-concentration bound.
- We generalize the hyperbolic Kadison-Singer theorem [Brä18] for vectors in sub-isotropic position, and prove a hyperbolic Spencer theorem for any constant hyperbolic rank vectors.

The classical matrix Chernoff and discrepancy results are based on determinant polynomial which is a special case of hyperbolic polynomials. To the best of our knowledge, this paper is the first work that shows either concentration or anti-concentration results for hyperbolic polynomials. We hope our findings provide more insights into hyperbolic and discrepancy theories.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** Hyperbolic polynomial, Chernoff bound, Concentration, Discrepancy theory, Anti-concentration

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.10

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2008.09593>

**Funding** *Ruizhe Zhang*: This work was supported by Dana Moshkovitz's NSF Grant CCF-1648712.

**Acknowledgements** We thank the anonymous reviewers for helpful comments. The authors would like to thank Petter Brändén and James Renegar for many useful discussions about the literature of hyperbolic polynomials. The authors would like to thank Yin Tat Lee and James Renegar, Scott Aaronson for encouraging us to work on this topic. The authors would like to thank Dana Moshkovitz for giving comments on the draft.



© Zhao Song and Ruizhe Zhang;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 10; pp. 10:1–10:19



Leibniz International Proceedings in Informatics

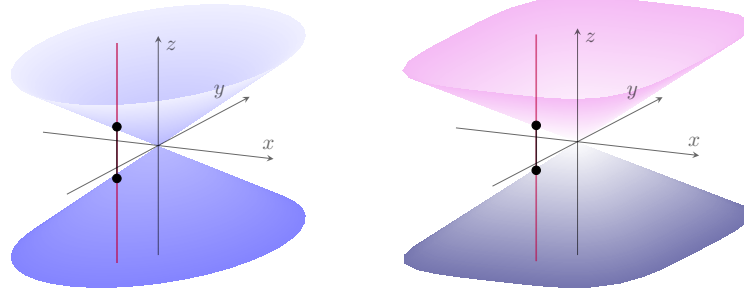
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

The study of concentration of sums of independent random variables dates back to Central Limit Theorems, and hence to de Moivre and Laplace, while modern concentration bounds for sums of random variables were probably first established by Bernstein [15] in 1924. An extremely popular variant now known as Chernoff bounds was introduced by Rubin and published by Chernoff [20] in 1952.

Hyperbolic polynomials are real, multivariate homogeneous polynomials  $p(x) \in \mathbb{R}[x_1, \dots, x_n]$ , and we say that  $p(x)$  is hyperbolic in direction  $e \in \mathbb{R}^n$  if the univariate polynomial  $p(te - x) = 0$  for any  $x$  has only real roots as a function of  $t$  (counting multiplicities). The study of hyperbolic polynomials was first proposed by Gårding in [27] and has been extensively studied in the mathematics community [28, 32, 14, 71]. Some examples of hyperbolic polynomials are as follows:

- Let  $h(x) = x_1 x_2 \cdots x_n$ . It is easy to see that  $h(x)$  is hyperbolic with respect to any vector  $e \in \mathbb{R}_+^n$ .
- Let  $X = (x_{i,j})_{i,j=1}^n$  be a symmetric matrix where  $x_{i,j} = x_{j,i}$  for all  $1 \leq i, j \leq n$ . The determinant polynomial  $h(x) = \det(X)$  is hyperbolic with respect to  $\tilde{I}$ , the identity matrix  $I$  packed into a vector. Indeed,  $h(t\tilde{I} - x) = \det(tI - X)$ , the characteristic polynomial of the symmetric matrix  $X$ , has only real roots by the spectral theorem.
- Let  $h(x) = x_1^2 - x_2^2 - \cdots - x_n^2$ . Then,  $h(x)$  is hyperbolic with respect to  $e = [1 \ 0 \ \cdots \ 0]^\top$ .



■ **Figure 1** The function on the left is  $h(x, y, z) = z^2 - x^2 - y^2$ , which is hyperbolic with respect to  $e = [0 \ 0 \ 1]^\top$ , since any line in this direction always has two intersections, corresponding to the two real roots of  $h(-x, -y, t - z) = 0$ . The function on the right is  $g(x, y, z) = z^4 - x^4 - y^4$ , which is *not* hyperbolic with respect to  $e$ , since it only has 2 intersections but the degree is 4.

Inspired by the eigenvalues of matrix, we can define the hyperbolic eigenvalues of a vector  $x$  as the real roots of  $t \mapsto h(te - x)$ , that is,  $\lambda_{h,e}(x) = (\lambda_1(x), \dots, \lambda_d(x))$  such that  $h(te - x) = h(e) \prod_{i=1}^d (t - \lambda_i(x))$  (see Fact 12). In other words, the hyperbolic eigenvalues of  $x$  are the zero points of the hyperbolic polynomial restricted to a real line through  $x$ . In this paper, we assume that  $h$  and  $e$  are fixed and we just write  $\lambda(x)$  omitting the subscript. Furthermore, similar to the spectral norm of matrix, the *hyperbolic spectral norm* of a vector  $x$  can be defined as

$$\|x\|_h = \max_{i \in [d]} |\lambda_i(x)|. \quad (1)$$

In this work, we study the concentration phenomenon of the roots of hyperbolic polynomials. More specifically, we consider the hyperbolic spectral norm of the sum of randomly signed vectors, i.e.,  $\|\sum_{i=1}^n r_i x_i\|_h$ , where  $r \in \{-1, 1\}^n$  are uniformly random signs and  $\{x_1, x_2, \dots, x_n\}$  are any fixed vectors in  $\mathbb{R}^m$ . This kind of summation has been studied in the following cases:



1. **Scalar case:**  $x_i \in \{-1, 1\}$  and the norm is just the absolute value, i.e.,  $|\sum_{i=1}^n r_i x_i|$ , the scalar version Chernoff bound [20] shows that

$$\Pr_{r \sim \{-1, 1\}^n} \left[ \left| \sum_{i=1}^n r_i x_i \right| > t \right] \leq 2 \exp(-t^2/(2n)),$$

corresponding to the case when  $h(x) = x$  for  $x \in \mathbb{R}$  and the hyperbolic direction  $e = 1$ .

2. **Matrix case:**  $x_i$  are  $d$ -by- $d$  symmetric matrices and the norm is the spectral norm, i.e.,  $\|\sum_{i=1}^n r_i x_i\|$ , the matrix Chernoff bound [86] shows that

$$\Pr_{r \sim \{-1, 1\}^n} \left[ \left\| \sum_{i=1}^n r_i x_i \right\| > t \right] \leq 2d \cdot \exp\left(-\frac{t^2}{2 \|\sum_{i=1}^n x_i^2\|}\right),$$

corresponding to  $h(x) = \det(X)$  and  $e = I$ .

We try to generalize these results to the hyperbolic spectral norm for any hyperbolic polynomial  $h$ , which is recognized as an interesting problem in this field by James Renegar [74].

## 1.1 Our results

In this paper, we can prove the following ‘‘Chernoff-type’’ concentration for hyperbolic spectral norm. We show that, when adding uniformly random signs to  $n$  vectors, the hyperbolic spectral norm of their summation will concentrate with an exponential tail.

► **Theorem 1** (Nearly optimal hyperbolic Chernoff bound for Rademacher sum). *Let  $h$  be an  $m$ -variate, degree- $d$  hyperbolic polynomial with respect to a direction  $e \in \mathbb{R}^m$ . Let  $1 \leq s \leq d$ ,  $\sigma > 0$ . Given  $x_1, x_2, \dots, x_n \in \mathbb{R}^m$  such that  $\text{rank}(x_i) \leq s$  for all  $i \in [n]$  and  $\sum_{i=1}^n \|x_i\|_h^2 \leq \sigma^2$ , where  $\text{rank}(x)$  is the number of nonzero hyperbolic eigenvalues of  $x$ . Then, we have*

$$\mathbb{E}_{r \sim \{\pm 1\}^n} \left[ \left\| \sum_{i=1}^n r_i x_i \right\|_h \right] \leq 2\sqrt{\log(s)} \cdot \sigma.$$

Furthermore, for every  $t > 0$ , and for some fixed constant  $c > 0$ ,

$$\Pr_{r \sim \{\pm 1\}^n} \left[ \left\| \sum_{i=1}^n r_i x_i \right\|_h > t \right] \leq 2 \exp\left(-\frac{ct^2}{\sigma^2 \log(s+1)}\right).$$

We discuss the optimality of Theorem 1 in different cases:

- **Degree-1 case:** When the hyperbolic polynomial’s degree  $d = s = 1$ , the hyperbolic polynomial is  $h(z) = z$ . Then, we have  $\|x\|_h = |x|$  and we get the Hoeffding’s inequality [37]:

$$\Pr_{r \sim \{\pm 1\}^n} \left[ \left| \sum_{i=1}^n r_i x_i \right| > t \right] \leq \exp\left(-\Omega\left(t^2 / \left(\sum_{i=1}^n x_i^2\right)\right)\right).$$

It implies that our result is optimal in this case.

- **A special degree-2 case:**  $h(z) = z_1^2 - z_2^2 - \dots - z_m^2$ . Let  $v_1, \dots, v_n$  be any  $(d-1)$ -dimensional vectors. Then, we define  $x_i := \begin{bmatrix} 0 & v_i \end{bmatrix} \in \mathbb{R}^d$  for  $i \in [n]$ . We know that  $\|x_i\|_h = \|v_i\|_2$ , and Theorem 1 gives the following result:

$$\Pr_{r \sim \{\pm 1\}^n} \left[ \left\| \sum_{i=1}^n r_i v_i \right\|_2 > t \right] \leq \exp(-\Omega(t^2/\sigma^2)),$$

where  $\sigma^2 := \sum_{i=1}^n \|v_i\|_2^2$ , which recovers the dimension-free vector-valued Bernstein inequality [63].

- **Constant degree case:** When  $d > 1$  is a constant, consider  $h$  being the determinant polynomial of  $d$ -by- $d$  matrix. Since  $s \leq d = O(1)$ , we can show that  $\sigma = (\sum_{i=1}^n \|x_i\|^2)^{1/2} = \Theta(\|\sum_{i=1}^n x_i^2\|^{1/2})$ , and Theorem 1 exactly recovers the matrix Chernoff bound [86], which implies that our result is also optimal in this case.
- **Constant rank case:** When all the vectors have constant hyperbolic rank, we still take  $h = \det(X)$ , but  $X_1, \dots, X_n$  are constant rank matrices with arbitrary dimension. In this case, we can obtain a dimension-free matrix concentration inequality:

$$\Pr_{r \sim \{\pm 1\}^n} \left[ \left\| \sum_{i=1}^n r_i X_i \right\| > t \right] \leq 2 \exp(-\Omega(t^2/\sigma^2)).$$

It will beat the general matrix Chernoff bound [86] when  $\sigma$  is not essentially larger than  $\|\sum_{i=1}^n X_i^2\|^{1/2}$ . Thus, Theorem 1 is nearly optimal in this case. However, Theorem 1 is also sub-optimal in this case if we consider the high degree polynomial  $h(z) = \prod_{i=1}^n z_i$ , and  $x_i = e_i \in \mathbb{R}^n$ . Then, we have  $\|x_i\|_h = 1$ , and  $\|\sum_{i=1}^n r_i x_i\|_h = 1$  for any  $r \in \{\pm 1\}^n$ . Therefore, the probability density function of the hyperbolic spectral norm of the Rademacher sum is a delta function<sup>i</sup> in this case. But our concentration result cannot characterize such a sharp transition.

Theorem 1 works for arbitrary vectors in  $\mathbb{R}^m$ . We also consider the maximum and minimum hyperbolic eigenvalues of the sum of random vectors in the hyperbolic cone, which is a generalization of the positive semi-definite (PSD) cone for matrices. Recall that for independent random PSD matrices  $\mathbf{X}_1, \dots, \mathbf{X}_n$  with spectral norm at most  $R$ , let  $\mu_{\max} := \lambda_{\max}(\sum_i \mathbb{E}[\mathbf{X}_i])$ . Then, matrix Chernoff bound for PSD matrices [86] shows that  $\Pr[\lambda_{\max}(\sum_i \mathbf{X}_i) \geq (1+\delta)\mu_{\max}] \leq de^{-\Omega(\delta\mu_{\max})}$  for any  $\delta \geq 0$ . The following theorem gives a hyperbolic version of this result:

► **Theorem 2** (Hyperbolic Chernoff bound for random vectors in hyperbolic cone). *Let  $h$  be an  $m$ -variate, degree- $d$  hyperbolic polynomial with hyperbolic direction  $e \in \mathbb{R}^m$ . Let  $\Lambda_+$  denote the hyperbolic cone<sup>ii</sup> of  $h$  with respect to  $e$ . Suppose  $x_1, \dots, x_n$  are  $n$  independent random vectors with supports in  $\Lambda_+$  such that  $\lambda_{\max}(x_i) \leq R$  for all  $i \in [n]$ . Define the mean of minimum and maximum eigenvalues as  $\mu_{\min} := \sum_{i=1}^n \mathbb{E}[\lambda_{\min}(x_i)]$  and  $\mu_{\max} := \sum_{i=1}^n \mathbb{E}[\lambda_{\max}(x_i)]$ .*

*Then, we have*

$$\Pr \left[ \lambda_{\max} \left( \sum_{i=1}^n x_i \right) \geq (1+\delta)\mu_{\max} \right] \leq d \cdot \left( \frac{(1+\delta)^{1+\delta}}{e^\delta} \right)^{-\mu_{\max}/R} \quad \forall \delta \geq 0,$$

$$\Pr \left[ \lambda_{\min} \left( \sum_{i=1}^n x_i \right) \leq (1-\delta)\mu_{\min} \right] \leq d \cdot \left( \frac{(1-\delta)^{1-\delta}}{e^{-\delta}} \right)^{-\mu_{\min}/R} \quad \forall \delta \in [0, 1].$$

## 1.2 Hyperbolic anti-concentration

Anti-concentration is an interesting phenomenon in probability theory, which studies the opposite perspective of concentration inequalities. A simple example is the standard Gaussian random variable, which has probability at most  $O(\Delta)$  for being in any interval of length  $\Delta$ .

<sup>i</sup> The delta function is defined as  $\delta(x) = \begin{cases} 1 & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases}$

<sup>ii</sup> The hyperbolic cone is a set containing all vectors with non-negative hyperbolic eigenvalues. See Definition 13 for the formal definition.

For Rademacher random variables  $x \sim \{\pm 1\}^d$ , the celebrated Littlewood-Offord theorem [53] states that for any degree-1 polynomial  $p(x) = \sum_{i=1}^d a_i x_i$  with  $|a_i| \geq 1$ , the probability of  $p(x)$  in any length-1 interval is at most  $O(\frac{\log d}{\sqrt{d}})$ . Later, the theorem was improved to  $O(\frac{1}{\sqrt{d}})$  by Erdős [26], and generalized to higher degree polynomials by [22, 79, 61]. From a geometric perspective, the Littlewood-Offord theorem says that the maximum fraction of hypercube points that lay in the boundary of a halfspace  $\mathbf{1}_{\langle a, x \rangle \leq \theta}$  with  $|a_i| \geq 1$  for  $i \in [d]$  is at most  $O(\frac{1}{\sqrt{d}})$ . [67] extended this result from half-space to polytope and [11] further extended to positive spectrahedron.

Following this line of research, we prove the following hyperbolic anti-concentration theorem, which shows that the hyperbolic spectral norm of Rademacher sum of vectors in the hyperbolic cone cannot concentrate within a small interval.

► **Theorem 3** (Hyperbolic anti-concentration theorem, informal). *Let  $h$  be an  $m$ -variate degree- $d$  hyperbolic polynomial with hyperbolic direction  $e \in \mathbb{R}^m$ . Let  $\{x_i\}_{i \in [n]} \subset \Lambda_+$  be a sequence of vectors in the hyperbolic cone such that  $\lambda_{\max}(x_i) \leq \tau$  for all  $i \in [n]$  and  $\sum_{i=1}^n \lambda_{\min}(x_i)^2 \geq 1$ .*

*Then, for any  $y \in \mathbb{R}^m$  and any  $\Delta \geq 20\tau \log d$ , we have*

$$\Pr_{\epsilon \sim \{-1, 1\}^n} \left[ \lambda_{\max} \left( \sum_{i=1}^n \epsilon_i x_i - y \right) \in [-\Delta, \Delta] \right] \leq O(\Delta).$$

From the geometric viewpoint, we can define a “positive hyperbolic-spectrahedron” as the space  $\{\alpha \in \mathbb{R}^n : \lambda_{\max}(\alpha_1 x_1 + \dots + \alpha_n x_n - y) \leq 0\}$ , where  $x_1, \dots, x_n$  are in the hyperbolic cone. Then, Theorem 3 states that the hyperbolic spectral norm of a positive hyperbolic-spectrahedron cannot be concentrated in a small region.

### 1.3 Hyperbolic discrepancy theory

Hyperbolic polynomial is an important tool in the discrepancy theory, which is an important subfield of combinatorics, with many applications in theoretical computer science. Following Meka’s blog post [60], by combining scalar Chernoff bound and union bound, we can easily prove that, for any  $n$  vectors  $x_1, \dots, x_n \in \{-1, 1\}^n$ , there exists  $r \in \{-1, 1\}^n$  such that  $|\langle r, x_i \rangle| \leq O(\sqrt{n \log n})$  for every  $i \in [n]$ . In a celebrated result “Six Standard Deviations Suffice”, Spencer showed that it can be improved to  $|\langle r, x_i \rangle| \leq 6\sqrt{n}$  [83].

For the matrix case, by the matrix Chernoff bound, it follows that for any symmetric matrix  $X_1, \dots, X_n \in \mathbb{R}^{d \times d}$  with  $\|X_i\| \leq 1$ , for uniformly random signs  $r \in \{-1, 1\}^n$ , with high probability,  $\|\sum_{i=1}^n r_i X_i\| \leq O(\sqrt{\log(dn)})$ .

An important open question is, can we shave the  $\log(d)$  factor for *some* choice of the signs?

► **Conjecture 4** (Matrix Spencer Conjecture). *For any symmetric matrices  $X_1, \dots, X_n \in \mathbb{R}^{d \times d}$  with  $\|X_i\| \leq 1$ , there exist signs  $r \in \{-1, 1\}^n$  such that  $\|\sum_{i=1}^n r_i X_i\| = O(\sqrt{n})$ .*

The breakthrough paper by Marcus, Spielman and Srivastava [56] proved the famous Kadison-Singer conjecture [41], which was open for more than half of a century.

► **Theorem 5** (Kadison-Singer, [41, 56]). *Let  $k \geq 2$  be an integer and  $\epsilon$  a positive real number. Let  $x_1, \dots, x_n \in \mathbb{C}^m$  such that  $\|x_i x_i^*\| \leq \epsilon \ \forall i \in [n]$ , and  $\sum_{i=1}^n x_i x_i^* = I$ . Then, there exists a partition  $S_1 \cup S_2 \cup \dots \cup S_k = [n]$  such that  $\|\sum_{i \in S_j} x_i x_i^*\| \leq (\frac{1}{\sqrt{k}} + \sqrt{\epsilon})^2 \ \forall j \in [k]$ .*

The Kadison-Singer theorem implies that for rank-1 matrices  $X_1, \dots, X_n$  with  $\|X_i\| \leq \epsilon$  in isotropic position<sup>iii</sup>, there exists a choice of  $r \in \{-1, 1\}^n$  such that  $\|\sum_{i=1}^n r_i X_i\| \leq O(\sqrt{\epsilon})$ .<sup>iv</sup>

Theorem 5 can be generalized for higher rank matrices by Cohen [21] and Brändén [18] independently. However, their results still need the isotropic condition. On the other hand, Kyng, Luh, and Song [44] proved a stronger version of rank-1 matrix Spencer theorem (Conjecture 4) by showing that when the spectral norm of the sum of the squared matrices (the variance of the random matrices) is bounded, the matrix discrepancy upper bound is at most four deviations. Formal theorem statements will be presented in the full version of this paper [82].

Similar to the scalar and matrix cases, the discrepancy theory can be further generalized to the hyperbolic spectral norm. Brändén [18] proved a hyperbolic Kadison-Singer theorem, which generalizes Theorem 5 to the hyperbolic spectral norm and vectors with arbitrary rank and in isotropic condition. Our first result relaxes the isotropic condition to sub-isotropic:

► **Theorem 6** (Hyperbolic Kadison-Singer with sub-isotropic condition, informal). *Let  $k \geq 2$  be an integer and  $\epsilon, \sigma > 0$ . Suppose  $h$  is hyperbolic with respect to  $e \in \mathbb{R}^m$ , and let  $x_1, \dots, x_n$  be  $n$  vectors in the hyperbolic cone such that*

$$\text{tr}_h[x_i] \leq \epsilon \quad \forall i \in [n], \quad \text{and} \quad \left\| \sum_{i=1}^n x_i \right\|_h \leq \sigma. \quad (2)$$

where  $\text{tr}_h[x] := \sum_{i=1}^d \lambda_i(x)$ . Then, there exists a partition  $S_1 \cup S_2 \cup \dots \cup S_k = [n]$  such that for all  $j \in [k]$ ,

$$\left\| \sum_{i \in S_j} x_i \right\|_h \leq \left( \sqrt{\epsilon} + \sqrt{\sigma/k} \right)^2.$$

Theorem 6 implies the high rank case of [56] result (Theorem 5) without the isotropic condition. We note that there is a naive approach to relax the isotropic condition in [56, 18]’s results by adding several small dummy vectors to make the whole set in isotropic position. (See [30] for more details.) However, Theorem 6 is slightly better than this approach, since the naive approach will increase the number of vectors which results in a worse bound.

Theorem 6 also implies the following hyperbolic discrepancy result:

► **Corollary 7** (Hyperbolic discrepancy for sub-isotropic vectors). *Let  $0 < \epsilon \leq \frac{1}{2}$ . Suppose  $h \in \mathbb{R}[z_1, \dots, z_m]$  is hyperbolic with respect to  $e \in \mathbb{R}^m$ , and let  $x_1, \dots, x_n \in \Lambda_+(h, e)$  that satisfy Eq. (2). Then, there exist signs  $r \in \{-1, 1\}^n$  such that*

$$\left\| \sum_{i=1}^n r_i x_i \right\|_h \leq 2\sqrt{\epsilon(2\sigma - \epsilon)}.$$

We note that this result is incomparable with [44] due to the following reasons: 1) [44] only works for rank-1 matrices while our result holds for arbitrary rank vectors in the hyperbolic cone; 2) the upper bound of [44] depends on  $\|\sum_{i=1}^n X_i^2\|^{1/2}$  while our result depends on the hyperbolic trace and spectral norm of the sum of vectors.

To obtain a hyperbolic discrepancy upper bound for arbitrary vectors (as in the case of Conjecture 4), we can apply hyperbolic Chernoff bound (Theorem 1) and get the following discrepancy result which holds with high probability:

<sup>iii</sup> Isotropic means  $X_1 + \dots + X_n = I$ .

<sup>iv</sup> For more details and consequences of the Kadison-Singer theorem, we refer the readers to [19, 58].

► **Corollary 8.** *Let  $h$  be a degree- $d$  hyperbolic polynomial with respect to  $e \in \mathbb{R}^m$ . We are given vectors  $x_1, x_2, \dots, x_n \in \mathbb{R}^m$  such that  $\|x_i\|_h \leq 1$  and  $\text{rank}(x_i) \leq s$  for all  $i \in [n]$  and some  $s \in \mathbb{N}_+$ . Then for uniformly random signs  $r \sim \{-1, 1\}^n$ ,*

$$\left\| \sum_{i=1}^n r_i x_i \right\|_h \leq O(\sqrt{n \log(s+1)})$$

*holds with probability at least 0.99.*

This result may not be tight when the ranks of the input vectors are large. It is thus interesting to study whether we can do better to improve the  $\sqrt{\log d}$  factor in the non-constructive case. We thus conjecture the following hyperbolic discrepancy bound:

► **Conjecture 9** (Hyperbolic Spencer Conjecture). *We are given vectors  $x_1, x_2, \dots, x_n \in \mathbb{R}^m$  and a degree  $d$  hyperbolic polynomial  $h \in \mathbb{R}[z_1, \dots, z_m]$  with respect to  $e \in \mathbb{R}^m$ , where  $\|x_i\|_h \leq 1$  for all  $i \in [n]$ . Then, there exist signs  $r \in \{-1, 1\}^n$ , such that*

$$\left\| \sum_{i=1}^n r_i x_i \right\|_h \leq O(\sqrt{n}).$$

Note that Conjecture 9 is more general than the matrix Spencer conjecture (Conjecture 4). And for constant degree  $d$  or constant maximum rank  $s$ , this conjecture is true by Corollary 8.

## 1.4 Related work

### Chernoff-type bounds

There is a long line of work generalizing the classical scalar Chernoff-type bounds to the matrix Chernoff-type bound [77, 5, 78, 85, 55, 29, 45, 66, 10, 40]. [77, 78] showed a Chernoff-type concentration of spectral norm of matrices which are the outer product of two random vectors. [5] first used Laplace transform and Golden-Thompson inequality [31, 84] to prove a Chernoff bound for general random matrices. It was improved by [85] and [68] independently. [55] proved a series of matrix concentration results via Stein's method of exchangeable pairs. Our work further extends this line of research from matrix to hyperbolic polynomials and can fully recover the result of [5]. On the other hand, [29] showed an expander matrix Chernoff bound. [45] prove a new matrix Chernoff bound for Strongly Rayleigh distributions.

### Hyperbolic polynomials

The concept of hyperbolic polynomials was originally studied in the field of partial differential equations [27, 39, 42]. Güler [32] first studied the hyperbolic optimization (hyperbolic programming), which is a generalization of LP and SDP. Later, a few algorithms [71, 64, 75, 72, 65, 73] were designed for hyperbolic programming. On the other hand, a lot of recent research focused on the equivalence between hyperbolic programming and SDP, which is closely related to the “Generalized Lax Conjecture” and its variants [36, 52, 17, 43, 80, 7, 69]. In addition to the hyperbolic programming, hyperbolic polynomial is a key component in resolving Kadison-Singer problem [56, 18] and constructing bipartite Ramanujan graphs [57]. Gurvits [34, 35] proved some Van der Waerden/Schrijver-Valiant like conjectures for hyperbolic polynomials, giving sharp bounds for the capacity of polynomials. [81] gave an approach to certify the non-negativity of polynomials via hyperbolic programming, generalizing the Sum-of-Squares method.

### Discrepancy theory

For discrepancy theory, we give a few literature in Section 1.3 and we provide more related work here. For Kadison-Singer problem, after the breakthrough result [56], Anari and Oveis Gharan [8] generalized it for Strongly Rayleigh distributions. Alishahi and Barzegar [6] extended the “paving conjecture” to real stable polynomials<sup>v</sup>. Zhang and Zhang [89] further relaxed the determinant polynomial in [8] and [44] to homogeneous real-stable polynomials. More recently, [38, 23] proved some special cases of the matrix Spencer conjecture. For algorithmic results, Bansal [12] proposed the first constructive version of partial coloring for discrepancy minimization. Based on this work, more approaches [54, 76, 51, 25, 13, 24] were discovered in recent years. For applications of the discrepancy theory, [8, 9] used the Strongly Rayleigh version of Kadison-Singer theorem to improve the integrality gap of the Asymmetric Traveling Salesman Problem. [47] used the rank-1 matrix Spencer theorem in [44] to obtain a two-sided spectral rounding result. For more applications, we refer to the excellent book by Matousek [59].

## 1.5 Technique overview

In this section, we provide a proof overview of our results. We first show how prove hyperbolic Chernoff bounds by upper bounding each polynomial moment. After that, we show how to apply our new concentration inequality to prove hyperbolic anti-concentration. Finally, we show how to relax the isotropic condition in [18], and also how to get a more general discrepancy result via hyperbolic concentration.

### 1.5.1 Our technique for hyperbolic Chernoff bound for Rademacher sum

The main idea of our proof of hyperbolic Chernoff bound is to upper bound the polynomial moments.

By definition, the hyperbolic spectral norm of  $X$  is the  $\ell_\infty$  norm of the eigenvalues  $\lambda(X)$ . Inspired by the proof of the matrix Chernoff bound by Tropp [87], we can consider the  $\ell_{2q}$  norm of  $\lambda(X)$ , for  $q \geq 1$ . When the hyperbolic polynomial  $h$  is the determinant polynomial, this norm is just the Schatten- $2q$  norm of matrices. For general hyperbolic polynomials, we define hyperbolic- $2q$  norm as  $\|x\|_{h,2q} := \|\lambda(x)\|_{2q}$ . By the result of [14], hyperbolic- $2q$  norm is actually a norm in  $\mathbb{R}^m$ . And the following inequality shows the connection between a hyperbolic spectral norm and hyperbolic- $2q$  norm:

$$\mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_h] \leq \left( \mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_{h,2q}^{2q}] \right)^{1/(2q)}.$$

In order to compute  $\|X\|_{h,2q}^{2q} = \sum_{i=1}^{\text{rank}(X)} \lambda_i(X)^{2q}$ , we use a deep result about hyperbolic polynomials: the Helton-Vinnikov Theorem [36], which proved a famous conjecture by Lax [48], to translate between hyperbolic polynomials and matrices. The theorem is stated as follows.

► **Theorem 10** ([36]). *Let  $f \in \mathbb{R}[x, y, z]$  be hyperbolic with respect to  $e = (e_1, e_2, e_3) \in \mathbb{R}^3$ . Then there exist symmetric real matrices  $A, B, C \in \mathbb{R}^{d \times d}$  such that  $f = \det(xA + yB + zC)$  and  $e_1A + e_2B + e_3C \succ 0$ .*

<sup>v</sup> A polynomial is real stable if it is hyperbolic with respect to every  $e \in \mathbb{R}_{\geq 0}^n$ .

Gurvits [33] proved a corollary (Corollary 22) that for any  $m$ -variate hyperbolic polynomial  $h$ , and  $x, y \in \mathbb{R}^m$ , there exist two symmetric matrices  $A, B \in \mathbb{R}^{d \times d}$  such that for any  $a, b \in \mathbb{R}$ ,  $\lambda(ax + by) = \lambda(aA + bB)$ , where the left-hand side means the hyperbolic eigenvalues of the vector  $ax + by$  and the right-hand side means the eigenvalues of the matrix  $aA + bB$ .

Therefore, we try to separate and consider one random variable  $r_i$  at a time. We first consider the expectation over  $r_1$ . By conditional expectation, let  $X_2 := \sum_{i=2}^n r_i x_i$  and we have

$$\mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_{h,2q}^{2q}] = \mathbb{E}_{r_2, \dots, r_n \sim \{\pm 1\}} \left[ \mathbb{E}_{r_1 \sim \{\pm 1\}} [\|r_1 x_1 + X_2\|_{h,2q}^{2q}] \right],$$

By Corollary 22, there exist two matrices  $A_1, B_1$  such that  $\lambda(r_1 x_1 + X_2) = \lambda(r_1 A_1 + B_1)$  holds for any  $r_1$ . And it follows that

$$\mathbb{E}_{r_1 \sim \{\pm 1\}} [\|r_1 x_1 + X_2\|_{h,2q}^{2q}] = \mathbb{E}_{r_1 \sim \{\pm 1\}} [\|r_1 A_1 + B_1\|_{2q}^{2q}].$$

It becomes much easier to compute the expected Schatten- $2q$  norm of matrices. We can prove that

$$\mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_{h,2q}^{2q}] \leq \sum_{k_1=0}^q \binom{2q}{2k_1} \|x_1\|_h^{2k_1} \cdot \mathbb{E}_{r_2, \dots, r_n} [\|X_2\|_{h,2q-2k_1}^{2q-2k_1}].$$

Now, we can iterate this process for the remaining expectation  $\mathbb{E}_{r_2, \dots, r_n} [\|X_2\|_{h,2q-2k_1}^{2q-2k_1}]$ . After  $n-1$  iterations, we get that

$$\left( \mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_{h,2q}^{2q}] \right)^{1/(2q)} \leq \sqrt{2q-1} \cdot s^{1/(2q)} \cdot \sigma, \quad (3)$$

where  $\sigma^2 = \sum_{i=1}^n \|x_i\|_h^2$  and  $s$  is the maximum rank of  $x_1, \dots, x_n$ . Then, by taking  $q := \log(s)$  and  $\|X\|_h \leq \|X\|_{h,2q}^{2q}$ , we get the desired upper bound for the expectation  $\mathbb{E}_{r \sim \{\pm 1\}^n} [\|\sum_{i=1}^n r_i x_i\|_h]$  in Theorem 1.

To obtain the concentration probability inequality, We can apply the result of Ledoux and Talagrand [49] for the concentration of Rademacher sums in a normed linear space, which will imply:

$$\Pr_{r \sim \{\pm 1\}^n} [\|X\|_h > t] \leq 2 \exp \left( -t^2 / \left( 32 \mathbb{E}_{r \sim \{\pm 1\}^n} [\|X\|_h^2] \right) \right). \quad (4)$$

However, we need to verify that the hyperbolic spectral norm  $\|\cdot\|_h$  is indeed a norm, which follows from the result of Gårding [28]. Since by Khinchin-Kahane inequality ([82, Theorem A.16]) the second moment of  $\|X\|_h$  can be upper-bounded via the first moment. Hence, we can put our expectation upper bound into Eq. (4) and have

$$\Pr_{r \sim \{\pm 1\}^n} [\|X\|_h > t] \leq C_1 \exp \left( -\frac{C_2 t^2}{\sigma^2 \log(s+1)} \right),$$

for constants  $C_1, C_2 > 0$ , and hence Theorem 1 is proved. We defer the formal proof in the full version [82, Section B].

### 1.5.2 Our technique for hyperbolic Chernoff bound for positive vectors

We can use similar techniques in the previous section to prove Theorem 2.

## 10:10 Hyperbolic Concentration, Anti-Concentration, and Discrepancy

For any random vectors  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \Lambda_+$ , we may assume  $\|\mathbf{x}_i\|_h \leq 1$ . Using the Taylor expansion of the mgf, we can show that:

$$\Pr \left[ \lambda_{\max} \left( \sum_{i=1}^n \mathbf{x}_i \right) \geq t \right] \leq \inf_{\theta > 0} e^{-\theta t} \cdot \sum_{q \geq 0} \frac{\theta^q}{q!} \mathbb{E} \left[ \left\| \sum_{i=1}^n \mathbf{x}_i \right\|_{h,q}^q \right]. \quad (5)$$

Then, for the  $q$ -th moment, we separate  $\mathbf{x}_1$  and  $\sum_{i=2}^n \mathbf{x}_i$  and have

$$\mathbb{E}_{\geq 1} \left[ \left\| \sum_{i=1}^n \mathbf{x}_i \right\|_{h,q}^q \right] = \mathbb{E}_{\geq 2} \mathbb{E}_1 [\text{tr}[(A_1 + B_1)^q]],$$

where  $A_1$  and  $B_1$  are two PSD matrices obtained via Gurvits's result (Corollary 22) such that  $A_1$  depends on  $\mathbf{x}_1$  and  $B_1$  depends on  $\mathbf{x}_2, \dots, \mathbf{x}_n$ . The next step is different from the case of Rademacher sum, since we cannot drop half of the terms by the distribution of  $\mathbf{x}_1$ . Instead, we can fully expand the matrix products in the trace and use Horn's inequality to upper bound the eigenvalue products. We have

$$\mathbb{E}_{\geq 2} \mathbb{E}_1 [\text{tr}[(A(\mathbf{x}_1) + B)^q]] \leq \mathbb{E}_1 \left[ \sum_{k_1=0}^q \binom{q}{k_1} \lambda_{\max}(\mathbf{x}_1)^{k_1} \cdot \mathbb{E}_{\geq 2} \left[ \left\| \sum_{i=2}^n \mathbf{x}_i \right\|_{h,q-k_1}^{q-k_1} \right] \right].$$

By repeating this process, we finally have

$$\mathbb{E} \left[ \left\| \sum_{i=1}^n \mathbf{x}_i \right\|_{h,q}^q \right] \leq d \cdot \mathbb{E} \left[ \left( \sum_{i=1}^n \|\mathbf{x}_i\|_h \right)^q \right].$$

Then, we put the above upper bound into Eq. (5), which gives:

$$\Pr \left[ \lambda_{\max} \left( \sum_{i=1}^n \mathbf{x}_i \right) \geq t \right] \leq \inf_{\theta > 0} e^{-\theta t} \cdot d \cdot \prod_{i=1}^n \mathbb{E} \left[ e^{\theta \|\mathbf{x}_i\|_h} \right].$$

Now, we use some similar calculations in the matrix case [85] to prove that

$$\Pr \left[ \lambda_{\max} \left( \sum_{i=1}^n \mathbf{x}_i \right) \geq t \right] \leq \inf_{\theta > 0} d \cdot \exp(-\theta t + (e^\theta - 1)\mu_{\max}).$$

By taking  $\theta := \log(t/\mu_{\max})$  and  $t := (1 + \delta)\mu_{\max}$ , we get that

$$\Pr \left[ \lambda_{\max} \left( \sum_{i=1}^n \mathbf{x}_i \right) \geq (1 + \delta)\mu_{\max} \right] \leq d \cdot \left( \frac{(1 + \delta)^{1+\delta}}{e^\delta} \right)^{-\mu_{\max}} \quad (6)$$

For the minimum eigenvalue case, we can define  $\mathbf{x}'_i := e - \mathbf{x}_i$  for  $i \in [n]$ . Then, by the property of hyperbolic eigenvalues (Fact 19) and the assumption that  $\|\mathbf{x}_i\|_h \leq 1$ , we know that  $\mathbf{x}'_i$  are also in the hyperbolic cone and  $\lambda_{\max}(\mathbf{x}'_i) = 1 - \lambda_{\min}(\mathbf{x}_i)$ . Therefore, we can obtain the Chernoff bound for the minimum eigenvalue of  $\mathbf{x}$  by applying Eq. (6) with  $\mathbf{x}'_i$ . We defer the formal proof in the full version [82, Section C].

### 1.5.3 Our technique for hyperbolic anti-concentration

In this part, we will show how to prove the hyperbolic anti-concentration result (Theorem 3) via the hyperbolic Chernoff bound for vectors in the hyperbolic cone (Theorem 2).



In [67], they studied the unate functions on hypercube  $\{-1, 1\}^n$ , which is defined as the function being increasing or decreasing with respect to any one of the coordinates. Then, they showed that the Rademacher measure of a unate function is determined by the expansion of its indicator set in hypercube. In particular, for the maximum hyperbolic eigenvalue, it is easy to see that the indicator function  $\left[\lambda_{\max}\left(\sum_{i=1}^n \epsilon_i x_i^j - y_j\right) \in [-\Delta, \Delta]\right]$  is unate when  $x_i \in \Lambda_+$ . Hence, we can show the anti-concentration inequality by studying the expansion in the hypercube, which by [11], is equivalent to lower-bound the minimum eigenvalue of each vector. However, for the initial input  $x_i$ , we only assume that  $\sum_{i=1}^n \lambda_{\min}(x_i)^2 \geq 1$ , but we need a  $\Omega(\frac{1}{\sqrt{\log d}})$  lower bound for each  $x_i$  to prove the theorem. To amplify the minimum eigenvalue, we follow the proof in [11] that uses a random hash function to randomly assign the input vectors into some buckets and considers the sum of the vectors in each bucket as the new input. They proved that the “bucketing” will not change the distribution. Then, we can use Theorem 2 to lower bound the minimum hyperbolic eigenvalue of each bucket, which is a sum of independent random vectors in the hyperbolic cone. Hence, we get that

$$\Pr \left[ \lambda_{\min} \left( \sum_{i=1}^n z_{i,j} x_i \right) \leq \Omega\left(\frac{1}{\sqrt{\log d}}\right) \right] \leq \frac{1}{10},$$

which  $z_{i,j} \in \{0, 1\}$  is a random variable indicating that  $x_i$  is hashed to the  $j$ -th bucket. Then, by the standard Chernoff bound for negatively associated random variables, we can prove that most of the buckets have large minimum eigenvalues, which concludes the proof of the hyperbolic anti-concentration theorem. We defer the formal proof in the full version [82, Section D].

### 1.5.4 Our technique for hyperbolic discrepancy

To relax the isotropic condition in [18], we basically follow their proof. The high-level idea is to construct a compatible family of polynomials<sup>vi</sup> such that the probability in the hyperbolic Kadison-Singer problem (Theorem 6) can be upper-bounded by the largest root of the expected polynomial of the family, which can be further upper-bounded by the largest root of the mixed hyperbolic polynomial  $h[v_1, \dots, v_n] \in \mathbb{R}[x_1, \dots, x_m, y_1, \dots, y_n]$ , defined as  $h[v_1, \dots, v_n] := \prod_{i=1}^m (1 - y_i D_{v_i}) h(x)$ , where  $D_{v_i}$  is the directional derivative with respect to  $v_i$ . In particular, we can consider the roots of the linear restriction  $h[v_1, \dots, v_n](te + \mathbf{1}) \in \mathbb{R}[t]$ . Then, using Gårding’s result [28] on hyperbolic cone, we know that the largest root equals the minimum  $\rho > 0$  such that the vector  $\rho e + \mathbf{1}$  is in the hyperbolic cone  $\Gamma_+$  of  $h[v_1, \dots, v_n]$ , which can be upper-bounded via similar techniques in [56, 44] to iteratively add each vector  $v_i$  while keeping the sum in the hyperbolic cone. Our key observation is that the proof in [18] essentially proved that

$$\frac{\epsilon \mu e + \left(1 - \frac{1}{n}\right) \delta \sum_{i=1}^n v_i}{1 + \frac{\mu-1}{n}} + \mathbf{1} \in \Gamma_+$$

holds for any vectors  $v_i \in \Lambda_+$ . Hence, once we assume that  $\|\sum_{i=1}^n v_i\|_h \leq \sigma$ , then by the convexity of the hyperbolic cone, we get that  $\rho \leq \frac{(\epsilon \mu + (1 - \frac{1}{n}) \delta \sigma)}{1 + \frac{\mu-1}{n}}$ , which will imply the upper bound in Theorem 6. We defer the formal proof in the full version [82, Section E].

<sup>vi</sup> The compatible family of polynomials is closely related to the interlacing family in [56, 57]. See [82, Definition E.16].

To obtain the discrepancy result for arbitrary vectors (Corollary 8), we can use the hyperbolic Chernoff bound for Rademacher sum (Theorem 1) to derive the discrepancy upper bound. For any vectors  $x_1, \dots, x_n$  with maximum rank  $s$ , by setting  $t = O(\sigma\sqrt{\log s})$  in Theorem 1, we get that  $\|\sum_{i=1}^n r_i x_i\|_h \leq O(\sigma\sqrt{\log s})$  holds with high probability for uniformly random signs  $r \sim \{\pm 1\}^n$ .

## 1.6 Discussion and Open problems

In this paper, we initiate the study of concentration with respect to the hyperbolic spectral norm, and we generalize several classical concentration and anti-concentration results to the hyperbolic polynomial setting. Our results are closely related to the discrepancy theory and pseudorandomness. We provide some open problems in below.

### Tighter hyperbolic Chernoff bound?

Our current result has a worse dependence on the variance  $\sigma^2$  than the matrix Chernoff bound [86]. Can we match the results when  $h = \det(X)$ ? We note that there is a limitation for using the techniques like Golden-Thompson inequality and Lieb's theorem, which were used in [68, 85] to improve the original matrix Chernoff bound [5], to tighten our result. Because for any symmetric matrix  $X$ , we can define a mapping such that  $\phi(X)$ 's eigenvalues are the  $p$ -th power of  $X$ 's eigenvalues for any  $p > 0$ , where the mapping is just  $X^p$ . However, we cannot find such a mapping for vectors with respect to the hyperbolic eigenvalues. Some new techniques may be required to get a hyperbolic Chernoff bound matching the matrix results.

### Resolving the hyperbolic Spencer conjecture?

Inspired by the matrix Spencer conjecture (due to Meka [60]), we came up with a more general conjecture for hyperbolic discrepancy. Can we prove or disprove this conjecture? It is also interesting to study the connection between hyperbolic Spencer conjecture and the generalized Lax conjecture [36, 52, 17, 43, 80, 7, 69]. If we assume the matrix Spencer conjecture and the generalized Lax conjecture, can we prove the hyperbolic Spencer conjecture? On the other hand, in a very recent work by Reis and Rothvoss [70], they conjectured a weaker matrix Spencer by considering the Schatten- $p$  norm of matrices. We can also define such an  $\ell_p$  version of the hyperbolic Spencer conjecture by looking at the  $\ell_p$ -norm of hyperbolic eigenvalues (the hyperbolic- $p$  norm). Any progress towards the  $\ell_p$ -hyperbolic Spencer conjecture will provide more insights in matrix and hyperbolic discrepancy theory.

### Fooling hyperbolic cone?

One of the results in this paper is showing an anti-concentration inequality with respect to the hyperbolic spectral norm, which generalizes the results in [67, 11]. They actually combined the anti-concentration results with the Meka-Zuckerman [62] framework to construct PRGs fooling polytopes/positive spectrahedrons. Hence, an open question in complexity theory and pseudorandomness is: can we apply the hyperbolic anti-concentration inequality to construct a PRG fooling positive hyperbolic-spectrahedrons, or even hyperbolic cones?

### Concentration of random tensors?

Tensor concentration is another natural generalization of matrix concentration. Although there have been a large number of works on this problem [46, 50, 4, 3, 88, 2], it is still unclear what is the optimal concentration bound for the Euclidean norm of random tensor  $X \in \mathbb{R}^{n^d}$ , even in the simple case when  $X = x_1 \otimes \cdots \otimes x_d$  for random vectors  $x_1, \dots, x_d \in \mathbb{R}^n$ . On the other hand, people also care about whether random tensors are well-conditioned, which is more related to TCS problems including tensor decompositions and learning Gaussian mixtures. The current results [88, 1, 16] have a large gap between the matrix case. For these tensor concentration problems, is it possible to study them via hyperbolic polynomials and obtain tighter bounds?

---

### References

- 1 Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed–muller codes for random erasures and errors. *IEEE Transactions on Information Theory*, 61(10):5229–5252, 2015.
- 2 Radosław Adamczak, Rafał Łatała, and Rafał Meller. Moments of gaussian chaoses in banach spaces. *Electronic Journal of Probability*, 26:1–36, 2021.
- 3 Radosław Adamczak and Paweł Wolff. Concentration inequalities for non-lipschitz functions with bounded derivatives of higher order. *Probability Theory and Related Fields*, 162(3):531–586, 2015.
- 4 Radosław Adamczak and Rafał Łatała. Tail and moment estimates for chaoses generated by symmetric random variables with logarithmically concave tails. *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*, 48(4):1103–1136, 2012. doi:10.1214/11-AIHP441.
- 5 Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002.
- 6 Kasra Alishahi and Milad Barzegar. Paving property for real stable polynomials and strongly rayleigh processes. *arXiv preprint*, 2020. arXiv:2006.13923.
- 7 Nima Amini. Spectrahedrality of hyperbolicity cones of multivariate matching polynomials. *Journal of Algebraic Combinatorics*, 50(2):165–190, 2019.
- 8 Nima Anari and Shayan Oveis Gharan. The kadison-singer problem for strongly rayleigh measures and applications to asymmetric tsp. *arXiv preprint*, 2014. arXiv:1412.1143.
- 9 Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 20–39. IEEE, 2015.
- 10 Richard Aoun, Marwa Banna, and Pierre Youssef. Matrix Poincaré inequalities and concentration, 2020. In *Advances in Mathematics*, volume 371. arXiv:1910.13797.
- 11 Srinivasan Arunachalam and Penghui Yao. Positive spectrahedrons: Geometric properties, invariance principles and pseudorandom generators. *arXiv preprint*, 2021. arXiv:2101.08141.
- 12 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 3–10. IEEE, 2010.
- 13 Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The gram-schmidt walk: a cure for the banaszczyk blues. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 587–597, 2018.
- 14 Heinz H Bauschke, Osman Güler, Adrian S Lewis, and Hristo S Sendov. Hyperbolic polynomials and convex analysis. *Canadian Journal of Mathematics*, 53(3):470–488, 2001.
- 15 Sergei Bernstein. On a modification of chebyshev’s inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.
- 16 Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 594–603, 2014.

- 17 Petter Brändén. Hyperbolicity cones of elementary symmetric polynomials are spectrahedral. *Optimization Letters*, 8(5):1773–1782, 2014.
- 18 Petter Brändén. Hyperbolic polynomials and the Kadison-Singer problem. *arXiv preprint*, 2018. [arXiv:1809.03255](#).
- 19 Peter G Casazza and Janet C Tremain. Consequences of the Marcus/Spielman/Srivastava solution of the Kadison-Singer problem. In *New Trends in Applied Harmonic Analysis*, pages 191–213. Springer, 2016.
- 20 Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.
- 21 Michael Cohen. Improved spectral sparsification and Kadison-Singer for sums of higher-rank matrices. In *Banff International Research Station for Mathematical Innovation and Discovery*. <https://open.library.ubc.ca/cIRcle/collections/48630/items/1.0340957>, 2016.
- 22 Kevin P Costello, Terence Tao, and Van Vu. Random symmetric matrices are almost surely nonsingular. *Duke Mathematical Journal*, 135(2):395–413, 2006.
- 23 Daniel Dadush, Haotian Jiang, and Victor Reis. A new framework for matrix discrepancy: Partial coloring bounds via mirror descent. *arXiv preprint*, 2021. [arXiv:2111.03171](#).
- 24 Daniel Dadush, Aleksandar Nikolov, Kunal Talwar, and Nicole Tomczak-Jaegermann. Balancing vectors in any norm. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10. IEEE, 2018.
- 25 Ronen Eldan and Mohit Singh. Efficient algorithms for discrepancy minimization in convex sets. *Random Structures & Algorithms*, 53(2):289–307, 2018.
- 26 Paul Erdős. On a lemma of littlewood and offord. *Bulletin of the American Mathematical Society*, 51(12):898–902, 1945.
- 27 Lars Gårding. Linear hyperbolic partial differential equations with constant coefficients. *Acta Mathematica*, 85:1–62, 1951.
- 28 Lars Gårding. An inequality for hyperbolic polynomials. *Journal of Mathematics and Mechanics*, pages 957–965, 1959.
- 29 Ankit Garg, Yin-Tat Lee, Zhao Song, and Nikhil Srivastava. A matrix expander chernoff bound, 2018. In *STOC*. [arXiv:1704.03864](#).
- 30 Shayan Oveis Gharan. Proof of kadison-singer conjecture and the extensions, 2015.
- 31 Sidney Golden. Lower bounds for the helmholtz function. *Physical Review*, 137(4B):B1127, 1965.
- 32 Osman Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22(2):350–377, 1997.
- 33 Leonid Gurvits. Combinatorics hidden in hyperbolic polynomials and related topics. *arXiv preprint*, 2004. [arXiv:math/0402088](#).
- 34 Leonid Gurvits. Hyperbolic polynomials approach to van der waerden/schrijver-valiant like conjectures: sharper bounds, simpler proofs and algorithmic applications. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 417–426, 2006.
- 35 Leonid Gurvits. Van der waerden/schrijver-valiant like conjectures and stable (aka hyperbolic) homogeneous polynomials: one theorem for all. *arXiv preprint*, 2007. [arXiv:0711.3496](#).
- 36 J William Helton and Victor Vinnikov. Linear matrix inequality representation of sets. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 60(5):654–674, 2007.
- 37 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.
- 38 Samuel B Hopkins, Prasad Raghavendra, and Abhishek Shetty. Matrix discrepancy from quantum communication. *arXiv preprint*, 2021. [arXiv:2110.10099](#).
- 39 L Hormander. The analysis of linear partial differential operators ii. *Grundlehren*, 257, 1983.
- 40 He Jia, Aditi Laddha, Yin Tat Lee, and Santosh S Vempala. Reducing isotropy and volume to KLS: An  $O^*(n^3\psi^2)$  volume algorithm. *arXiv preprint*, 2020. [arXiv:2008.02146](#).

- 41 Richard V Kadison and Isadore M Singer. Extensions of pure states. *American journal of mathematics*, 81(2):383–400, 1959.
- 42 N.V. Krylov. On the general notion of fully nonlinear second-order elliptic equations. *Transactions of the American Mathematical Society*, 347(3):857–895, 1995.
- 43 Mario Kummer, Daniel Plaumann, and Cynthia Vinzant. Hyperbolic polynomials, interlacers, and sums of squares. *Mathematical Programming*, 153(1):223–245, 2015.
- 44 Rasmus Kyng, Kyle Luh, and Zhao Song. Four deviations suffice for rank 1 matrices, 2020. In *Advances in Mathematics*. [arXiv:1901.06731](#).
- 45 Rasmus Kyng and Zhao Song. A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees, 2018. In *FOCS*. [arXiv:1810.08345](#).
- 46 Rafał Łatała. Estimates of moments and tails of gaussian chaoses. *The Annals of Probability*, 34(6):2315–2331, 2006.
- 47 Lap Chi Lau and Hong Zhou. A spectral approach to network design. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 826–839, 2020.
- 48 Peter D Lax. Differential equations, difference equations and matrix theory. Technical report, New York Univ., New York. Atomic Energy Commission Computing and Applied, 1957.
- 49 Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.
- 50 Joseph Lehec. Moments of the gaussian chaos. In *Séminaire de Probabilités XLIII*, pages 327–340. Springer, 2011.
- 51 Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss. Deterministic discrepancy minimization via the multiplicative weight update method. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 380–391. Springer, 2017.
- 52 Adrian Lewis, Pablo Parrilo, and Motakuri Ramana. The lax conjecture is true. *Proceedings of the American Mathematical Society*, 133(9):2495–2499, 2005.
- 53 John Edensor Littlewood and Albert Cyril Offord. On the number of real roots of a random algebraic equation (iii). *Rec. Math. [Mat. Sbornik] N.S.*, 12(3):277–286, 1943.
- 54 Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM Journal on Computing*, 44(5):1573–1582, 2015.
- 55 Lester Mackey, Michael I Jordan, Richard Y Chen, Brendan Farrell, and Joel A Tropp. Matrix concentration inequalities via the method of exchangeable pairs. *The Annals of Probability*, 42(3):906–945, 2014.
- 56 Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem, 2015. [arXiv:1306.3969](#).
- 57 Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families IV: Bipartite ramanujan graphs of all sizes. *SIAM Journal on Computing*, 47(6):2488–2509, 2018.
- 58 Adam W Marcus and Nikhil Srivastava. The solution of the Kadison-Singer problem, 2016. In *Current Developments in Mathematics*. [arXiv:1712.08874](#).
- 59 Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 2009.
- 60 Raghu Meka. Discrepancy and beating the union bound. In *Windows on theory, a research blog*. <https://windowsontheory.org/2014/02/07/discrepancy-and-beating-the-union-bound/>, 2014.
- 61 Raghu Meka, Oanh Nguyen, and Van Vu. Anti-concentration for polynomials of independent random variables. In *Theory Of Computing*. arXiv preprint, 2017. [arXiv:1507.00829](#).
- 62 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.
- 63 Stanislav Minsker. On some extensions of bernstein’s inequality for self-adjoint operators. *Statistics & Probability Letters*, 127:111–119, 2017.
- 64 Tor Myklebust and Levent Tunçel. Interior-point algorithms for convex optimization based on primal-dual metrics. *arXiv preprint*, 2014. [arXiv:1411.2129](#).

- 65 Simone Naldi and Daniel Plaumann. Symbolic computation in hyperbolic programming. *Journal of Algebra and Its Applications*, 17(10):1850192, 2018.
- 66 Assaf Naor, Shrawas Rao, and Oded Regev. Concentration of markov chains with bounded moments. In *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*, volume 56, pages 2270–2280. Institut Henri Poincaré, 2020.
- 67 Ryan O’Donnell, Rocco A Servedio, and Li-Yang Tan. Fooling polytopes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 614–625, 2019.
- 68 Roberto Imbuzeiro Oliveira. Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint*, 2009. [arXiv:0911.0600](#).
- 69 Prasad Raghavendra, Nick Ryder, Nikhil Srivastava, and Benjamin Weitz. Exponential lower bounds on spectrahedral representations of hyperbolicity cones. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2322–2332. SIAM, 2019.
- 70 Victor Reis and Thomas Rothvoss. Vector balancing in lebesgue spaces. *arXiv preprint*, 2020. [arXiv:2007.05634](#).
- 71 James Renegar. Hyperbolic programs, and their derivative relaxations. *Foundations of Computational Mathematics*, 6(1):59–79, 2006.
- 72 James Renegar. “Efficient” subgradient methods for general convex optimization. *SIAM Journal on Optimization*, 26(4):2649–2676, 2016.
- 73 James Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1-2):1–35, 2019.
- 74 James Renegar. Personal communication, 2019.
- 75 James Renegar and Mutiara Sondjaja. A polynomial-time affine-scaling method for semidefinite and hyperbolic programming. *arXiv preprint*, 2014. [arXiv:1410.6734](#).
- 76 Thomas Rothvoss. Constructive discrepancy minimization for convex sets. *SIAM Journal on Computing*, 46(1):224–234, 2017.
- 77 Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.
- 78 Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4), 2007.
- 79 Mark Rudelson and Roman Vershynin. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18, 2013.
- 80 James Saunderson. A spectrahedral representation of the first derivative relaxation of the positive semidefinite cone. *Optimization Letters*, 12(7):1475–1486, 2018.
- 81 James Saunderson. Certifying polynomial nonnegativity via hyperbolic optimization. *SIAM Journal on Applied Algebra and Geometry*, 3(4):661–690, 2019.
- 82 Zhao Song and Ruizhe Zhang. Hyperbolic concentration, anti-concentration, and discrepancy, 2020. [arXiv:2008.09593](#).
- 83 Joel Spencer. Six standard deviations suffice. *Transactions of the American mathematical society*, 289(2):679–706, 1985.
- 84 Colin J Thompson. Inequality with applications in statistical mechanics. *Journal of Mathematical Physics*, 6(11):1812–1813, 1965.
- 85 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.
- 86 Joel A Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.
- 87 Joel A Tropp. Second-order matrix concentration inequalities. *Applied and Computational Harmonic Analysis*, 44(3):700–736, 2018.
- 88 Roman Vershynin. Concentration inequalities for random tensors. *Bernoulli*, 26(4):3139–3162, 2020.
- 89 Ruizhe Zhang and Xinzhi Zhang. A real stable generalization of Anari, Oveis Gharan and Kyng, Luh, Song. *manuscript*, 2021.



## A Basics of Hyperbolic Polynomial

### A.1 Basic definitions of hyperbolic polynomials

We provide the definition of hyperbolic polynomial.

► **Definition 11** (Hyperbolic polynomial). *A homogeneous polynomial  $h : \mathbb{R} \rightarrow \mathbb{R}$  is hyperbolic with respect to a vector  $e \in \mathbb{R}^m$  if  $h(e) \neq 0$ , and for all  $x \in \mathbb{R}^m$ , the univariate polynomial  $t \mapsto h(te - x)$  has only real zeros.*

The following fact shows how to factorize a hyperbolic polynomial, which easily follows from the homogeneity of the polynomial:

► **Fact 12** (Hyperbolic polynomial factorization). *For a degree- $d$  polynomial  $h \in \mathbb{R}[z_1, \dots, z_m]$  hyperbolic with respect to  $e \in \mathbb{R}^m$ , we have*

$$h(te - x) = h(e) \prod_{i=1}^d (t - \lambda_i(x))$$

where  $\lambda_1(x) \geq \lambda_2(x) \geq \dots \geq \lambda_d(x)$  are real roots of  $h(te - x)$ .

All the vectors with nonnegative hyperbolic eigenvalues form a cone, which is proved by Gårding [28]. It is a very important object related to the geometry of hyperbolic polynomials. The formal definition is as follows:

► **Definition 13** (Hyperbolic cone). *For a degree  $d$  hyperbolic polynomial  $h$  with respect to  $e \in \mathbb{R}^m$ , its hyperbolic cone is*

$$\Lambda_+(e) := \{x : \lambda_d(x) \geq 0\}.$$

The interior of  $\Lambda_+^m$  is

$$\Lambda_{++}(e) := \{x : \lambda_d(x) > 0\}.$$

Gårding [28] showed the following fundamental properties of the hyperbolic cone:

- **Theorem 14** ([28]). *Suppose  $h \in \mathbb{R}[z_1, \dots, z_m]$  is hyperbolic with respect to  $e \in \mathbb{R}^n$ . Then,*
1.  $\Lambda_+(e), \Lambda_{++}(e)$  are convex cones.
  2.  $\Lambda_+ + (e)$  is the connected component of  $\{x \in \mathbb{R}^m : h(x) \neq 0\}$  which contains  $e$ .
  3.  $\lambda_{\min} : \mathbb{R}^m \rightarrow \mathbb{R}$  is a concave function, and  $\lambda_{\max} : \mathbb{R}^m \rightarrow \mathbb{R}$  is convex.
  4. If  $e' \in \Lambda_{++}(e)$ , then  $h$  is also hyperbolic with respect to  $e'$  and  $\Lambda_{++}(e') = \Lambda_{++}(e)$ .

For simplicity, we may use  $\Lambda_+$  and  $\Lambda_{++}$  to denote  $\Lambda_+(e), \Lambda_{++}(e)$ , when  $e$  is clear from context. In this paper, we always assume that  $e$  is any fixed vector in the hyperbolic cone of  $h$ .

We define the trace, rank and spectral norm respect to hyperbolic polynomial  $h$ .

► **Definition 15** (Hyperbolic trace, rank, spectral norm). *Let  $h$  be a degree  $d$  hyperbolic polynomial with respect to  $e \in \mathbb{R}^m$ . For any  $x \in \mathbb{R}^m$ ,*

$$\text{tr}_h[x] := \sum_{i=1}^d \lambda_i(x), \quad \text{rank}(x) := \#\{i : \lambda_i(x) \neq 0\}, \quad \|x\|_h := \max_{i \in [d]} |\lambda_i(x)| = \max\{\lambda_1(x), -\lambda_d(x)\}.$$

We define the  $p$  norm with respect to hyperbolic polynomial  $h$ .

## 10:18 Hyperbolic Concentration, Anti-Concentration, and Discrepancy

► **Definition 16** ( $\|\cdot\|_{h,p}$  norm). For any  $p \geq 1$ , we define the hyperbolic  $p$ -norm  $\|\cdot\|_{h,p}$  defined as:

$$\|x\|_{h,p} := \|\lambda(x)\|_p = \left( \sum_{i=1}^d |\lambda_i(x)|^p \right)^{1/p} \quad \forall x \in \mathbb{R}^m.$$

It has been shown that  $\|\cdot\|_h$  and  $\|\cdot\|_{h,p}$  are indeed norms:

► **Theorem 17** ([28, 18, 73]).  $\|\cdot\|_h$  is a semi-norm.

Furthermore, if  $\Lambda_+$  is regular, i.e.,  $(\Lambda_+ \cap -\Lambda_+) = \{0\}$ , then  $\|\cdot\|_h$  is a norm on  $\mathbb{R}^m$ .

► **Theorem 18** ([14]). For any  $p \geq 1$ ,  $\|\cdot\|_{h,p}$  is a semi-norm. Moreover, if the hyperbolic cone  $\Lambda_+$  is regular, then  $\|\cdot\|_{h,p}$  is a norm.

### A.2 Basic properties of hyperbolic polynomials

We state a fact for the eigenvalues  $\lambda(\cdot)$  of degree- $d$  hyperbolic polynomial  $h$ .

► **Fact 19** ([14]). For all  $i \in [d]$ ,

$$\lambda_i(s \cdot x + t \cdot e) = \begin{cases} s \cdot \lambda_i(x) + t, & \text{if } s \geq 0; \\ s \cdot \lambda_{d-i}(x) + t, & \text{if } s < 0. \end{cases}$$

Then, we show that the elementary symmetric sum-products of eigenvalues can be computed from the directional derivatives of the polynomial.

► **Observation 20** ([14]). For a degree- $d$  hyperbolic polynomial  $h$  with respect to  $e$ , we have

$$h(te + x) = p(e) \cdot \prod_{i=1}^d (t + \lambda_i(x)) = \sum_{i=0}^d s_i(\lambda(x)) \cdot t^{d-i},$$

where  $\lambda(x) = (\lambda_1(x), \dots, \lambda_d(x))$  are the hyperbolic eigenvalues of  $x$  and  $s_i : \mathbb{R}^d \rightarrow \mathbb{R}$  is the  $i$ -th elementary symmetric polynomial:

$$s_i(y) := \begin{cases} \sum_{S \in \binom{[d]}{i}} \prod_{j \in S} y_j, & \forall i \in [d]; \\ 1 & \text{if } i = 0. \end{cases}$$

Furthermore, for each  $i \in \{0, 1, \dots, d\}$ ,

$$h(e) \cdot s_i(\lambda(x)) = \frac{1}{(d-i)!} \cdot \nabla^{d-i} h(x) \underbrace{[e, e, \dots, e]}_{(d-i) \text{ terms}}.$$

If  $i \in [d]$ , then  $s_i \circ \lambda$  is hyperbolic with respect to  $e$  of degree  $i$ .

► **Corollary 21.**  $\text{tr}[x]$  is a linear function.

**Proof.** By Observation 20, we have

$$\text{tr}_h[x] = s_1(\lambda(x)) = \frac{1}{h(e) \cdot (d-1)!} \cdot \nabla^{d-1} h(x) [e, e, \dots, e].$$

Since  $h$  is of degree  $d$ ,  $\nabla^{d-1} h$  is a degree-1 polynomial. Hence,  $\text{tr}_h[x]$  is a linear function. ◀



### A.3 Helton-Vinnikov Theorem

We state a corollary of Helton-Vinnikov Theorem (Theorem 10), proved by Gurvits [33]:

► **Corollary 22** (Proposition 1.2 in [33]). *Let  $h(x)$  be a  $m$ -variable degree- $d$  hyperbolic polynomial. Then, for  $x, y \in \mathbb{R}^m$ , there exists two symmetric real matrices  $A, B \in \mathbb{R}^{d \times d}$  such that for any  $a, b \in \mathbb{R}$ , the ordered eigenvalues  $\lambda(ax + by) = \lambda(aA + bB)$ .*

This Corollary relates the hyperbolic eigenvalues of a vector  $ax + by$  to the eigenvalues of matrix  $aA + bB$ , which allows us to study some properties of hyperbolic eigenvalues using results in matrix theory.



# Improved Local Testing for Multiplicity Codes

Dan Karliner ✉

Department of Computer Science, Tel Aviv University, Israel

Amnon Ta-Shma ✉

Department of Computer Science, Tel Aviv University, Israel

---

## Abstract

Multiplicity codes are a generalization of Reed-Muller codes which include derivatives as well as the values of low degree polynomials, evaluated in every point in  $\mathbb{F}_p^m$ . Similarly to Reed-Muller codes, multiplicity codes have a local nature that allows for local correction and local testing. Recently, [6] showed that the *plane test*, which tests the degree of the codeword on a random plane, is a good local tester for *small enough degrees*. In this work we simplify and extend the analysis of local testing for multiplicity codes, giving a more general and tight analysis. In particular, we show that multiplicity codes  $\text{MRM}_p(m, d, s)$  over prime fields with *arbitrary*  $d$  are locally testable by an appropriate *k-flat test*, which tests the degree of the codeword on a random  $k$ -dimensional affine subspace. The relationship between the degree parameter  $d$  and the required dimension  $k$  is shown to be nearly optimal, and improves on [6] in the case of planes.

Our analysis relies on a generalization of the technique of *canonical monomials* introduced in [5]. Generalizing canonical monomials to the multiplicity case requires substantially different proofs which exploit the algebraic structure of multiplicity codes.

**2012 ACM Subject Classification** Theory of computation → Error-correcting codes

**Keywords and phrases** local testing, multiplicity codes, Reed Muller codes

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.11

**Category** RANDOM

**Related Version** *Previous Version:* <https://eccc.weizmann.ac.il/report/2022/078/>

**Funding** *Dan Karliner:* The research leading to these results was supported by Len Blavatnik and the Blavatnik Family foundation and by the Israel Science Foundation grant number 952/18.

*Amnon Ta-Shma:* The research leading to these results was supported by the Israel Science Foundation grant number 952/18.

## 1 Introduction

The Reed-Muller code  $\text{RM}_p(m, d)$  is the set of evaluation tables of  $m$ -variate degree- $d$  polynomials. That is, a function  $f : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$  is in  $\text{RM}_p(m, d)$  if there exists a polynomial  $P$  of degree at most  $d$  such that  $f(a) = P(a)$  for any  $a \in \mathbb{F}_p^m$ . The RM code is a popular building block in CS constructions, due, to a large extent, to its strong local properties.

We say a code  $C \subset \Sigma^n$  is *locally-testable* if given a word  $w \in \Sigma^n$ , the tester distinguishes between the case  $w \in C$  and the case that  $w$  is  $\epsilon$ -far from  $C$  while reading few characters of  $w$ . More precisely, for a code  $C$  and a word  $w$ , we define  $\delta(w, C)$  to be the relative Hamming distance of  $w$  to the closest codeword in  $C$ , i.e.,  $\delta(w, C) = \min_{z \in C} (\Pr_{i \in [n]}(w_i \neq z_i))$ . Then,

► **Definition 1.** A local tester  $\mathcal{A}$  for  $C \subset \Sigma^n$  is a distribution on subsets of  $[n]$ .

- We say  $\mathcal{A}$  is  $q$ -query if any subset in its support is of size  $\leq q$ .
- We say  $\mathcal{A}$  has soundness function  $s$  if for any  $w \in \Sigma^n$ ,

$$\text{REJ}_{\mathcal{A}}(w) = \Pr_{S \sim \mathcal{A}} (w|_S \notin C|_S) \geq s(\delta(w, C)).$$

A typical soundness function  $s$  is of the form  $s(\delta) = \min(\alpha\delta, c)$  for some constants  $\alpha$  and  $c$ . We say  $\mathcal{A}$  is a good local test for  $C$  if it has a nonzero soundness function independent of  $n$ .



© Dan Karliner and Amnon Ta-Shma;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 11; pp. 11:1–11:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 11:2 Improved Local Testing for Multiplicity Codes

We also work with a weaker notion called *local characterization*. We say  $\mathcal{A}$  is a local characterization for  $C$  if  $\text{REJ}_{\mathcal{A}}(w) = 0$  implies  $w \in C$ .

Local testing Reed-Muller codes has been studied extensively and in several parameter regimes [14, 3, 1, 8, 2, 5, 7]. A natural local tester for  $\text{RM}_p(m, d)$  is the *line test*, where we pick a random line and check if its restriction is consistent with a low-degree polynomial. More generally, the *k-flat test* is uniformly distributed over  $k$ -dimensional affine subspaces of  $\mathbb{F}_p^m$ . We denote the rejection probability of the  $k$ -flat test by  $\text{REJ}_{k,d}$ .

Any polynomial in  $k$  variables is equal everywhere to one whose degree in every variable is at most  $p - 1$ , and therefore of total degree at most  $d_k \stackrel{\text{def}}{=} k(p - 1)$ . Therefore, the  $k$ -flat test is not a local characterization for  $\text{RM}_p(m, d)$  when  $d \geq d_k$ .

Quite surprisingly, it was shown in [8] that for any  $d < k(p - 1)$  the  $k$ -flat test is a local characterization for  $\text{RM}_p(m, d)$ , and that it has soundness independent of  $m$ . That is, whenever the line test is not trivially bad, it is a good local test. More concretely, suppose a word  $w : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$  has distance  $\delta$  from  $\text{RM}_p(m, d)$ . The  $k$ -flat test selects  $p^k$  points in  $\mathbb{F}_p^m$ , and so the probability that a “bad” character is read is  $\leq \delta p^k$ . Therefore, the best soundness one could hope for in the  $k$ -flat test is  $\delta p^k$ . Remarkably, later analysis of the  $k$ -flat [5, 7] test shows it is essentially optimal given the number of queries in a wide range of parameters <sup>1</sup>:

► **Theorem 2** (Soundness of the RM  $k$ -flat test, [7]). *There exists a constant  $c > 0$  (independent of  $p$ ) such that the  $k$ -flat test rejects with probability at least  $p^{-c} \min(p^k \delta, 1)$*

### 1.1 The $k$ -flat test for Multiplicity codes

Multiplicity codes were defined in [13, 12, 4, 11].  $\text{MRM}_p(m, d, s)$  is the set of evaluation tables of  $m$ -variate, degree  $d$  polynomials, where we also record the evaluations of all its derivatives up to order  $s$ . More precisely, we define a “multiplicity table” as a function  $T : \mathbb{F}_p^m \rightarrow \Sigma_{m,s}$ , where  $\Sigma_{m,s} \cong \mathbb{F}_p^{\binom{m+s-1}{s-1}}$  is indexed by  $m$ -tuples of weight less than  $s$ . Given a polynomial  $P \in \mathbb{F}_p[x_1, x_2, \dots, x_m]$  we define its evaluation table  $T^P$  as a multiplicity table satisfying, for any  $x \in \mathbb{F}_p^m$  and any  $m$ -tuple  $\mathbf{I}$  with  $\text{wt}(\mathbf{I}) < s$ ,

$$T^P(x)_{\mathbf{I}} = P^{(\mathbf{I})}(x)$$

where  $P^{(\mathbf{I})}(x)$  denotes the direction- $\mathbf{I}$  Hasse derivative of  $P$  at the point  $x$  (see Section 2). Then, the multiplicity code  $\text{MRM}_p(m, d, s)$  is defined as the set of evaluation tables of polynomials of degree at most  $d$ . Notice that this definition makes sense even for  $d > p$ .

With some care, the  $k$ -flat test may be adapted to multiplicity codes. When restricting  $\text{MRM}_p(m, d, s)$  to a  $k$ -flat we want to reduce the alphabet from  $\Sigma_{m,s}$  to  $\Sigma_{k,s}$ . Given a  $k$ -flat  $Q$  with a chosen basis for its linear part  $\mathbf{h}_1, \dots, \mathbf{h}_k$ , one may define the chain rule map  $\phi : \Sigma_{m,s} \rightarrow \Sigma_{k,s}$  given in [6] (following the  $k = 1$  case from [10]) by:

$$(\phi(z))_{\mathbf{J}} = \sum_{\mathbf{I} \in \mathbb{N}^m} z_{\mathbf{I}} \cdot \sum_{\substack{\mathbf{I}_1 + \dots + \mathbf{I}_k = \mathbf{I} \\ \text{wt}(\mathbf{I}_r) = j_r}} \binom{\mathbf{I}}{\mathbf{I}_1, \dots, \mathbf{I}_k} \prod_{i=1}^k \mathbf{h}_i^{\mathbf{I}_i} \quad (1)$$

For a polynomial  $P$ , this is the map that calculates the derivative in direction  $\mathbf{J}$  of  $P|_Q$  from the directional derivatives of  $P$ . Accordingly, if  $w : \mathbb{F}_p^m \rightarrow \Sigma_{m,s}$  is in  $\text{MRM}_p(m, d, s)$  then  $\phi \circ w|_Q$  is in  $\text{MRM}_p(k, d, s)$ .

<sup>1</sup> We note the above discussion can be generalized to prime power fields where the following is known: if  $\mathbb{F}_q$  is of characteristic  $p$  then [8] show the  $k$ -flat test is a local characterization for  $d < k(q - \frac{q}{p})$  and that this bound is tight. Additionally, in this case the  $k$ -flat test also gives a good local test.

## 1.2 For which degree can the $k$ -flat test be effective?

Given a function  $f : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$ , any function equivalent to it mod

$$\mathcal{I}_m = \langle x_1^p - x_1, x_2^p - x_2, \dots, x_m^p - x_m \rangle$$

takes the same values on all of  $\mathbb{F}_p^m$ , and a polynomial  $P$  has the same evaluation table as  $Q$  if and only if  $P \equiv Q \pmod{\mathcal{I}_m}$ .

It is established in [6] that analogously to the Reed-Muller case, two polynomials  $P, Q$  have the same multiplicity tables if and only if their difference  $P - Q$  is in the ideal

$$\mathcal{I}_m^s = \left\langle \prod_{k=1}^s (x_{i_k}^p - x_{i_k}) \mid (i_1, \dots, i_s) \in [m]^s \right\rangle$$

This fact establishes a degree bound on any multiplicity table given  $m, s$ . If a monomial  $\prod x_i^{e_i}$  has  $\sum \left\lfloor \frac{e_i}{p} \right\rfloor \geq s$  then we may subtract a multiple of one of the generators of  $\mathcal{I}_m^s$  to lower its degree. It follows that any polynomial is equivalent (in the sense of having the same multiplicity table) to one with  $\sum \left\lfloor \frac{e_i}{p} \right\rfloor < s$ , which implies  $d \leq d_{k,s} \stackrel{\text{def}}{=} k(p-1) + (s-1)p$ .

## 1.3 Previous work: The plane test is effective for degree $d < ps$

The previous discussion means that the  $k$ -flat test does not characterize  $\text{MRM}_p(m, d, s)$  for  $d \geq d_{k,s}$ . As  $d_{k,s}$  is larger than  $d_k$  - and significantly so for large  $s$  - one may hope that the  $k$ -flat test is a local test for larger  $d$  in the case of multiplicity codes than for Reed-Muller codes. For example, one could hope that the line test is useful even for degrees up to  $sp$ . However, a simple example in [6] shows the line test fails for  $s = 2$  even for  $d = p + 1$ .

Local testing for multiplicity codes is studied in [6], with an emphasis on the 2-flat (“plane”) test. Two main results are obtained: one for characterization and one for robustness. For characterization, [6] show that the *plane* test is a local characterization in degrees nearly reaching  $d_{k,s}$ . Concretely,

► **Theorem 3** (The plane test is a local characterization). *Let  $\mathbb{F}_q$  be a field of size  $q$  of characteristic  $p$  and assume  $s \leq \min\{d, q-1\}$ . Let  $d < q(s - \frac{1}{p})$ . Then the plane test is a local characterization for  $\text{MRM}_p(m, d, s)$ .*

In this paper we focus on the prime field case, in which case the condition becomes  $d < ps - 1$ . The bound  $d < ps - 1$  should be compared to  $d_{2,s} = 2(p-1) + (s-1)p = ps + p - 2$ . While not tight, this result comes close to the trivial limit  $d_{2,s}$ .

The second result in [6] concerns robustness. It shows that if the  $k$ -flat test is a good local test for  $\text{RM}_p(m, d)$  then it is also a local characterization and local test for  $\text{MRM}_p(m, d, s)$ , albeit with worse soundness. This is intuitive because multiplicity tables contain function evaluations, and the derivatives only add more information, and what is left to be shown is that when we pass the test the derivatives are also consistent with the function evaluations.

► **Theorem 4** (Local testing is preserved from RM to MRM, [6]). *Let  $\mathbb{F}_p$  be a field of size  $q$  of characteristic  $p$ , and assume  $s \leq \min\{d, q-1\}$ . Suppose for  $\text{RM}(q, m, d)$  there exists  $\alpha > 0$  and  $c_0 \leq 1$  such that for every  $f$  the rejection probability of the  $k$ -flat test satisfies*

$$\text{REJ}_{k,d}^{\text{RM}}(f) \geq \min\{\alpha \cdot \delta(f, \text{RM}(q, m, d)), c_0\}.$$

Then, for every  $T$  we have

$$\text{REJ}_{k,d}^{\text{MRM}}(T) \geq \min\{\alpha' \cdot \delta(T, \text{MRM}(q, m, d, s)), c_0\}$$

for

$$\alpha' = \alpha \frac{q - (s-1)}{q} \frac{1}{\alpha + q^{d/(p-1)}}$$

Combining Theorems 3 and 4 one gets that under the same conditions as in Theorem 3, the *plane* test is a good local test.

## 1.4 Our new results

The main result of this paper is a new analysis of the plane test, which is based on the canonical monomials of [5], and that we explain in detail in Section 1.5.1. This new analysis is simpler, applies to general  $k$ -flat test ( $k \geq 2$ ) rather than just the plane test, and, more importantly, is tighter. Concretely, we prove:

► **Theorem 5.** *Let  $p$  be prime,  $m \geq 1$ ,  $k \geq 2$  and  $s < p$ . Then the  $k$ -flat test is a local characterization for  $\text{MRM}_p(m, d, s)$  for any  $d < d_{k,s} - (s - 1)$ .*

Thus, the theorem generalize the plane test result of [6] to general  $k$ . Moreover, let us compare the  $k = 2$  case, we see that the trivial argument shows the  $k$ -flat test must fail for  $d \geq d_{2,s} = 2(p - 1) + (s - 1)p = (s + 1)p - 2$ , [6] show the test is a local characterization for  $d \leq ps - 2$ , and, our results show the test is a local characterization for  $d \leq d_{2,s} - s = (s + 1)p - s - 2$ .

We remark, that as before, under the same conditions the  $k$ -flat test is also a good local test. The technique used in [6] does not give good enough soundness in the general case, so we use a different technique based on the soundness analysis in [5]

► **Theorem 6.** *There exist constants  $c_1, c_2$  such that for any prime  $p$ , integers  $m \geq 1$ ,  $k \geq 2$ ,  $s < p$  and  $d < d_{k,s} - (s - 1)$  the  $k$ -flat test is a local tester with soundness function  $\min(\delta p^{-4s-c_1}, p^{-4s-c_2})$ .*

Result-wise our works raises several intriguing questions:

- The question of what is the true degree threshold is intriguing and we suspect that the true answer is indeed the bound  $d_{k,s} - (s - 1)$  that we obtained, i.e., that there is an example of a polynomial of degree  $d_{k,s} - (s - 1) + 1$  where the  $k$ -flat test fails to be a local characterization. In Appendix C we give an example showing tightness for the case  $k = 2, s = 2$  in as well as an example that shows that the degree bound cannot be improved within our technique.
- Another intriguing question is the appearance of the condition  $s < p$  in our results (and also in [6]). Is there an inherent obstacle that appears when we try to take the (Hasse) multiplicity above the field size?
- The state of the art RM results give nearly-optimal soundness for the  $k$ -flat test as long as it is a local characterization. Can this be done for multiplicity codes as well? For instance, is it possible to show soundness on the order of  $\approx p^k \delta$  for small  $\delta$ ?
- This work deals with prime fields, while previous works [5, 6] handle general finite fields for Reed Muller codes and multiplicity codes respectively. Can the improvements in this work be applied to the general finite field case?

We now explain the canonical monomial method of [5] and its use for multiplicity codes.

## 1.5 The technique

We continue the discussion in Section 1.2. Multiplicity tables of multiplicity  $s$  are equivalent to elements of

$$R_{m,s} \stackrel{\text{def}}{=} \mathbb{F}_p[x_1, x_2, \dots, x_m] \mod \mathcal{I}_m^s \quad (2)$$

That is, any multiplicity table has a unique representative in  $R_{m,s}$ , and any two polynomials have the same evaluation table if and only if their difference is in  $\mathcal{I}_m^s$ . We choose

$$\mathcal{B}_{m,s} = \left\{ \prod_{i=1}^m x_i^{e_i} : \sum_{i=1}^m \left\lfloor \frac{e_i}{p} \right\rfloor < s \right\} \quad (3)$$

as a basis for  $R_{m,s}$  (this basis is different than the one chosen in [6]). A table is in  $\text{MRM}_p(m, d, s)$  if and only if its representative polynomial in  $R_{m,s}$  when written in the basis  $\mathcal{B}_{m,s}$  has no monomials of degree larger than  $d$ .

We may view the  $k$  flat test for multiplicity codes algebraically. Given a linear map  $L : \mathbb{F}_p^k \rightarrow \mathbb{F}_p^m$ , any polynomial  $P \in \mathcal{I}_m^s$  has  $P \circ L \in \mathcal{I}_k^s$ . Therefore,  $L$  reduces to a map  $\bar{L} : R_{m,s} \rightarrow R_{k,s}$ . Phrased this way, the  $k$ -flat test takes a polynomial  $P \in R_{m,s}$ , applies a random full-rank affine map  $\bar{L} : R_{m,s} \rightarrow R_{k,s}$  and asks whether  $\bar{L}(P)$  is of degree larger than  $d$  (when written using  $\mathcal{B}_{k,s}$ ). This view of the  $k$ -flat test will be crucial for the soundness analysis appearing in Section 5.

### 1.5.1 Canonical monomials for Reed-Muller codes

An important observation is that both the code  $\text{RM}_p(m, d)$  and the  $k$ -flat test are affine invariant. In fact, many of the results regarding Reed-Muller codes generalize to general affine-invariant codes, see e.g. [9].

In [5], this fact is used to analyze the soundness of the  $k$ -flat test. The idea is, given a polynomial  $P$ , to first find an affine transformation  $L$  that puts  $P$  into a form convenient for analyzing, and then prove the soundness for the polynomial  $P \circ L$ .

To this end they introduce the notion of a *canonical monomial*.

► **Definition 7** ([5, Definition 4.1]). *A canonical monomial of degree  $d$  in  $n \leq m$  variables in  $\mathbb{F}_p[x_1, \dots, x_m]$  is a monomial  $\prod_{i=1}^n x_i^{e_i}$  such that (1)  $\sum_{i=1}^n e_i = d$  (2) For every  $1 \leq i < n$   $e_i = p - 1$  (3)  $e_n \leq p - 1^2$ .*

Intuitively, this is a monomial which is supported on as few variables as possible.

Further, in [5] it is shown that any polynomial can be composed with a linear map  $L$  so that  $P \circ L$  contains a canonical monomial of degree  $\deg P$ . Given that a polynomial contains a canonical monomial, local characterization and testing proofs become much easier.

A map  $L$  for which  $P \circ L$  contains a canonical monomial is given by the linear transformation maximizing (in the graded lexicographic order) the maximal monomial of  $P \circ L$ . The proof contains two stages:

- First, the result is shown for the special case  $m = 2$ .
- An inductive argument generalizes this to any number of variables.

We recount the  $m = 2$  case here.

► **Lemma 8** ([5, Lemma 4.2]). *Let  $f(x_1, x_2)$  be a degree  $d \leq 2(p-1)$  polynomial in  $\mathbb{F}_p[x_1, x_2]$ . Then there exists  $\alpha \in \mathbb{F}_p$  such that  $f(x_1, x_2 + \alpha x_1)$  contains a canonical monomial of degree  $d$ .*

The proof is given in Appendix A for completeness.

---

<sup>2</sup> The definition for prime power fields is more complicated.

### 1.5.2 Canonical monomials for multiplicity codes

When composed with the correct chain rule map defined in Equation (1), multiplicity codes are also affine invariant. Similarly to [5] we want to establish a canonical monomial result for multiplicity codes. This is made more complicated by the fact that individual degrees may be larger than  $p$ .

Let  $s = 2$ . The polynomial  $D_2 = x_2^p x_1 - x_2 x_1^p$  is the minimal representative of its class in  $R_{2,2}$ . For a linear map  $L : \mathbb{F}_p^2 \rightarrow \mathbb{F}_p^2$  we have  $D_2 \circ L = \det(L) D_2$ . Therefore, despite the fact that the degree of  $x_1$  is not at the maximum possible value, we cannot shift the monomial  $x_1^p y_1$  into  $x_1^{p+1}$ .

Where does the proof of Lemma 8 fail? Looking at the coefficient of  $x_1^{p+1}$  in  $f(x_1, x_2 + zx_1)$ , we see it is equal to  $g(z) \stackrel{\text{def}}{=} z - z^p$ . While this polynomial is nonzero, it still evaluates to 0 everywhere on  $\mathbb{F}_p$ . This is possible because its degree is larger than  $p$ .

Let  $P$  be a reduced polynomial in  $R_{2,2}$  of degree  $d < 2p$ . As in the proof of Lemma 8, the coefficient of  $x_1^d$  in  $P(x_1, x_2 + zx_1)$  is  $c_d(z) = \sum_{r \leq d} \alpha_{d-r} z^r$ . As seen above, this polynomial may be 0 everywhere, in which case we may not be able to achieve the monomial  $x_1^d$ . This happens precisely when  $g(z) = z^p - z \mid c_d$ .

Compromising, we next look at the coefficient of  $x_1^{d-1} x_2$ .

$$c_{d-1}(z) = \sum_{r \leq d-1} \alpha_{d-1-r} \binom{r+1}{1} z^r$$

It is readily observed that  $c_{d-1}$  is in fact the *Hasse derivative* of  $c_d$ . If both  $c_d$  and  $c_{d-1}$  are zero everywhere in  $\mathbb{F}_p$ , it follows that in fact  $g(z)^2 \mid c_d$ . However, this implies that  $P$  has degree at least  $2p$ , a contradiction. Therefore, we see that when  $d < 2p$  either the monomial  $x_1^d$  or  $x_1^{d-1} x_2$  can be achieved.

For larger  $s$ , the polynomial  $D_2^{s-1}$  has leading monomial  $x_1^{q(s-1)} x_2^{s-1}$ , and due to its linear invariance we cannot get a higher degree for  $x_1$ . The argument from the preceding paragraph can be applied, and it shows that (if  $d < ps$ ) one of the monomials  $x_1^d, x_1^{d-1} x_2, \dots, x_1^{d-(s-1)} x_2^{s-1}$  must appear in some composition  $P \circ L$ .

The case  $d \geq ps$  is trickier but still true. In general, we prove

► **Theorem 9.** *Let  $p$  be prime,  $s < p$  and let  $P$  be a reduced polynomial in  $R_{2,s}$  of degree  $d$ . Let  $d_{\max}^x = p(s-1) + (p-1)$  and let  $d_{\text{opt}}^x = \min(d, d_{\max}^x)$ . There exists a linear map  $L : \mathbb{F}_p^2 \rightarrow \mathbb{F}_p^2$  such that  $P \circ L$  contains a monomial  $x_1^e x_2^{d-e}$  with  $e \geq d_{\text{opt}}^x - (s-1)$ .*

We clarify the different degree variables introduced so far:

- The degree  $d_{m,s}$  is the highest total degree a reduced polynomial in  $\mathcal{B}_{m,s}$  can have.
- The degree  $d_{\max}^x$  is the highest degree in  $x_1$  a reduced polynomial in  $\mathcal{B}_{m,s}$  can have.
- The degree  $d_{\text{opt}}^x$  is the highest degree in  $x_1$  we might hope for  $P \circ L$  to have. Indeed, by definition its degree in  $x_1$  will be  $\leq d_{\max}^x$ , and the total degree of  $P \circ L$  is  $d$ , so its degree in  $x_1$  cannot be larger than this.

The proof of this theorem is given in Section 3. The proof, while similar in spirit to Lemma 8 requires analyzing several of the polynomials  $c_k$  together as well as careful use of which monomials exist in  $\mathcal{B}_{m,s}$  and which do not.

We state a simple corollary of Theorem 9

► **Corollary 10.** *Under the same assumptions as Theorem 9, there exists a linear map  $L : \mathbb{F}_p^2 \rightarrow \mathbb{F}_p^2$  such that the maximal monomial of  $P \circ L$ ,  $x_1^a x_2^b$  satisfies  $b \leq p-1$ .*



**Proof.** We take the same linear map as in Theorem 9. Suppose  $d_{\max}^x = d$ . Then  $a \geq d - (s-1)$  and so  $b \leq (s-1) < p-1$ . The other case is  $d_{\max}^x = p(s-1) + (p-1)$  in which case  $a \geq p(s-1)$  and so due to  $x_1^a x_2^b$  being in  $\mathcal{B}_{2,s}$  it must be the case that  $b < p$ . ◀

Like in the Reed-Muller case, Theorem 9 can be extended inductively to a canonical monomial statement about general multivariate polynomials. The reduction is slightly more complicated because the product of an  $\mathcal{I}_{m_1}^s$ -reduced polynomial and an  $\mathcal{I}_{m_2}$ -reduced polynomial is not necessarily  $\mathcal{I}_{m_1+m_2}$  reduced when  $s > 1$ .

Taking a slightly different approach from the Reed-Muller definitions, we define canonical monomials as the highest (in graded lexicographic order) monomial achievable by composing with a linear map, and then display their properties.

► **Definition 11** (Canonical monomial – general  $s$ ). *Let  $m, s$  be integers,  $m \geq 2$ ,  $s \geq 1$ . Let  $P \in \mathbb{F}_p[X_1, \dots, X_m]$  be reduced modulo  $\mathcal{I}_m^s$ . The canonical monomial of  $P$  modulo  $\mathcal{I}_m^s$ , denoted  $\text{Can}(P, m, s)$ , is the largest leading monomial of  $P \circ L \bmod \mathcal{I}_m^s$  in the deg-lex ordering (where  $X_1 > \dots > X_m$ ), where the maximum is taken over all linear transformations  $L : \mathbb{F}_p^m \rightarrow \mathbb{F}_p^m$ .*

► **Theorem 12** (Canonical monomials – general  $s$ ). *Let  $p$  be a prime,  $m \geq 2$  and  $s \leq p-2$ . Let  $P \in \mathbb{F}_p[X_1, \dots, X_m]$  be reduced modulo  $\mathcal{I}_m^s$  and suppose  $\prod_{i=1}^m x_i^{e_i} \in \mathcal{B}_{m,s}$  is the canonical monomial of  $P$  modulo  $\mathcal{I}_m^s$ . Then,*

1.  $\sum_{i=1}^m e_i = \deg(P)$
2.  $e_i \geq e_{i+1}$  for all  $i \in [m-1]$ .
3.  $e_1 \geq \min\{p(s-1) + (p-1), d\} - (s-1)$ .
4. If  $n$  is the last integer such that  $e_n > 0$ , then  $e_i = p-1$  for all  $i \in \{2, \dots, n-1\}$ .

This theorem is proved in Section 4.

### 1.5.3 Canonical monomials imply local testing

Suppose a reduced polynomial  $P$  in  $R_{m,s}$  has degree  $> d$  and distance  $\delta$  from  $\text{MRM}_p(m, d, s)$ . Informally, the approach to proving the robustness of the plane test in [6] is to select a plane by first selecting an intermediate uniform  $2s$ -dimensional subspace  $H$ , and within it a uniform plane  $Q$ . The reason this method has soundness on the order of  $p^{-O(s)}$  is:

- Due to Theorem 4 with probability  $\geq \delta \frac{1}{p}$  the restriction  $P|_H$  has degree  $> d$ .
- Due to Theorem 3 at least one plane  $Q$  in  $H$  has  $\deg P|_Q > d$ .
- The number of planes in  $H$  is  $O(p^{cs})$  for some constant  $c$ , so the overall soundness is  $\geq \delta \Omega(p^{-cs-1})$ .

When trying to generalize this approach to the  $k$ -dimensional test, some issues occur. As the degree bound on  $d$  is  $\approx (p-1)k + (s-1)p$ , the space  $H$  needs to have dimension  $k+2s$ . In this case the first step still works. However, the number of  $k$ -dimensional subspaces in  $\mathbb{F}_p^{k+2s}$  can be on the order of  $p^{O(k+s^2)}$ , and this would affect the soundness.

Instead, we replace the second stage with a stronger statement regarding the soundness of the  $k$ -dimensional test within  $\mathbb{F}_p^{k+2s}$ , analogous to the following lemma in [5].

► **Lemma 13** ([5], Lemma 4.6). *Let  $d < k(p-1)$  and let  $f : \mathbb{F}_p^{k+1} \rightarrow \mathbb{F}_p$  have degree larger than  $d$ . Then the  $k$ -dimensional test rejects  $f$  with probability  $\geq \frac{1}{p}$ .*

For the case of multiplicity codes, we show

► **Lemma 14.** *Let  $d < k(p-1) + (s-1)p - (s-1)$  and let  $f : \mathbb{F}_p^{k+1} \rightarrow \Sigma_{k+1,s}$  have degree larger than  $d$ . Then the  $k$ -dimensional test rejects  $f$  with probability  $\geq \frac{1}{p^2}$ .*

This lemma is then applied repeatedly  $2s$  times, showing total soundness of at least  $p^{-4s}$ .

It follows that the probability that a  $k$ -dimensional subspace  $Q$  within the intermediate subspace  $H$  has  $\deg P|_Q > d$  is at least  $p^{-O(s)}$ , giving Theorem 6.

## 2 Preliminaries

For a comprehensive survey of multiplicity codes, see [10]. We present some properties that we use here for completeness. We denote the polynomial ring  $\mathbb{F}_p[X_1, \dots, X_m]$  by  $F[\mathbf{X}]$ . Given a non-negative tuple  $\mathbf{i} = (i_1, \dots, i_m)$ ,  $\mathbf{X}^{\mathbf{i}}$  denotes the monomial  $\prod_{j=1}^m X_j^{i_j}$ .

► **Definition 15** (Hasse derivative). For  $P(\mathbf{X}) \in \mathbb{F}_p[\mathbf{X}]$  and a non-negative tuple  $\mathbf{i}$ , the direction  $\mathbf{i}$  Hasse derivative of  $P$ , denoted  $P^{(\mathbf{i})}(\mathbf{X})$  is the coefficient of  $\mathbf{Z}^{\mathbf{i}}$  in the polynomial  $P(\mathbf{X} + \mathbf{Z})$ .

► **Proposition 16** (Basic properties of Hasse derivatives). Let  $P(\mathbf{X}), Q(\mathbf{X}) \in \mathbb{F}_p[\mathbf{X}]^m$  and let  $\mathbf{i}, \mathbf{j}$  be vectors of non-negative tuples. Then:

1.  $P^{(\mathbf{i})}(\mathbf{X}) + Q^{(\mathbf{i})}(\mathbf{X}) = (P + Q)^{(\mathbf{i})}(\mathbf{X})$ .
2.  $(P \cdot Q)^{(\mathbf{i})}(\mathbf{X}) = \sum_{0 \leq \vec{e} \leq \mathbf{i}} P^{(\vec{e})}(\mathbf{X}) \cdot Q^{(\mathbf{i} - \vec{e})}(\mathbf{X})$ .
3.  $(P^{(\mathbf{i})})^{(\mathbf{j})}(\mathbf{X}) = \binom{\mathbf{i} + \mathbf{j}}{\mathbf{i}} P^{(\mathbf{i} + \mathbf{j})}(\mathbf{X})$ .

► **Definition 17** (Vanishing multiplicity). We say a polynomial  $P$  has vanishing multiplicity  $s$  at  $x$ , and write  $\text{Mult}(P; x) \geq s$ , if for any  $\mathbf{i}$  with  $\text{wt}(\mathbf{i}) < s$ ,  $P^{(\mathbf{i})}(x) = 0$ . We say  $P$  vanishes with multiplicity exactly  $s$  at  $x$ , if  $\text{Mult}(P; x)$  is at least  $s$  but not  $s + 1$ .

A simple fact derived from Item 2 is:

► **Corollary 18.**  $\text{Mult}(P \cdot Q; x) \geq \text{Mult}(P; x) + \text{Mult}(Q; x)$ .

We also need the following:

► **Lemma 19** (See, e.g. [6]). A polynomial has vanishing multiplicity  $\geq s$  if and only if

$$P \in \mathcal{I}_m^s = \left\langle \prod_{k=1}^s (x_{i_k}^p - x_{i_k}) \mid (i_1, \dots, i_s) \in [m]^s \right\rangle$$

► **Lemma 20.** Let  $P$  be a bivariate homogeneous polynomial, and  $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p \setminus (0, 0)$ . Then,  $\text{Mult}(P; (a, b)) \geq t$  iff  $(bx - ay)^t | P$ .

**Proof.** From Item 2 it is clear that  $(bx - ay)^t | P$  implies  $\text{Mult}(P; (a, b)) \geq t$ . We prove the other direction by induction on  $t$ . Suppose  $\text{Mult}(P; (a, b)) \geq t$ , and let  $d = \deg(P)$ .

For  $t = 1$ , suppose w.l.o.g. that  $b \neq 0$  and define  $p(x) = P(x, 1)$ . Then

$$0 = P(a, b) = P(b \cdot (\frac{a}{b}, 1)) = b^d \cdot p(\frac{a}{b}).$$

Thus  $p(\frac{a}{b}) = 0$  and  $(x - \frac{a}{b}) | p$ , i.e.,  $(bx - a) | p$ . Then, the homogeneous form of  $bx - a$  divides the homogeneous form of  $p$ , i.e.,  $(bx - ay) | P$ .

Now let us assume for  $t \geq 1$  and prove for  $t + 1$ . Suppose  $\text{Mult}(P; (a, b)) \geq t + 1$ . Then, by induction,  $P = (bx - ay)^t \cdot Q$  for some homogeneous polynomial  $Q$ . Let  $\mathbf{i}$  be of weight  $t$ . Then,

$$0 = P^{(\mathbf{i})}(a, b) = Q^{(0,0)}(a, b) \cdot ((bx - ay)^t)^{\mathbf{i}}(a, b),$$

because all  $(bx - ay)^t$  derivatives of weight less than  $t$  vanish. However, for some  $\mathbf{i}$  of weight  $t$  we must have  $((bx - ay)^t)^{\mathbf{i}}(a, b) \neq 0$  (e.g., if  $a \neq 0$ , take  $\mathbf{i} = (t, 0)$ ) and therefore  $Q(a, b) = 0$ . Thus, by the base case,  $bx - ay | Q$ , and therefore  $(bx - ay)^{t+1} | P$  as desired. ◀

## 2.1 The Moore matrix

We pay special attention to the case where  $m = 2$  and  $P$  is homogeneous. The Moore matrix of order 2 is  $\begin{pmatrix} x & x^p \\ y & y^p \end{pmatrix}$  and the Moore determinant of order 2 is

$$D_2(x, y) \stackrel{\text{def}}{=} \det \begin{pmatrix} x & x^p \\ y & y^p \end{pmatrix} = xy^p - yx^p.$$

$D_2 = x(y^p - y) - y(x^p - x)$  is a homogeneous polynomial with vanishing multiplicity 1. As  $D_2$  vanishes on the whole of  $\mathbb{F}_p \times \mathbb{F}_p$ , by Lemma 20 we get the well known fact:

► **Corollary 21.**

$$D_2(x, y) = (-y) \cdot \prod_{a \in \mathbb{F}_p} (x - ay).$$

We show  $D_2$  is essentially the only example of a bivariate homogeneous polynomial vanishing over  $\mathbb{F}_q \times \mathbb{F}_q$ :

► **Lemma 22.** *Let  $s < p$ . Suppose  $P$  is a degree- $d$  homogeneous polynomial that vanishes over  $\mathbb{F}_p \times \mathbb{F}_p$  with multiplicity  $s$ . Then  $P$  is divisible by  $D_2^s$ .*

**Proof.** For every point  $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p \setminus (0, 0)$  such that  $\text{Mult}(P; (a, b)) \geq s$ , we have by Lemma 20 that  $(bx - ay)^t | P$ . Taking the points  $\{(a, 1)\}_{a \in \mathbb{F}_p^*}$  and  $(1, 0)$  we see that  $(-y)^t, (x - ay)^t$  divide  $P$ , for every  $a \in \mathbb{F}_p^*$ . As these polynomials are co-prime we get that their product divides  $P$ . Using Corollary 21 we see that  $D_2^t | P$  as desired. ◀

We also need:

► **Lemma 23.** *Let  $P = \sum_i \alpha_i x^i y^{d-i}$  be a degree- $d$  homogeneous bivariate polynomial. Suppose  $P$  is divisible by  $D_2^r$ . Then each polynomial  $P_c = \sum_{i \equiv c \pmod{p-1}} \alpha_i x^i y^{d-i}$  is individually divisible by  $D_2^r$ .*

**Proof.** Let  $P = D_2^r \cdot Q$ . Write  $Q = \sum_i \beta_i x^i y^{d-i}$ , and define  $Q_c = \sum_{i \equiv c \pmod{p-1}} \beta_i x^i y^{d-i}$ . Notice that all the powers of  $x$  in  $D_2^r = (xy^p - x^p y)^r$  are  $r \pmod{p-1}$ . Therefore,  $P_c = D_2^r \cdot Q_{c-r \pmod{p-1}}$ . ◀

## 2.2 The basis $\mathcal{B}_{m,s}$

We recall the definition

$$\mathcal{B}_{m,s} = \left\{ \prod_{i=1}^m x_i^{e_i} : \sum_{i=1}^m \left\lfloor \frac{e_i}{p} \right\rfloor < s \right\}$$

We set up the notation  $e_i = pe_i^1 + e_i^0$  where  $e_i^0 < p$ . That is,  $e^1 e^0$  is the base  $p$  expansion of  $e$ . Due to working with  $s < p$ , we require only two digits for the exponents. With this notation, the restriction on the set of exponents becomes  $\sum_{i=1}^m e_i^1 < s$ .

The following results on  $\mathcal{B}_{m,2}$  are technical and given in Appendix B.

▷ **Claim 24.** The highest degree in  $x$  a monomial in  $\mathcal{B}_{2,s}$  can have is

$$d_{\max}^x = p(s-1) + (p-1) = ps - 1.$$

## 11:10 Improved Local Testing for Multiplicity Codes

▷ **Claim 25.** Let  $s < p$  and suppose  $d = d_{\max}^x + d_{\text{gap}}$  where  $d_{\text{gap}} \geq 0$  (and notice that  $d_{\text{gap}} \leq p - 1$ ). The monomial  $x^i y^{d-i}$  is in  $\mathcal{B}_{2,s}$  if and only if  $0 \leq i \leq d$  and  $i \bmod p \in \{d_{\text{gap}}, d_{\text{gap}} + 1, \dots, p - 1\}$ .

We now show that a homogeneous polynomial with few monomials is not divisible by a high power of  $D_2$ .

► **Lemma 26.** Let  $P = \sum_i \alpha_i x^i y^{d-i}$  be a non-zero, degree- $d$  homogeneous bivariate polynomial, reduced modulo  $\mathcal{I}_m^s$  for  $s < p$ . Suppose further that the set

$$\{i \bmod p \mid \alpha_i \neq 0\} \subseteq \{t, t + 1, \dots, t + k\},$$

i.e., it is contained in a consecutive sequence of at most  $k + 1$  integers. Then  $P$  is not divisible by  $D_2^{k+1}$ .

**Proof.** Let  $c$  be such that  $P_c = \sum_{i \equiv c \bmod (p-1)} \alpha_i x^i y^{d-i}$  is non-zero. We can write

$$P_c = \sum_{j \in J} \alpha_{c+j(p-1)} x^{c+j(p-1)} y^{d-(c+j(p-1))},$$

for some non-empty  $J \subset \mathbb{N}$ , where for every  $j \in J$ ,  $\alpha_{c+j(p-1)} \neq 0$ .

▷ **Claim 27.**  $J \subseteq \{c - t - k, \dots, c - t\}$ .

**Proof.** As  $P$  is reduced modulo  $\mathcal{I}_m^s$  its degree in  $x$  is at most  $ps - 1$ . Therefore, for  $j \in J$ ,  $c + j(p - 1) < ps$ . Hence,  $j < p \cdot \frac{s}{p-1} \leq p$ . Now notice that  $c + j(p - 1) = c - j \bmod p$ . Thus, the assumption that  $\{i \bmod p \mid \alpha_i \neq 0\}$  is contained in  $\{t, \dots, t + k\}$ , implies that  $J \subseteq \{c - t - k, \dots, c - t\}$ . ◁

Therefore, the number of nonzero monomials in  $P_c$  is at most  $k + 1$  (because different  $j$  lead to different  $i \bmod p$ , as  $j < p$ ) and it can be written as

$$\begin{aligned} P_c &= \sum_{j=c-t-k}^{c-t} \alpha_{c+j(p-1)} x^{c+j(p-1)} y^{d-(c+j(p-1))} \\ &= x^{c+(c-t-k)(p-1)} y^{d-c-(c-t)(p-1)} \sum_{j=0}^k \alpha_{c+(c-t-k+j)(p-1)} x^{j(p-1)} y^{(k-j)(p-1)}. \end{aligned}$$

Suppose  $r$  is the largest integer such that  $D_2^r$  divides  $P$ . By Lemma 23  $D_2^r$  divides  $P_c$ . By Corollary 21,  $\prod_{a \in \mathbb{F}_p^*} (x - ay)$  divides  $D_2$ , and therefore

$$\left( \prod_{a \in \mathbb{F}_p^*} (x - ay) \right)^r \mid \sum_{j=0}^k \alpha_{c+(c-t-k+j)(p-1)} x^{j(p-1)} y^{(k-j)(p-1)}.$$

Thus, a polynomial of degree  $r(p - 1)$  divides a polynomial of degree  $k(p - 1)$ . It follows that  $r \leq k$  as desired. ◀

### 3 The two variable case

We restate the main result proven in this section.

► **Theorem 28.** Let  $p$  be prime,  $2 \leq s < p$ , and let  $P$  be a reduced polynomial in  $\mathcal{R}_{2,s}$  of degree  $d$ . Let  $d_{\text{opt}}^x = \min(d, d_{\max}^x) = \min(d, p(s - 1) + (p - 1))$ . There exists a linear map  $L : \mathbb{F}_p^2 \rightarrow \mathbb{F}_p^2$  such that  $P \circ L \bmod \mathcal{I}_m^s$  contains a monomial  $x^i y^{d-i}$  with  $i \geq d_{\text{opt}}^x - (s - 1)$ .

Recall that  $d_{\text{opt}}^x$  is the highest degree we could hope for  $P \circ L$  to have in  $x$ : its degree in  $x$  cannot be higher than  $d$  and cannot be higher than  $d_{\text{max}}^x$ . The lemma states that while we cannot guarantee reaching  $d_{\text{opt}}^x$ , we can get close to it.

**Proof.** We first note that it suffices to prove the lemma in the case where  $P$  is a degree  $d$  homogeneous polynomial. Indeed, given a general polynomial  $P$  of degree  $d$ , express it as  $P = P_d + P_{\text{rest}}$ , where  $P_d$  is homogeneous degree  $d$ , and  $\deg(P_{\text{rest}}) < d$ . Thus, if we know the result for homogeneous polynomials, then  $P_d \circ L$  contains a monomial as required, and  $P_{\text{rest}} \circ L$  cannot cancel that monomial, because  $\deg(P_{\text{rest}} \circ L) \leq \deg(P_{\text{rest}}) < d$ , and therefore all monomials in  $P_{\text{rest}} \circ L$  have degree smaller than  $d$ .

So assume  $P$  is homogeneous of degree  $d$  and write  $P = \sum_{i=0}^d \alpha_i x^i y^{d-i}$ . Let  $\text{MON}(P)$  be the union over all linear maps  $L : \mathbb{F}_p^2 \rightarrow \mathbb{F}_p^2$  of the monomials of  $\mathcal{B}_{m,s}$  that appear in  $(P \circ L) \bmod \mathcal{I}_{m,s}$ .

▷ **Claim 29.** Suppose  $\ell < p$ . If  $x^{d-\ell} y^\ell$  appears in  $\mathcal{B}_{m,s}$  but not in  $\text{MON}(P)$  then for every  $t_1, t_2$  such that  $t_1 + t_2 = \ell$ ,  $P^{(t_1, t_2)}$  vanishes over  $\mathbb{F}_p \times \mathbb{F}_p$ .

**Proof.** Suppose for any linear map  $L : x \rightarrow a_1 x + a_2 y, y \rightarrow b_1 x + b_2 y$  the coefficient of  $x^{d-\ell} y^\ell$  in  $(P \circ L) \bmod \mathcal{I}_{m,s}$  is 0. We write out the coefficient of  $x^{d-\ell} y^\ell$  explicitly:

$$\begin{aligned} P \circ L(x, y) &= \sum_{i=0}^d \alpha_i (a_1 x + a_2 y)^i (b_1 x + b_2 y)^{d-i} \\ &= \sum_{i=0}^d \alpha_i \sum_{\ell=0}^d x^{d-\ell} y^\ell \sum_{t_1+t_2=\ell} \binom{i}{t_1} a_1^{i-t_1} a_2^{t_1} \cdot \binom{d-i}{t_2} b_1^{d-i-t_2} b_2^{t_2}. \end{aligned}$$

Therefore, the coefficient of  $x^{d-\ell} y^\ell$  in  $P \circ L$  is

$$c_\ell(a_1, a_2, b_1, b_2) = \sum_{i=0}^d \alpha_i \sum_{t_1+t_2=\ell} \binom{i}{t_1} a_1^{i-t_1} a_2^{t_1} \cdot \binom{d-i}{t_2} b_1^{d-i-t_2} b_2^{t_2}$$

We now look at  $(P \circ L) \bmod \mathcal{I}_{m,s}$ . Notice that each monomial  $x^i y^j$  either appears in  $\mathcal{B}_{m,s}$ , in which case it is left untouched, or not, in which case it gets reduced and becomes a strictly lower degree polynomial. By assumption  $x^{d-\ell} y^\ell$  appears in  $\mathcal{B}_{m,s}$  and is reduced modulo  $\mathcal{I}_{m,s}$ . It also has total degree  $d$ , and therefore cannot be mixed with residues from other terms. Thus, the fact that it does not appear in  $\text{MON}(P)$  implies that  $c_\ell(a_1, a_2, b_1, b_2) = 0$  for all  $a_1, a_2, b_1, b_2 \in \mathbb{F}_p$ .

Now fix arbitrary  $a_1, a_2 \in \mathbb{F}_p$  and look at  $C_{\ell, a_1, a_2}(a_2, b_2) = c_\ell(a_1, a_2, b_1, b_2)$ .  $C_{\ell, a_1, a_2}$  is a homogeneous polynomial in  $a_2, b_2$  of degree  $\ell < p$ . Since it is zero on all of  $\mathbb{F}_p \times \mathbb{F}_p$ , by Schwartz-Zippel it must be the zero polynomial. Hence, for all  $(a_1, a_2) \in \mathbb{F}_p \times \mathbb{F}_p$  and all  $t_1, t_2$  such that  $t_1 + t_2 = \ell$ , we have:

$$\sum_{i=0}^d \alpha_i \cdot \binom{i}{t_1} a_1^{i-t_1} \cdot \binom{d-i}{t_2} b_1^{d-i-t_2} = 0.$$

The value on the left is  $P^{(t_1, t_2)}(a_1, a_2)$ , and therefore  $P^{(t_1, t_2)}(a_1, a_2) = 0$  ◁

▷ **Claim 30.** If  $d_{\text{opt}}^x = d$ ,  $P$  contains a monomial  $x^e y^{d-e}$  with  $e \geq d_{\text{opt}}^x - (s-1)$ .

**Proof.** Suppose  $d_{\text{max}}^x = d$ . We want to show there exists a monomial  $x^{d-\ell} y^\ell \in \text{MON}$  with  $0 \leq \ell \leq s-1$ , because then  $d-\ell = d_{\text{max}}^x - \ell \geq d_{\text{max}}^x - (s-1)$  as desired.

## 11:12 Improved Local Testing for Multiplicity Codes

Suppose not. Then, for every  $0 \leq \ell \leq s-1$ ,  $x^{d-\ell}y^\ell$  is not in  $MON$ . Also, notice that for every such  $\ell$ ,  $x^{d-\ell}y^\ell$  is a monomial in  $\mathcal{B}_{m,s}$  (because  $d \leq d_{\max}^x$  and  $\ell < s < q$ ). Thus, by Claim 29, and using  $s-1 < p$ ,  $P^{(t_1, t_2)}$  vanishes over  $\mathbb{F}_p \times \mathbb{F}_p$  for all  $(t_1, t_2)$  such that  $t_1 + t_2 < s$ . In other words,  $\text{Mult}(P, \mathbb{F}_p^2) \geq s$  and  $P \in I_{2,s}$ . Thus, the reduced form of  $P$  in  $R_{2,s}$  is zero. A contradiction to  $P$  being degree  $d$ .  $\triangleleft$

Define  $d_{gap} = d - d_{\text{opt}}^x$ . When  $d_{gap} = 0$ , i.e.,  $d_{\text{opt}}^x = d$ , we proved the theorem (Claim 30). We now assume  $d_{gap} > 0$ . Define  $r = \min\{p-1-d_{gap}, s-1\}$ .

► **Lemma 31.** *If  $d_{gap} \geq 0$  then for every  $(t_1, t_2)$  with  $t_1 + t_2 = d_{gap}$ ,  $P^{(t_1, t_2)}$  is not divisible by  $D_2^{r+1}$ .*

**Proof.** As  $r = \min\{p-1-d_{gap}, s-1\}$  we have two cases:

■ Case 1:  $r = s-1$ .

Let  $\alpha x^i y^j$  be a monomial in  $P$  with a nonzero coefficient ( $i+j=d$ ). Let  $(t_1, t_2)$  be such that  $t_1 + t_2 = d_{gap}$ . The derivative  $P^{(t_1, t_2)}$  contains the term  $\alpha \binom{i}{t_1} \binom{j}{t_2} x^{i-t_1} y^{j-t_2}$ . However, by Claim 25 we know  $i \bmod p \geq d_{gap}$ , and by definition  $d_{gap} = t_1 + t_2 \geq t_1$ , so  $i \bmod p \geq t_1$ . Hence, by Lucas' theorem, the binomial coefficient  $\binom{i}{t_1}$  is nonzero. Similarly,  $\binom{j}{t_2}$  is nonzero. Thus, since  $P$  is nonzero so is  $P^{(t_1, t_2)}$ .  $P^{(t_1, t_2)}$  is still reduced mod  $\mathcal{I}_m^s$  and, homogeneous and nonzero, and so,  $P^{(t_1, t_2)}$  is not divisible by  $D_2^s$ .

■ Case 2:  $r = p-1-d_{gap}$ .

Let  $(t_1, t_2)$  be such that  $t_1 + t_2 = d_{gap}$ . Write  $P^{(t_1, t_2)} = \sum \beta_i x^i y^{d-d_{gap}-i}$  and note that

$$\beta_i = \alpha_{i+t_1} \cdot \binom{i+t_1}{t_1} \cdot \binom{d-(i+t_1)}{t_2}.$$

Applying Claim 25 to  $P$  we see any  $i$  with  $\alpha_i \neq 0$  has

$$i \bmod p \in \{d_{gap}, d_{gap}+1, \dots, p-1\}$$

Therefore, any  $i$  with  $\beta_i \neq 0$  has

$$i \bmod p \in \{d_{gap}-t_1, d_{gap}-t_1+1, \dots, p-1-t_1\}.$$

By Lemma 26 the largest power of  $D_2$  dividing  $P^{(t_1, t_2)}$  is at most  $(p-1)-d_{gap} = r$ . I.e.,  $P^{(t_1, t_2)}$  is not divisible by  $D_2^{r+1}$ .  $\blacktriangleleft$

We are now ready to prove:

► **Lemma 32.** *If  $d_{gap} > 0$  then,  $P$  contains a monomial  $x^i y^{d-i}$  with  $i \geq d_{\text{opt}}^x - (s-1)$ .*

**Proof.** We want to show there exists a monomial  $x^{d-\ell}y^\ell \in MON$  with  $d_{gap} \leq \ell \leq d_{gap} + r$ , because then  $d-\ell \geq (d-d_{gap})-r = d_{\text{opt}}^x - r \geq d_{\text{opt}}^x - (s-1)$  as desired.

Suppose not. Then, for every  $d_{gap} \leq \ell \leq d_{gap} + r$ ,  $x^{d-\ell}y^\ell$  is not in  $MON$ . Also, notice that for every such  $\ell$ ,  $x^{d-\ell}y^\ell$  is a monomial in  $\mathcal{B}_{m,s}$  (because  $d-\ell \leq d-d_{gap} = d_{\max}^x$  and  $\ell \leq d_{gap} + r < p$ ). Thus, by Claim 29, and using  $d_{gap} + r < p$ ,  $P^{(t_1, t_2)}$  vanishes over  $\mathbb{F}_p \times \mathbb{F}_p$  for all  $(t_1, t_2)$  such that  $d_{gap} \leq t_1 + t_2 \leq d_{gap} + r$ .

Let  $t_1, t_2$  be some non-negative integers with  $t_1 + t_2 = d_{gap}$ . Using property 3 in Proposition 16 we conclude that for any non-negative  $s_1, s_2$  with  $s_1 + s_2 \leq r$ ,

$$(P^{(t_1, t_2)})^{(s_1, s_2)} = \binom{t_1 + s_1}{s_1} \binom{t_2 + s_2}{s_2} P^{(t_1 + s_1, t_2 + s_2)},$$

vanishes over  $\mathbb{F}_p \times \mathbb{F}_p$ . Therefore it follows that  $P^{(t_1, t_2)} \in \mathcal{I}_2^{r+1}$ , or, equivalently (using Lemma 22) that  $D_2^{r+1}$  divides  $P^{(t_1, t_2)}$ . But this is a contradiction to Lemma 31.  $\blacktriangleleft$

Thus, no matter if  $d_{gap} = 0$  or  $d_{gap} > 0$ , in either case  $P$  contains a monomial  $x^i y^{d-i}$  with  $i \geq d_{opt}^x - (s-1)$ , and the proof is complete.  $\blacktriangleleft$

#### 4 The multivariate case

In this section we prove:

► **Theorem 33** (Canonical monomials – general  $s$ ). *Let  $p$  be a prime,  $m \geq 2$  and  $s < p$ . Let  $P \in \mathbb{F}_p[X_1, \dots, X_m]$  be reduced modulo  $\mathcal{I}_m^s$  and suppose  $\prod_{i=1}^m x_i^{e_i} \in \mathcal{B}_{m,s}$  is the canonical monomial of  $P$  modulo  $\mathcal{I}_m^s$ . Then,*

1.  $\sum_{i=1}^m e_i = \deg(P)$
2.  $e_i \geq e_{i+1}$  for all  $i \in [m-1]$ .
3.  $e_1 \geq \min\{p(s-1) + (p-1), d\} - (s-1)$ .
4. If  $n$  is the last integer such that  $e_n > 0$ , then  $e_i = p-1$  for all  $i \in \{2, \dots, n-1\}$ .

Notice that Theorem 33 gives Definition 7 when  $s = 1$ .

**Proof.** The proof is by reduction to one of the following base cases:

- $m = 1$  and arbitrary  $s$  (vacuous),
- $m = 2$  and arbitrary  $s$  (as follows from Theorem 9)
- $s = 1$  and arbitrary  $m$  (from [5]).

Let  $L$  be the linear map maximizing the leading monomial of  $P \circ L \bmod \mathcal{I}_m^s$  in the deg-lex order. Notice that  $\deg(P \circ L \bmod \mathcal{I}_m^s) = \deg(P)$ , because otherwise the leading monomial of  $P$  is larger than that of  $P \circ L \bmod \mathcal{I}_m^s$  in the deg-lex order. We replace  $P$  by  $P \circ L \bmod \mathcal{I}_m^s$ . Let  $\prod_{i=1}^m x_i^{e_i}$  be the leading monomial of  $P$ . It is immediate that  $e_1 \geq e_2 \geq \dots \geq e_m$ , for otherwise changing variables gives a larger leading monomial in the deg-lex order. Thus, we immediately have properties Items 1 and 2.

Before we start proving properties Items 3 and 4 we prove a general principle:

► **Lemma 34.** *Let  $P \in \mathbb{F}[X_1, \dots, X_m]$  be reduced modulo  $\mathcal{I}_m^s$ . Suppose  $\prod_{i=1}^m x_i^{e_i}$  is the canonical monomial of  $P$  modulo  $\mathcal{I}_m^s$ .*

*Let  $J \subset [m]$  be a set of cardinality  $t$ . For notational clarity, suppose  $J = \{a_1, \dots, a_t\}$  and  $[m] \setminus J = \{b_1, \dots, b_{m-t}\}$ . Express  $P$  as*

$$P(x_1, \dots, x_m) = \sum_{i_1, \dots, i_{m-t}} P_{(i_1, \dots, i_{m-t})}(x_{a_1}, \dots, x_{a_t}) \cdot x_{b_1}^{i_1} \cdot \dots \cdot x_{b_{m-t}}^{i_{m-t}},$$

and denote  $s_{rest} = \sum_{i \notin J} \lfloor \frac{e_i}{p} \rfloor$ . Then

$$\prod_{j \in J} x_j^{e_j} = x_{a_1}^{e_{a_1}} \cdot \dots \cdot x_{a_t}^{e_{a_t}}$$

is the canonical monomial of  $P_{(e_{b_1}, \dots, e_{b_{m-t}})}$  modulo  $\mathcal{I}_t^{s-s_{rest}}$ .

**Proof.** Suppose not. Then there exists a linear transformation  $L' : \mathbb{F}_p^t \rightarrow \mathbb{F}_p^t$  such that

$$P_{(e_{b_1}, \dots, e_{b_{m-t}})} \circ L' \bmod \mathcal{I}_t^{s-s_{rest}}$$

gives a larger monomial in the deg-lex ordering. Define a linear transformation on  $L'' : \mathbb{F}_p^m \rightarrow \mathbb{F}_p^m$  that applies  $L'$  on the variables in location  $a_1, \dots, a_t$  and is identity otherwise. Then we claim that  $P \circ L'' \bmod \mathcal{I}_m^s$  gives a larger monomial than  $\prod_{i=1}^m x_i^{e_i}$ .

## 11:14 Improved Local Testing for Multiplicity Codes

Intuitively, since by our assumption,  $P_{(e_{b_1}, \dots, e_{b_{m-t}})} \circ L' \bmod \mathcal{I}_t^{s-s_{rest}}$  has a monomial  $\prod_{j \in J} x_j^{f_j}$  that is larger than  $\prod_{j \in J} x_j^{e_j}$  in the deg-lex ordering, then also

$$\left( (P_{(e_{b_1}, \dots, e_{b_{m-t}})} \circ L') \bmod \mathcal{I}_t^{s-s_{rest}}(x_{a_1}, \dots, x_{a_t}) \cdot \prod_{i \notin J} x_i^{e_i} \right) \bmod \mathcal{I}_m^s$$

has the monomial  $\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} x_i^{e_i}$  that is larger than  $\prod_i x_i^{e_i}$  in the deg-lex ordering. What remains to be shown is that this is true even without the  $(\bmod \mathcal{I}_t^{s-s_{rest}})$  term in the middle, i.e., that

$$\left( (P_{(e_{b_1}, \dots, e_{b_{m-t}})} \circ L')(x_{a_1}, \dots, x_{a_t}) \cdot \prod_{i \notin J} x_i^{e_i} \right) \bmod \mathcal{I}_m^s$$

has the same monomial  $\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} x_i^{e_i}$  as a coefficient, which is a contradiction to the maximality of  $\prod_i x_i^{e_i}$ .

To prove this we define the polynomial

$$\tilde{P}(x_1, \dots, x_m) = \sum_{i_1, \dots, i_{m-t}} P_{(i_1, \dots, i_{m-t})}(x_{a_1}, \dots, x_{a_t}) \cdot \phi(x_{b_1}, i_1) \cdot \dots \cdot \phi(x_{b_{m-t}}, i_{m-t}),$$

where  $\phi(x, j) = (x^p - x)^{j^1} x^{j-j^1}$  and  $j^1 = \lfloor \frac{j}{p} \rfloor$ . Notice that  $\tilde{P}$  is not homogeneous, and that the maximal degree homogeneous part of  $\tilde{P}$  is exactly  $P$ . Therefore, the maximal degree part of  $\tilde{P} \circ L'' \bmod \mathcal{I}_m^s$  equals the maximal degree part of  $P \circ L'' \bmod \mathcal{I}_m^s$ . Hence, if  $\tilde{P} \circ L'' \bmod \mathcal{I}_m^s$  has a maximal-degree monomial larger than  $\prod_i x_i^{e_i}$  in the deg-lex order, so does  $P \circ L'' \bmod \mathcal{I}_m^s$ . We are therefore allowed to look at  $\tilde{P} \circ L'' \bmod \mathcal{I}_m^s$  instead of  $P \circ L'' \bmod \mathcal{I}_m^s$ . When working with  $\tilde{P} \circ L'' \bmod \mathcal{I}_m^s$ , it is that there it is easy to see the inner modulo is correct. Indeed:

■ We first look at the part contributed by  $i_1 = e_{b_1}, \dots, i_{m-t} = e_{b_{m-t}}$ . We see that:

$$\begin{aligned} & (P_{(e_{b_1}, \dots, e_{b_t}} \circ L')(x_{a_1}, \dots, x_{a_t}) \cdot \prod_{i \notin J} \phi(x_i, e_i)) \bmod \mathcal{I}_m^s \\ &= (P_{(e_{b_1}, \dots, e_{b_t}} \circ L') \bmod \mathcal{I}_m^{s-s_{rest}}(x_{a_1}, \dots, x_{a_t}) \cdot \prod_{i \notin J} \phi(x_i, e_i) \bmod \mathcal{I}_m^s, \end{aligned}$$

because  $\prod_{i \notin J} \phi(x_i, e_i) \in \mathcal{I}_m^{s_{rest}}$ .

- Thus,  $\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} \phi(x_i, e_i)$  appears as a monomial of the above term, because it is reduced modulo  $\mathcal{I}_m^s$  (because  $\sum_{j \in J} \lfloor \frac{f_j}{p} \rfloor + \sum_{j \notin J} \lfloor \frac{e_j}{p} \rfloor \leq (s - s_{rest} - 1) + s_{rest} = s - 1$ ).
- Furthermore, this term is not cancelled by terms contributed by other  $(i_1, \dots, i_{m-t})$ , because the monomials  $\phi(x_{b_1}, e_{b_1}) \cdot \dots \cdot \phi(x_{b_{m-t}}, i_{m-t})$  are independent. Therefore, we conclude that  $\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} \phi(x_i, e_i)$  appears as a monomial of  $(\tilde{P} \circ L'') \bmod \mathcal{I}_m^s$ .

By the above discussion, the maximal-degree homogeneous part of

$$\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} \phi(x_i, e_i)$$

appears as a monomial of  $(P \circ L'') \bmod \mathcal{I}_m^s$ . Thus,

$$\prod_{j \in J} x_j^{f_j} \cdot \prod_{i \notin J} x_i^{e_i}$$

appears as a monomial of  $(P \circ L'') \bmod \mathcal{I}_m^s$ . This is a contradiction to the maximality of  $\prod_i x_i^{e_i}$ , and the proof is complete.  $\blacktriangleleft$



Similarly, we can prove:

► **Lemma 35.** *Let  $P \in F[X_1, \dots, X_m]$  be reduced modulo  $\mathcal{I}_m^s$ . Suppose  $\prod_{i=1}^m x_i^{e_i}$  is the canonical monomial of  $P$  modulo  $\mathcal{I}_m^s$ .*

*Let  $J \subset [m]$  be a set of cardinality  $t$ . For notational clarity, suppose  $J = \{a_1, \dots, a_t\}$  and  $[m] \setminus J = \{b_1, \dots, b_{m-t}\}$ . Express  $P$  as*

$$P(x_1, \dots, x_m) = \sum_{i_1, \dots, i_{m-t}} P_{(i_1, \dots, i_{m-t})}(x_{a_1}, \dots, x_{a_t}) \cdot x_{b_1}^{i_1} \cdot \dots \cdot x_{b_{m-t}}^{i_{m-t}},$$

*and denote  $s' = \sum_{i \in J} \lfloor \frac{e_i}{p} \rfloor$ . Then  $\prod_{j \in J} x_j^{e_j} = x_{a_1}^{e_{a_1}} \cdot \dots \cdot x_{a_t}^{e_{a_t}}$  is the canonical monomial of  $P_{(e_{b_1}, \dots, e_{b_{m-t}})}$  modulo  $\mathcal{I}_t^{s'+1}$ .*

**Proof.** Suppose for some  $L'$ ,  $P_{e_{b_1}, \dots, e_{b_t}} \circ L' \bmod \mathcal{I}_t^{s'+1}$  has a larger monomial in the deg-lex ordering. Since  $s' \leq s - s_{rest} - 1$  so does  $P_{e_{b_1}, \dots, e_{b_t}} \circ L' \bmod \mathcal{I}_t^{s-s_{rest}}$ . The claim then follows from Lemma 34. ◀

With Lemmas 34 and 35 we prove:

▷ **Claim 36.**  $e_2 \leq p - 1$ .

**Proof.** Let  $s_{rest} = \sum_{i \geq 3} \lfloor \frac{e_i}{p} \rfloor \leq s - 1$ . By Lemma 34,  $x_1^{e_1} x_2^{e_2}$  is the canonical monomial of  $P_{(e_3, \dots, e_m)}(x_1, x_2)$  modulo  $\mathcal{I}_2^{s-s_{rest}}$ . However, Corollary 10 shows that for  $m = 2$  (and any  $s' \geq 1$ ) the canonical monomial  $x_1^{i_2} x_2^{i_2}$  has  $i_2 < p$ . Thus  $e_2 < p$ . ◀

Thus, for all  $i \geq 2$  we have  $e_i \leq p - 1$ . Next we prove Item 4:

▷ **Claim 37.** Let  $n$  be the largest integer such that  $e_n > 0$ . Then  $e_2 = e_3 = \dots = e_{n-1} = p - 1$ .

**Proof.** Let  $s' = \sum_{i=2}^m \lfloor \frac{e_i}{p} \rfloor$ . As  $e_i \leq p - 1$  for all  $i \geq 2$ , we have  $s' = 0$ . By Lemma 35,  $x_2^{e_2} \cdot \dots \cdot x_n^{e_n}$  is the canonical monomial of  $P_{(e_1)}(x_2, \dots, x_m)$  modulo  $\mathcal{I}_{m-1}^{s'+1}$ . As  $s' + 1 = 1$ , Definition 7 implies that  $e_2 = e_3 = \dots = e_{n-1} = p - 1$  as desired. ◀

Finally we prove Item 3:

▷ **Claim 38.**  $e_1 \geq \min \{(s-1)p + (p-1), d\} - (s-1)$ .

**Proof.** Let  $s_{rest} = \sum_{i \geq 3} \lfloor \frac{e_i}{p} \rfloor$ . As  $e_i \leq p - 1$  for all  $i \geq 2$ , we have  $s_{rest} = 0$ . By Lemma 34,  $x_1^{e_1} x_2^{e_2}$  is the canonical monomial of  $P_{(e_3, \dots, e_m)}(x_1, x_2)$  modulo  $\mathcal{I}_2^{s-s_{rest}}$ , i.e., modulo  $\mathcal{I}_2^s$ . By Theorem 28 we see that

$$e_1 \geq \min \{p(s-1) + p - 1, e_1 + e_2\} - (s-1).$$

- If  $e_3 = 0$  we have  $e_1 + e_2 = d$ . Thus,  $e_1 \geq \min \{p(s-1) + p - 1, d\} - (s-1)$  as desired.
- If  $e_3 > 0$ , then  $e_2 = p - 1$ . If  $e_1 + e_2 \leq p(s-1) + p - 1$ , then  $e_1 \geq e_1 + e_2 - (s-1)$ . Thus,  $e_2 \leq s - 1 < p - 1$ . A contradiction. Thus  $p(s-1) + (p-1) \leq e_1 + e_2$ . But then  $e_1 \geq p(s-1) + (p-1) - (s-1)$  as desired. ◀

## 5 Proof of the main theorem

In this section, we use Theorem 12 to prove our main theorem, Theorem 6.

We start with simple consequence of the definition of canonical monomials for multiplicity codes. The lemma shows that if the canonical monomial has more than two variables, the variable  $x$  already has the largest multiple of  $p$  possible in  $\mathcal{B}_{m,s}$  in its exponent.

► **Lemma 39.** *Suppose  $\prod_{i=1}^m x_i^{e_i}$  is a canonical monomial for  $P$  of degree  $d$ . Then either  $e_1 \geq p(s-1) + (p-1) - (s-1)$  or  $m \leq 2$ .*

**Proof.** By the definition of canonical monomials, we know  $e_1 \geq \min\{p(s-1) + (p-1), d\} - (s-1)$ . If  $m > 2$ , we know  $e_2 = p-1$  and  $e_3 \geq 1$ . Therefore,  $e_1 < d - p < d - (s-1)$ , so it must be the case that  $\min\{p(s-1) + (p-1), d\} = p(s-1) + (p-1)$ . Therefore,  $e_1 \geq p(s-1) + (p-1) - (s-1)$ . ◀

We proceed similarly to [5] and show that reducing the dimension of tests from  $k+1$  to  $k$  does not hurt soundness too much. This is Lemma 13 in the RM case. It should be noted that this lemma is the central tool in the soundness analysis in [7].

We now show an analogous lemma for multiplicity codes

► **Lemma 40.** *Let  $d < k(p-1) + (s-1)p - (s-1)$  and let  $f : \mathbb{F}_p^{k+1} \rightarrow \Sigma_{k+1,s}$  have degree larger than  $d$ . Then the  $k$ -dimensional test rejects  $f$  with probability at least  $\frac{1}{p^2}$ .*

Again, we assume WLOG that  $f$  contains a canonical monomial  $\prod_{i=1}^m x_i^{e_i}$ ,  $m \leq k+1$  and we consider the restriction to a dimension  $k$  space as modding out by a single linear equation  $L$ . The general strategy of the proof is to show that if the  $x_1$  coefficient of the linear equation  $L$  is zero, everything behaves like the Reed-Muller case. As the  $x_1$  coefficient is zero with probability  $\frac{1}{p}$ , the overall rejection probability will be at least  $\frac{1}{p} \cdot \frac{1}{p} = \frac{1}{p^2}$ .

Essentially, because the power of  $x_1$  is  $\geq (s-1)p$  and because we are focused on monomials with the highest  $x_1$ -degree, we can do the same calculation as in the Reed-Muller case.

**Proof.** Write  $L = L_1x_1 + L_2x_2 + \dots + L_{k+1}x_{k+1} + c$ . We first handle the case  $m = 2$ . In this case, any  $L$  with  $L_1 = 0, L_2 = 0$  will retain the monomial  $x_1^{e_1}x_2^{e_2}$ , which has degree  $\deg f$ . Therefore, the probability that  $\deg(f|_{L=0}) \geq \deg f > d$  is at least  $\frac{1}{p^2}$ .

Otherwise, write  $f = \sum_{i=0}^{e_1} x_1^i f_i(x_2, \dots, x_{k+1})$ . By Lemma 39 we may assume  $e_1 \geq (s-1)p + (p-1) - (s-1)$ . Due to  $e_1 \geq (s-1)p$ , all degrees in  $f_{e_1}$  are  $< p$ , because otherwise  $e_1 f_{e_1}$  would be  $\mathcal{I}_{k+1}$ -reducible. Additionally,  $\deg(f_{e_1}) = \deg(f) - e_1 > d - e_1$ , and  $d - e_1 < (k-1)(p-1)$ . Hence  $f_{e_1}$  satisfies the conditions of Lemma 13 for with  $d = d - e_1$  and  $k = k - 1$ . Therefore, conditioned on  $L_1 = 0$ , we know that with probability at least  $\frac{1}{p}$

there exists a polynomial  $g$  with  $\deg(g) > d - e_1$  and  $h \stackrel{\text{def}}{=} (g - f_{e_1}|_{L=0}) \in \mathcal{I}_k$ .

In this case,  $h(x_1^p - x_1)^{s-1} \in \mathcal{I}_{k+1}^s$ , therefore so is  $hx_1^{e-s(p-1)}(x_1^p - x_1)^{s-1}$ . Subtracting this from  $(x_1^{e_1} f_{e_1}|_{L=0}) = x_1^{e_1} (f_{e_1}|_{L=0})$  we see that  $x_1^{e_1} f_{e_1}|_{L=0}$  is  $\mathcal{I}_{k+1}^s$ -equivalent to  $x_1^{e_1} g$  plus terms with a lower power of  $x_1$ . This means  $\deg f \geq \deg(x_1^{e_1} f_{e_1}|_{L=0}) = e_1 + \deg g > d$ . ◀

► **Corollary 41.** *Let  $d < k(p-1) + (s-1)p - (s-1)$  and let  $f : \mathbb{F}_p^{k+t} \rightarrow \Sigma_{k+t,s}$  have degree larger than  $d$ . Then the  $k$ -dimensional test rejects  $f$  with probability at least  $\frac{1}{p^{2t}}$ .*

**Proof.** This corollary is simply  $t$  repeated applications of Lemma 40, when noting that the distribution on  $k$ -dimensional affine subspaces in  $\mathbb{F}_p^{k+t}$  given by selecting a  $k+t-1$  dimensional subspace uniformly, and within it a  $k+t-2$  dimensional subspace is uniform over all  $k+t-2$  dimension subspaces. ◀

We now prove Theorem 6.

► **Theorem 42.** *There exist constants  $c_1, c_2$  such that for any prime  $p$ , integers  $m \geq 1$ ,  $k \geq 2$ ,  $s < p$  and  $d < d_{k,s} - (s - 1)$  the  $k$ -flat test is a local tester with soundness function  $\min(\delta p^{-4s-c_1}, p^{-4s-c_2})$ .*

**Proof.** Let  $f : \mathbb{F}_p^m \rightarrow \Sigma_{m,s}$ , and let  $\delta = \delta(f, \text{MRM}_p(m, d, s))$ .

We will choose our  $k$ -dimensional subspace by choosing a  $k + 2s$ -dimensional subspace  $H_1$  and within it a  $k$ -dimensional subspace  $H_2$ .  $H_2$  is uniformly distributed.

By Theorem 4 together with Theorem 2 we know that there exists a universal constant  $c$  such that  $\mathbb{P}_{H_1}(\deg f|_{H_1} > d) \geq \min\{\alpha, p^{-c}\}$  with

$$\alpha = p^{k+2s-c} \frac{p - (s - 1)}{p} \frac{1}{p^{k+2s-c} + p^{d/(p-1)}} = \frac{1}{p^{O(1)}}$$

By Corollary 41 we know

$$\mathbb{P}_{H_2}(\deg f|_{H_2} > d) \geq p^{-4s} \mathbb{P}_{H_1}(\deg f|_{H_1} > d) \geq p^{-4s} \min\{\delta p^{-c_1}, p^{-c_2}\},$$

and the proof is complete. ◀

## References

- 1 Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.
- 2 Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of reed-muller codes. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 488–497. IEEE, 2010.
- 3 Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pages 190–198. IEEE, 1995.
- 4 Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.
- 5 Elad Haramaty, Amir Shpilka, and Madhu Sudan. Optimal testing of multivariate polynomials over small prime fields. *SIAM Journal on Computing*, 42(2):536–562, 2013.
- 6 Dan Karliner, Roie Salama, and Amnon Ta-Shma. The plane test is a local tester for multiplicity codes. *preprint*, 2022. URL: <https://eccc.weizmann.ac.il/report/2022/028/>.
- 7 Tali Kaufman and Dor Minzer. Improved optimal testing results from global hypercontractivity, 2022. [arXiv:2202.08688](https://arxiv.org/abs/2202.08688).
- 8 Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing*, 36(3):779–802, 2006.
- 9 Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 403–412, 2008.
- 10 Swastik Kopparty. Some remarks on multiplicity codes. *Discrete Geometry and Algebraic Combinatorics*, 625:155–176, 2013.
- 11 Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM (JACM)*, 61(5):1–20, 2014.
- 12 Rasmus Refslund Nielsen. *List decoding of linear block codes*. PhD thesis, DTU, 2001.
- 13 M Yu Rosenbloom and Michael Anatol’evich Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.
- 14 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

## A

 Proof of bivariate Reed Muller canonincal monomials

The following proof is taken from [5].

**Proof.** Write  $f(x_1, x_2) = \sum_{e: 0 \leq e, d-e < p} \alpha_e x_1^e x_2^{d-e}$ . Monomials of degree lower than  $d$  may be ignored because they will never affect the degree- $d$  homogeneous part of  $f \circ L$ . Let  $e_{\max}$  be the maximal degree of  $x_1$  in  $f$ . If  $e_{\max} = p - 1$  we are done. Otherwise, consider the polynomial  $f(x_1, x_2 + zx_1)$ . By the binomial theorem it follows that

$$f(x_1, x_2 + zx_1) \equiv_{\mathcal{I}_2} \sum_{e \leq d} \alpha_e x_1^e \sum_{r \leq d-e} \binom{d-e}{r} (zx_1)^r x_2^{d-e-r}$$

Look at the coefficient of  $x_1^{e_{\max}+1} x_2^{d-(e_{\max}+1)}$  as a polynomial in  $z$ . It is equal to

$$\sum_{r \leq e_{\max}+1} \alpha_{e_{\max}+1-r} \binom{d-(e_{\max}+1-r)}{r} z^r$$

This is a polynomial of degree at most  $p - 1$ . It is not the zero polynomial because the coefficient of  $z$  is  $\alpha_{e_{\max}} \binom{d-e_{\max}}{1} \not\equiv 0 \pmod{p}$ . Therefore, it is nonzero when  $z = \alpha$  for some  $\alpha \in \mathbb{F}_p$ .

In this way we may increase the maximal degree of  $x_1$  until we obtain a maximal monomial. ◀

## B

 Monomials in  $\mathcal{B}_{m,2}$ 

In this appendix, we establish bounds on the degrees of monomials in  $\mathcal{B}_{m,2}$ .

- For  $d < ps$ , any monomial of degree  $d$  is contained in  $\mathcal{B}_{m,s}$ . Indeed, if  $\sum_{i=1}^m e_i^1 \geq s$  then  $d = \sum e_i \geq ps$ .
- On the other hand, the highest possible degree is

$$d_{m,s} = p(s-1) + (p-1)m.$$

Indeed,  $p \sum_{i=1}^m e_i^1$  is bounded by  $p(s-1)$  and  $\sum_{i=1}^m e_i^0$  is bounded by  $(p-1)m$ . We now check which monomials of degree  $ps \leq d \leq d_{m,s}$  appear in  $\mathcal{B}_{m,s}$ .

In the case  $m = 2$  this gives

▷ **Claim 43.** The highest degree in  $x$  a monomial in  $\mathcal{B}_{2,s}$  can have is

$$d_{\max}^x = p(s-1) + (p-1) = ps - 1.$$

In general,

▷ **Claim 44.** Let  $s < p$  and suppose  $d = d_{\max}^x + d_{\text{gap}}$  where  $d_{\text{gap}} \geq 0$  (and notice that  $d_{\text{gap}} \leq p-1$ ). The monomial  $x^i y^{d-i}$  is in  $\mathcal{B}_{2,s}$  if and only if  $0 \leq i \leq d$  and  $i \bmod p \in \{d_{\text{gap}}, d_{\text{gap}} + 1, \dots, p-1\}$ .

**Proof.** Fix  $x^i y^{d-i}$ . Let us denote  $j = d - i$ . We have:

$$\begin{aligned} i + j &= d = d_{\max}^x + d_{\text{gap}} = ps - 1 + d_{\text{gap}}, \text{ and,} \\ i + j &= p(i^1 + j^1) + (i^0 + j^0) \end{aligned}$$

and hence

$$ps + d_{\text{gap}} - 1 = p(i^1 + j^1) + (i^0 + j^0). \quad (4)$$

Now, if  $x^i y^{d-i}$  is in  $\mathcal{B}_{2,s}$  then, by definition,  $i^1 + j^1 \leq s - 1$ . In fact  $i^1 + j^1 = s - 1$  for otherwise  $i^0 + j^0 \leq 2(p - 1)$  cannot complete  $p(s - 2)$  to  $d \geq ps - 1$ . Thus,

$$i^0 + j^0 = p - 1 + d_{\text{gap}}.$$

As  $j^0 \leq p - 1$  we have  $i^0 \geq d_{\text{gap}}$  as desired.

For the other direction, if  $x^i y^{d-i}$  is not in  $\mathcal{B}_{m,s}$  then  $i^1 + j^1 \geq s$ . Hence,

$$ps + d_{\text{gap}} - 1 = p(i^1 + j^1) + (i^0 + j^0) \geq ps + i_0 + j_0.$$

It follows that  $i_0 \leq i_0 + j_0 \leq d_{\text{gap}} - 1$  as desired.  $\triangleleft$

## C Tightness of results

In this subsection, we demonstrate that some of the degree bounds in the results described above are tight. The first example is derived from the Moore determinant,  $D_2 = x_1^p x_2 - x_2^p x_1$ . More properties of the Moore determinant are given in Section 2.1.

► **Lemma 45.** *The loss of  $(s - 1)$  in Theorem 9 cannot be improved. That is, there exists a polynomial  $P$  in  $R_{2,s}$  for which, for any linear map  $L$ , the degree of  $x_1$  is at most  $d_{\text{opt}}^x - (s - 1)$*

**Proof.** The polynomial  $P = D_2^{s-1}$  is of degree  $(p + 1)(s - 1)$ , so for it  $d_{\text{opt}}^x = \deg(P)$ . This polynomial has leading monomial  $x_1^{p(s-1)} x_2^{s-1}$ . As is shown in Section 2.1,  $D_2 \circ L = \det(L) D_2$ . Therefore, the degree  $p(s - 1)$  is the highest achievable for  $x_1$ .  $\blacktriangleleft$

We now give an example of the tightness of Theorem 5 for a special case. This example was found using linear algebra.

► **Lemma 46.** *The degree bound in is tight for  $k = 2, s = 2$ .*

To demonstrate this, we need to show a polynomial of degree  $d_{2,2} = 2(p - 1) + p(2 - 1) = 3p - 2$  that drops a degree when restricted to any plane. Define

$$P = x_1 x_2^{p-1} x_3^{p-1} (-x_1^{p-1} + x_2^{p-1} + x_3^{p-1})$$

This polynomial is of degree  $3p - 2$ . We claim that it drops a degree when restricted to any plane.

**Proof.** When restricting to a plane with variables  $y_1, y_2$ , there are only two relevant monomials of degree  $3p - 2$  in  $\mathcal{B}_{2,2}$ :  $y_1^{2p-1} y_2^{p-1}$  and  $y_1^{p-1} y_2^{2p-1}$ . Given a generic linear substitution  $(x_1, x_2, x_3) = (a_1, a_2, a_3)y_1 + (b_1, b_2, b_3)y_2$ , the coefficient of  $y_1^{2p-1} y_2^{p-1}$  may be expressed as a polynomial  $C(a_i, b_i)$  in  $a_1, a_2, a_3, b_1, b_2, b_3$ . The polynomial  $C$  may be confirmed to be divisible by  $-a_2^7 a_3 b_1 + a_2 a_3^7 b_1 + a_1^7 a_3 b_2 - a_1 a_3^7 b_2 - a_1^7 a_2 b_3 + a_1 a_2^7 b_3$ , which is 0 for all values of  $a_i, b_i$ . By symmetry, the coefficient of  $y_1^{p-1} y_2^{2p-1}$  will also always be 0.  $\blacktriangleleft$



# Unbalanced Expanders from Multiplicity Codes

Itay Kalev ✉

Department of Computer Science, Tel Aviv University, Israel

Amnon Ta-Shma ✉ 🏠 

Department of Computer Science, Tel Aviv University, Israel

---

## Abstract

In 2007 Guruswami, Umans and Vadhan gave an explicit construction of a lossless condenser based on Parvaresh-Vardy codes. This lossless condenser is a basic building block in many constructions, and, in particular, is behind the state of the art extractor constructions.

We give an alternative construction that is based on Multiplicity codes. While the bottom-line result is similar to the GUV result, the analysis is very different. In GUV (and Parvaresh-Vardy codes) the polynomial ring is closed to a finite field, and every polynomial is associated with related elements in the finite field. In our construction a polynomial from the polynomial ring is associated with its iterated derivatives. Our analysis boils down to solving a differential equation over a finite field, and uses previous techniques, introduced by Kopparty (in [9]) for the list-decoding setting. We also observe that these (and more general) questions were studied in differential algebra, and we use the terminology and result developed there.

We believe these techniques have the potential of getting better constructions and solving the current bottlenecks in the area.

**2012 ACM Subject Classification** Theory of computation → Pseudorandomness and derandomization

**Keywords and phrases** Condensers, Multiplicity codes, Differential equations

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.12

**Category** RANDOM

**Funding** *Itay Kalev*: Supported by Len Blavatnik and the Blavatnik Family foundation and by the Israel Science Foundation grant number 952/18.

*Amnon Ta-Shma*: Supported by the Israel Science Foundation grant number 952/18.

## 1 Introduction

A condenser is a probabilistic mapping from a large universe  $\{0, 1\}^n$  to a smaller universe  $\{0, 1\}^m$  that preserves the entropy of not too large sets. More formally,  $C : \{0, 1\}^n \times [D] \rightarrow \{0, 1\}^m$  is a  $(k_1, k_2, \epsilon)$  condenser, if for every distribution  $X$  on  $\{0, 1\}^n$  with  $k_1$  min-entropy, the output distribution  $C(X, U_D)$  is  $\epsilon$ -close to having  $k_2$  min-entropy (see Definition 6 for a formal definition).

Ideally, we would like to explicitly build a condenser for any  $n$ ,  $k_1 < n$ , and  $\epsilon = \epsilon(n) > 0$  and have  $D$  as small as possible,  $k_2$  as close as possible to  $k_1 + \log(D)$ , and have  $k_2$  as close as possible to  $m$ . Let us call  $d = \log(D)$  the *seed length* of  $C$ , it measures the amount of randomness the probabilistic construction uses, and clearly the smaller the better. Similarly, let us call  $k_1 + d - k_2$  the *entropy loss* of  $C$ . The entropy loss measures the difference between the amount of entropy in the system ( $k_1 + d$ ) and the amount of entropy we preserve ( $k_2$ ), and we want it small. Finally, let us call  $m - k_2$  the *entropy gap* of  $C$ . The entropy gap measures how dense the output distribution  $C(X, U_D)$  is in its ambient space  $\{0, 1\}^m$ , and the smaller the better. Thus, in this terminology, given  $n$ ,  $k_1$  and  $\epsilon$  we would like to find an explicit construction simultaneously minimizing the seed length, entropy loss and entropy gap of the condenser.



© Itay Kalev and Amnon Ta-Shma;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 12; pp. 12:1–12:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An important special case is when the entropy gap  $m - k_2$  is 0, and then  $C$  is called a  $(k_1, \epsilon)$  extractor. Non-explicitly, there are extractors (and so the entropy gap zero) with entropy loss  $2 \log(\frac{1}{\epsilon}) + O(1)$  and seed length  $\log(n - k_1) + 2 \log(\frac{1}{\epsilon}) + O(1)$ , and each one of these bounds is tight (even individually) [13].

Dodis et al. [3] observe that if we allow some entropy gap (and in particular even if it is only a constant) then non-explicitly the entropy loss dramatically drops to  $O(\log \log(\frac{1}{\epsilon}))$  and the seed length to  $\log(n - k) + 1 \cdot \log(\frac{1}{\epsilon}) + O(1)$ . With larger entropy gaps, the entropy loss continues to drop until it basically turns into zero, and then we get a *lossless* condenser. For the dependence of the entropy loss on the entropy gap see [3] (and also [1]).

The GUV lossless condenser [7] has logarithmic seed length and constant fraction entropy gap. Specifically,

► **Theorem 1** (The GUV condenser, [7, Theorem 1.7]). *For every  $n \in \mathbb{N}, k_{max} \leq n, \epsilon > 0$ , and  $0 < \alpha \leq 1$ , there exists an  $m \leq 2d + (1 + \alpha)k_{max}$  and an explicit function*

$$C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$$

*with  $d = (1 + 1/\alpha) \cdot (\log n + \log k_{max} + \log 1/\epsilon) + O(1)$  such that for all  $k \leq k_{max}$ ,  $C$  is an  $(n, k) \rightarrow_\epsilon (m, k + d)$  (lossless) condenser.*

The GUV condenser has found numerous applications (as can be easily seen by looking at the hundreds of papers that cite it). In particular, GUV present an extractor construction by first applying the GUV lossless condenser, and then an extractor construction specifically designed for high min-entropy sources (see [7, Section 4]). Roughly speaking, this extractor construction inherits its entropy loss from the *entropy gap* of the lossless condenser. As a result, the extractor construction presented in [7] has linear entropy loss.

The problem of constructing explicit extractors with short seed length and small entropy loss is widely open and there has been only modest improvement over the extractor of [7] that has linear entropy loss. Specifically, [4] construct explicit extractors with the slightly sub-linear entropy loss  $\frac{k}{\text{polylog}(k)}$ . Their construction uses improved mergers that are obtained using the polynomial method with multiplicities. In another work, [15] modify the GUV condenser construction and using again the multiplicity method of [4] together with other ideas, give a condenser with small entropy loss and the slightly sub-linear *entropy gap*  $\frac{m}{\text{polylog}(n)}$ . This condenser implies an explicit extractor with a short seed and the same slightly sub-linear entropy loss. Constructing an extractor with a short seed and a better entropy loss is still a major open problem.

In this paper we give another explicit construction of a GUV like lossless condenser. While we do not improve the parameters, our construction uses a different analysis that we believe has the potential to substantially improve current state of the art results. Specifically, we prove:

► **Theorem 2** (Our condenser). *For every  $n \in \mathbb{N}, k_{max} \leq n, \epsilon > 0$ , and  $\frac{16 \log \frac{n}{\epsilon}}{\sqrt{k_{max}}} \leq \alpha \leq 1$ , there is an  $m \leq d + (1 + \alpha)k_{max}$  and an explicit function*

$$C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$$

*with  $d = (1 + 1/\alpha) \cdot (\log n + \log k_{max} + \log 1/\epsilon) + O(1)$  such that for all  $k \leq k_{max}$ ,  $C$  is an  $(n, k) \rightarrow_\epsilon (m, k + d)$  (lossless) condenser.*

In a similar fashion to [7], our condenser follows from a new construction of an unbalanced bipartite expander graph.



► **Theorem 3.** *For every field  $\mathbb{F}_q$ ,  $n, s \in \mathbb{N}$  such that  $15 \leq s + 2 \leq n \leq \text{char}(\mathbb{F}_q)$ , there exists an explicit graph  $\Gamma : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^{s+2}$ , which is a  $(K, A)$  expander for every  $K > 0$  with*

$$A = q - \frac{n(s+2)}{2} \cdot (qK)^{\frac{1}{s+2}}. \quad (1)$$

In [7] there is a similar expression with  $A = q - (n-1)(s+1)(K^{\frac{1}{s+1}} - 1)$ .

While the bound on  $m$  in Theorem 2 is slightly better than the one in Theorem 1, the former has more restrictions on  $\alpha$  than the latter. In any case, those two differences are minor, and as stated before, the main contribution of Theorem 2 is the method used to prove it, which is very different than the one used in [7], as we next explain.

## 1.1 Our construction and the GUV construction

Both our construction and the GUV construction have the following structure. The input that we want to condense is interpreted as a degree  $n-1$  uni-variate polynomial over  $\mathbb{F}_q$ , i.e., as an element  $f$  from  $\mathbb{F}_q^{<n}[X]$ . Given the output length  $s+2 \in \mathbb{N}$  (with  $s+2 < n$ ) both constructions associate  $f$  with  $s+1$  different polynomials  $f_0, \dots, f_s$  where  $f_i \in \mathbb{F}_q^{<n}[X]$ . In GUV the association is done as follows:

1. First, put a field structure on  $\mathbb{F}_q^{<n}[X]$  and fix  $h \in \mathbb{N}$ , that way  $f^{h^i}$  (where multiplication and powering is in the field) can also be interpreted as a degree less than  $n$  polynomial.
2. Define  $f_i = f^{h^i}$ .

For example, one may choose a degree  $n$  irreducible polynomial  $E \in \mathbb{F}_q[X]$  and define the field  $\mathbb{F} = \mathbb{F}_q[X] \bmod E$ . Then, the condenser construction is as follows:

| <u>The condenser <math>C</math></u>                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>Parameters:</u> Fix a field <math>\mathbb{F}_q</math>, <math>n, s \in \mathbb{N}</math>, <math>n, s \geq 1</math>. Identify the elements of <math>\mathbb{F}_q^n</math> with univariate polynomials of degree less than <math>n</math>.</p> <p><u>Construction:</u> Define <math>C : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^{(s+2)}</math> by:</p> $C(f, y) = (y, f_0(y), f_1(y), \dots, f_s(y)) \quad (2)$ |

Our construction has the same structure, but our choice of the associated functions  $f_0, \dots, f_s$  is different. Instead of choosing  $f_0, \dots, f_s$  as in GUV, we choose

$$f_i = f^{(i)},$$

i.e.,  $f^{(i)}$  is the  $i$ 'th iterated derivative of  $f$  in  $\mathbb{F}_q[X]$ .

To see why our construction is natural, let us look at it from a coding theory perspective. We can associate a function  $C : V \times [D] \rightarrow \Sigma$  with a linear code of length  $D$  and alphabet  $\Sigma$ , where for every  $v \in V$  we have the codeword

$$(c(v)_1, \dots, c(v)_D) \in \Sigma^D$$

where  $c(v)_i = C(v, i)$ . Using this translation, the GUV construction exactly corresponds to the PV code [12] and our construction exactly corresponds to Multiplicity codes [10, 8].

PV codes and Multiplicity codes are among the few explicit constructions of ECC with close to optimal list-decoding capacity. In the list-decoding problem our goal is to find a construction such that for every given word  $(w_1, \dots, w_D) \in \Sigma^D$  there are few  $v \in V$  such that  $c(v)$  is close to  $w$ . In the condenser construction problem we wish to solve a problem similar to the *list-recoverability* problem, our input is a large subset  $W \subseteq \Sigma$ , and the output

should be the (hopefully few)  $v \in V$  such that  $c(v)_i \in W$  for every  $i \in [D]$  (or the variant where  $c(v)_i \in W$  for most  $i \in [D]$ ). Indeed, GUV write that the known connection between codes and extractors (pointed out, e.g., in [17]) and the fact that PV codes have list-decoding close to capacity motivated them to explore whether PV codes give condensers with good list-recoverability.

Looking at it from this perspective, in this paper we ask whether Multiplicity codes, which are known to have list-decoding close to capacity, also have good list-recoverability and hence give good condensers. In Theorems 2 and 3 we show that this is indeed the case.

Another code which has close to optimal list-decoding capacity is the Folded Reed-Solomon code defined in [6]. Consequently, the condenser it produces has been analyzed in [7, Section 6], and achieved worse parameters than the PV based condenser. Interestingly, the parameters are also worse than the ones achieved by our Multiplicity condenser, making this the first time, to the best of our knowledge, that a construction based on Multiplicity codes achieves better results than one based on FRS codes.

While our construction and the GUV construction are similar in structure, they are very different in implementation. In GUV the ring of polynomials  $\mathbb{F}_q^{<n}[X]$  is “lifted” to a finite field, and the associated functions  $f_i$  are chosen so that they lie on a curve, specifically, over the extension field  $\mathbb{F}$ , all the functions  $f_i$  are just polynomials in one common variable. The challenge is proving that if  $Q(y, f_0, \dots, f_m)$  is a non-zero polynomial in the polynomial ring, then  $Q$  composed with the curve is a *non-zero*, univariate polynomial over the extension field  $\mathbb{F}$ . In general, proving that a non-zero polynomial composed with a given curve remains non-zero is a non-trivial challenge, and GUV solve it with a specific trick, that works, but gives constant entropy gap.

In contrast, our construction does not lift to an extension field. Instead the associated functions are just the derivatives of the given input. Thus, we completely avoid the question of proving that a non-zero polynomial composed with a curve remains non-zero, and, instead, we are left with a question similar to interpolation from derivatives. This leads to a widely different analysis as we explain next. We hope that further extensions of it might lead to constructions better than the current state of the art.

## 1.2 The proof technique

We give a proof sketch of Theorem 3 (the expanding graph). It is enough to prove that for every  $W \subseteq \mathbb{F}_q^{s+2}$  of size at most  $AK - 1$  we have  $|\text{LIST}(W)| < K$ . Fix a set  $W \subseteq \mathbb{F}_q^{s+2}$  of size  $AK - 1$ . Our goal is to bound the number of degree  $n - 1$  polynomials  $f$  such that  $\Gamma(f) \subseteq W$ .

Our starting point is to find a non-zero, low-degree, multi-variate polynomial  $Q(X, Y_0, \dots, Y_s)$  such that  $Q(w) = 0$  for every  $w \in W$ . This step is identical to the first step in the proof of GUV. The total degree of  $Q$  is  $O(|W|^{1/(s+2)}s)$ . It is a standard observation that for every  $f$  with  $\Gamma(f) \subseteq W$  it must be that

$$Q \circ \overline{df} = Q(x, f(x), f'(x), \dots, f^{(s)}(x))$$

is the zero polynomial, i.e.,  $f$  solves the differential equation  $Q$ . The challenge now is to bound that number of functions  $f$  such that  $\Gamma(f) \subseteq W$ .

To bound the number of degree  $n - 1$  polynomials such that  $\Gamma(f) \subseteq W$  we adapt the list-decoding algorithm of [9] to the list-recovery setting (much the same as GUV adapt the [12] list-decoding algorithm to the list-recovery setting). The main lemma Kopparty uses is that given  $(y, w_0, \dots, w_s) \in \mathbb{F}_q \times \mathbb{F}_q^{s+1}$ , there is usually at most one degree  $n - 1$  polynomial  $f$  such that:

- The first  $s$  derivatives of  $f$  at  $y$  agree with  $w_0, \dots, w_s$ , i.e.,  $f^{(i)}(y) = w_i$ , for  $i = 1, \dots, s$ , and,
- $Q \circ \overline{df}$  is the zero polynomial.

Formally, this is true whenever the *Separant* of the equation,  $\frac{\partial Q}{\partial Y_s}$ , is *non-singular* at  $\mathbf{w}$ , i.e.,

$$\frac{\partial Q}{\partial Y_s}(y, w_0, \dots, w_s) \neq 0.$$

Kopparty proves this lemma using Hensel lifting. We rephrase the proof using differential algebra terminology and intuition from [14]. We believe our proof is simpler, and also more amenable to generalizations. Furthermore, this theory was generalized in [11, 5], where generalized Separants were introduced, and we believe these generalization might be useful for future improvements of the analysis.

Going back to the list-recovery problem, and following the list-decoding algorithm from [9], let us denote by  $W_1$  the set of all  $\mathbf{w} \in W$  such that  $\frac{\partial Q}{\partial Y_s}(\mathbf{w}) \neq 0$ . We see that for every  $f$  such that  $\Gamma(f) \subseteq W$  and  $\Gamma(f) \cap W_1 \neq \emptyset$ , we can recover  $f$  by going over all  $\mathbf{w} \in W_1$ , and for each such  $\mathbf{w}$  output the unique suitable degree  $n - 1$  polynomial, given by the above main lemma.

We are then left with the task of outputting all the degree  $n - 1$  polynomials such that  $\Gamma(f) \subseteq W_0 = W \setminus W_1$ . We notice that each of these polynomials solve the lower degree differential equation  $\frac{\partial Q}{\partial Y_s}(x, f(x), \dots, f^{(s)}) = 0$ . Reiterating the process we get a new list of solution. As each time we get a lower degree differential equation, we can iterate the process at most  $\deg(Q)$  times. Doing the calculation more carefully (as is done in [9]) saves even this loss, and, furthermore, shows expansion by a factor of about  $q - sn^{s+2}\sqrt{|W|}$ . We explain the thin details in Section 4.

## 2 Preliminaries

We use the following notation:

$$(n)_t = n \cdot (n - 1) \cdot \dots \cdot (n - t + 1) = \frac{n!}{(n - t)!},$$

where for  $t = 0$ ,  $(n)_0 = 1$ . Thus,  $(n)_t = t! \binom{n}{t}$ .

Also, for  $\mathbf{J} = (j_1, \dots, j_m)$  and  $\mathbf{I} = (i_1, \dots, i_m)$  we define

$$\begin{aligned} (\mathbf{J})_{\mathbf{I}} &= \prod_{\ell=1}^m (j_{\ell})_{i_{\ell}}, \\ \binom{\mathbf{J}}{\mathbf{I}} &= \prod_{\ell=1}^m \binom{j_{\ell}}{i_{\ell}}, \text{ and,} \\ \mathbf{I}! &= \prod_{\ell=1}^m i_{\ell}!. \end{aligned}$$

Thus,  $(\mathbf{J})_{\mathbf{I}} = \mathbf{I}! \binom{\mathbf{J}}{\mathbf{I}}$ . Finally,  $\mathbf{J} - \mathbf{I} = (j_1 - i_1, \dots, j_m - i_m)$ .

### 2.1 Multi-variate derivatives

Let  $R = \mathbb{F}[X_1, \dots, X_m]$  be the ring of polynomials in  $m$  variables over  $\mathbb{F}$ . For  $\mathbf{I} = (i_1, \dots, i_m)$  with  $i_1, \dots, i_m \in \mathbb{N}$  we define the *partial derivative* in direction  $\mathbf{I}$  as the linear operator on  $R$  defined by  $\frac{\partial X^{\mathbf{J}}}{\partial \mathbf{I}} = (\mathbf{J})_{\mathbf{I}} \cdot \mathbf{X}^{\mathbf{J}-\mathbf{I}}$ . We denote

$$Q^{(\mathbf{I})}(\mathbf{X}) = \frac{\partial Q}{\partial \mathbf{I}}(\mathbf{X}).$$

The order of  $\mathbf{I}$  is  $w(\mathbf{I}) = i_1 + \dots + i_m$ . Notice that for uni-variate polynomials  $Q(X)$ ,  $Q^{(i)}(X)$  coincides with the  $i$ 'th iterated derivative.

Let  $\mathbf{w} = (w_1, \dots, w_m)$  where  $w_i \in \mathbb{N}$ . The  $\mathbf{w}$ -weighted degree of a monomial  $\mathbf{X}^{\mathbf{J}} = X_1^{j_1} \dots X_m^{j_m}$  is  $\sum_{i=1}^m w_i \cdot j_i$ . The  $\mathbf{w}$ -weighted degree of  $Q$ , denoted  $\deg_{\mathbf{w}}(Q)$ , is the largest  $\mathbf{w}$ -weighted degree of a monomial in  $Q$ . We let  $|\mathbf{w}|$  denote  $\sum w_i$ ,  $\Pi(\mathbf{w}) = \prod w_i$ , and  $M_{\mathbf{w},t}$  the number of monomials  $\mathbf{X}^{\mathbf{J}}$  with  $\mathbf{w}$ -weighted degree at most  $t$ . Beged-Dov gave upper and lower bounds on  $M_{\mathbf{w},t}$ :

► **Lemma 4** ([2]).

$$\frac{t^m}{m! \cdot \Pi(\mathbf{w})} \leq M_{\mathbf{w},t} \leq \frac{(t + |\mathbf{w}|)^m}{m! \cdot \Pi(\mathbf{w})}$$

## 2.2 Condensers

In this subsection let  $C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ .

► **Definition 5.** We say  $C$  is a  $(K, A)$  expander if for every  $S \subseteq \{0, 1\}^n$  of cardinality  $K$  the set

$$\Gamma(S) = \bigcup_{s \in S, y \in \{0, 1\}^d} C(s, y)$$

has cardinality at least  $K \cdot A$ .

We next define a condenser:

► **Definition 6.** We say  $C$  is an  $(n, k) \rightarrow_{\epsilon} (m, k')$  condenser if for all distributions  $X$  with min-entropy at least  $k$ , the distribution  $C(X, U_d)$  is  $\epsilon$ -close to a distribution with min-entropy at least  $k'$ . The condenser is explicit if  $C$  can be computed in time  $\text{poly}(n, \frac{1}{\epsilon})$ .

To prove that a function is a condenser or an expander, we use the “list-decoding” approach described in [7]. For  $C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  and  $T \subseteq \{0, 1\}^m$  define:

$$\text{LIST}(T) = \{x : \Gamma(x) \subseteq T\}$$

$$\text{LIST}(T, \epsilon) = \left\{ x : \Pr_y [C(x, y) \in T] \geq \epsilon \right\}$$

► **Lemma 7** ([7, Lemma 3.2]).  $C$  is a  $(K, A)$  expander iff for every set  $T \subseteq \{0, 1\}^m$  of cardinality at most  $AK - 1$ ,  $\text{LIST}(T)$  has cardinality at most  $K - 1$ .

And for condensers:

► **Lemma 8** ([16, Theorem 8.1], [7, Lemma 5.4]). Let  $C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a function.

- If  $C$  is a  $(K, (1 - \epsilon)2^d)$  expander, then  $C$  is a  $(n, k) \rightarrow_{\epsilon} (m, k + d)$  condenser, i.e., it is a lossless condenser with error  $\epsilon$ ,
- If for all  $T \subseteq \{0, 1\}^m$  of size at most  $L$  the set  $\text{LIST}(T, \epsilon)$  has cardinality at most  $H$ , then  $C$  is a  $(n, \log(\frac{H}{\epsilon})) \rightarrow_{2\epsilon} (m, \log(\frac{L}{\epsilon}) - 1)$  condenser.

### 3 The Separant

Let  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$ . When we think of  $Q$  as a differential equation, we look for all (low-degree) polynomials  $f \in \mathbb{F}_q[X]$  such that

$$Q(X, f(X), f^{(1)}(X), \dots, f^{(s)}(X)) = 0 \in \mathbb{F}_q[X].$$

Let us define

$$\overline{df} = (X, f(X), f^{(1)}(X), \dots, f^{(s)}(X), \dots, f^{(n)}(X), \dots)$$

Notice that if  $f \in \mathbb{F}_q^{<n}[X]$ , then  $f^{(i)}(X)$  is identically zero for all  $i \geq n$ . Let us also think of  $Q$  as a polynomial  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s, \dots, Y_n, \dots]$  that depends only on  $X$  and  $Y_0, \dots, Y_s$ . In this notation  $f$  solves the differential equation  $Q$  iff  $Q \circ \overline{df} = 0 \in \mathbb{F}_q[X]$ .

A differential equation  $Q$  can be itself derived. While formally  $Q$  depends on  $X$  and  $Y_0, \dots, Y_n, \dots$ , we think of  $Y_0$  as a function depending on  $X$ ,  $Y_0 = f(X)$  and of  $Y_{i+1}$  as  $\frac{\partial Y_i}{\partial X}$ . This motivates the following definition:

► **Definition 9.** Let  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$ , define the infinite sequence of polynomials  $Q^{(0)}, Q^{(1)}, \dots$  where  $Q^{(k)} \in \mathbb{F}_q[X, Y_0, \dots, Y_{k+s}]$  is defined by:

$$\begin{aligned} Q^{(0)} &= Q \\ Q^{(k+1)} &= \frac{\partial Q^{(k)}}{\partial X} + \sum_{i=0}^{k+s} \frac{\partial Q^{(k)}}{\partial Y_i} \cdot Y_{i+1}. \end{aligned}$$

The motivation behind this definition is apparent given:

► **Lemma 10.** For every  $f \in \mathbb{F}_q[X]$  and  $\ell \geq 0$

$$(Q \circ \overline{df})^{(\ell)} = Q^{(\ell)} \circ \overline{df}.$$

**Proof.** By induction. The case  $\ell = 0$  is immediate. Assume for  $\ell$  and let us prove for  $\ell + 1$ . Using the chain rule:

$$\begin{aligned} (Q \circ \overline{df})^{(\ell+1)} &= ((Q \circ \overline{df})^{(\ell)})' = (Q^{(\ell)} \circ \overline{df})' \\ &= \frac{\partial Q^{(\ell)}}{\partial X} \circ \overline{df} + \sum_{i=0}^{s+\ell} \frac{\partial Q^{(\ell)}}{\partial Y_i} \circ \overline{df} \cdot \frac{\partial f^{(i)}}{\partial X} \\ &= \frac{\partial Q^{(\ell)}}{\partial X} \circ \overline{df} + \sum_{i=0}^{s+\ell} \frac{\partial Q^{(\ell)}}{\partial Y_i} \circ \overline{df} \cdot f^{(i+1)} \\ &= \left( \frac{\partial Q^{(\ell)}}{\partial X} + \sum_{i=0}^{s+\ell} \frac{\partial Q^{(\ell)}}{\partial Y_i} \cdot Y_{i+1} \right) \circ \overline{df} \\ &= Q^{(\ell+1)} \circ \overline{df}, \end{aligned}$$

where the first equality is because we use iterated derivations, the second is induction, the third is the chain rule (and notice that  $Q^{(\ell)}$  depends on  $X, Y_0, \dots, Y_{s+\ell}$ ). ◀

We call  $Q^{(\ell)}$  the  $\ell$ -th derivative of  $Q$ . This operation comes from differential algebra [14]. As its name suggests, this operator has some properties similar to regular derivative

## 12:8 Unbalanced Expanders from Multiplicity Codes

▷ **Claim 11** ([14]).

1. (linearity) For every  $Q, P \in \mathbb{F}_q[X, Y_0, \dots]$ ,  $\lambda, \mu \in \mathbb{F}_q$ ,  $\ell \geq 0$

$$(\lambda Q + \mu P)^{(\ell)} = \lambda Q^{(\ell)} + \mu P^{(\ell)}$$

2. (Leibniz product rule) For every  $Q, P \in \mathbb{F}_q[X, Y_0, \dots]$

$$(P \cdot Q)^{(1)} = P^{(1)} \cdot Q + P \cdot Q^{(1)}$$

3. (repeated derivation) For every  $Q \in \mathbb{F}_q[X, Y_0, \dots]$ ,  $\ell_1, \ell_2 \geq 0$

$$(Q^{(\ell_1)})^{(\ell_2)} = Q^{(\ell_1 + \ell_2)}$$

▷ **Claim 12.** Let  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  and  $\ell \in \mathbb{N}$ .

- $\deg_{(0,1,1,\dots)} Q^{(\ell)} = \deg_{(0,1,1,\dots)} Q$ , and,
- $\deg_{(0^{s+2},1,2,3,\dots)} Q^{(\ell)} \leq \ell$ . I.e., if we give  $X, Y_0, \dots, Y_s$  weight 0, and  $Y_{s+j}$  weight  $j$ , then the  $\ell$ 'th derivative degree is at most  $\ell$ .

**Proof.** For the first item notice that  $\frac{\partial Q^{(\ell)}}{\partial X}$  is either zero or does not change the degree in  $Y_0, \dots$ . Also, the effect of  $\frac{\partial Q}{\partial Y_i} \cdot Y_{i+1}$  is to reduce the degree in  $Y_i$  by one and increase the degree in  $Y_{i+1}$  by one.

For the second item, we prove by induction. The case  $\ell = 0$  is immediate. For the induction step,  $\frac{\partial Q^{(\ell)}}{\partial X}$  and  $\frac{\partial Q^{(\ell)}}{\partial Y_i} \cdot Y_{i+1}$  for  $i < s$ , are either zero or do not change the weighted degree, while  $\frac{\partial Q^{(\ell)}}{\partial Y_i} \cdot Y_{i+1}$  for  $i \geq s$  increase the weighted degree by one. ◁

One consequence of Claim 12 is that  $Y_{s+\ell}$  appears with degree at most 1 in  $Q^{(\ell)}$  and that the coefficient of  $Y_{s+\ell}$  in  $Q^{(\ell)}$  is a function of  $X, Y_0, \dots, Y_s$  alone. Indeed, we next prove the coefficient of  $Y_{s+\ell}$  in  $Q^{(\ell)}$  is  $\frac{\partial Q}{\partial Y_s}$ .

► **Definition 13** (Separant). Let  $Q \in \mathbb{F}[X, Y_0, \dots, Y_s]$ . The Separant of  $Q$ , denoted  $S_Q$ , is

$$S_Q = \frac{\partial Q}{\partial Y_s}.$$

A classical lemma from differential algebra (see [14, Page 30]) states that:

► **Lemma 14.** For every  $\ell \geq 1$ ,

$$Q^{(\ell)} = S_Q \cdot Y_{s+\ell} + R_\ell$$

where  $R_\ell \in \mathbb{F}[X, Y_0, \dots, Y_{s+\ell-1}]$  does not depend on  $Y_{s+\ell}$ .

**Proof.** By induction. For  $\ell = 1$ , the only way to get  $Y_{s+1}$  in  $Q^{(1)}$  is in the term  $\frac{\partial Q}{\partial Y_s} \cdot Y_{s+1}$ . Assume for  $\ell$  and let us prove for  $\ell + 1$ . The only way to get  $Y_{s+\ell+1}$  in  $Q^{(\ell+1)}$  is by taking  $\frac{\partial Q^{(\ell)}}{\partial Y_{s+\ell}}$ . By induction,  $Y_{s+\ell}$  only appears in  $Q^{(\ell)}$  in the linear term  $S_Q \cdot Y_{s+\ell}$ . Thus, the only term involving  $Y_{s+\ell+1}$  in  $Q^{(\ell+1)}$  is  $\frac{\partial(S_Q \cdot Y_{s+\ell})}{\partial Y_{s+\ell}} \cdot Y_{s+\ell+1} = S_Q \cdot Y_{s+\ell+1}$ . ◀

► **Lemma 15.** Fix  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$ ,  $(\alpha, \mathbf{b}) = (\alpha, b_0, \dots, b_s) \in \mathbb{F}_q^{s+2}$  and  $S_Q(\alpha, \mathbf{b}) \neq 0$ . Suppose  $f \in \mathbb{F}_q[X]$  such that:

- $f^{(i)}(\alpha) = b_i$ , for  $i = 0, \dots, s$ , and
- $Q \circ \overline{df} = 0$ .

Then there are unique values  $b_{s+1}, \dots, b_n$  such that  $f^{(i)}(\alpha) = b_i$ .

**Proof.** We prove by induction on  $n$ . The base case  $n = s$  is clear. Assume for  $n$  and let us prove for  $n + 1$ . By assumption we know there are unique values  $b_{s+1}, \dots, b_n$  such that  $b_i = f^{(i)}(\alpha)$  for  $i = s + 1, \dots, n$ . Our goal is to show there is a unique value possible for  $f^{(n+1)}(\alpha)$ .

We will use  $Q^{(n-s+1)}$  and the fact that  $Y_{n+1}$  appears linearly in it with coefficient  $S_Q$ , and that at  $(\alpha, \mathbf{b})$ ,  $S_Q(\alpha, \mathbf{b}) \neq 0$ . First we notice that

$$\begin{aligned} Q^{(n-s+1)}(\alpha, b_0, \dots, b_n, f^{(n+1)}(\alpha)) &= Q^{(n-s+1)}(\alpha, f(\alpha), \dots, f^{(n+1)}(\alpha)) \\ &= Q^{(n-s+1)} \circ \overline{df}(\alpha) \\ &= (Q \circ \overline{df})^{(n-s+1)}(\alpha) = 0, \end{aligned}$$

where the first equality is by induction, the second by definition, the third using Lemma 10, and the last equality because we know  $Q \circ \overline{df}$  is the zero polynomial in  $\mathbb{F}_q[X]$ .

Next we recall that by Lemma 14

$$Q^{(n-s+1)}(X, Y_0, \dots, Y_n) = S_Q(X, Y_0, \dots, Y_s) \cdot Y_{n+1} + R(X, Y_0, \dots, Y_n),$$

and therefore

$$\begin{aligned} 0 &= Q^{(n-s+1)}(\alpha, b_0, \dots, b_n, f^{(n+1)}(\alpha)) \\ &= S_Q(\alpha, \mathbf{b}) \cdot f^{(n+1)}(\alpha) + R(\alpha, b_0, \dots, b_n). \end{aligned}$$

Thus,  $f^{(n+1)}(\alpha) = -\frac{R(\alpha, b_0, \dots, b_n)}{S_Q(\alpha, \mathbf{b})}$  is uniquely determined.  $\blacktriangleleft$

In words, this means the following.  $f$  solves the differential equation if  $Q \circ \overline{df} = 0$ . We can think of the conditions  $f^{(i)}(\alpha) = b_i$ , for  $i = 0, \dots, s$ , as  $s + 1$  initial conditions on the Taylor expansion of  $f$  at  $\alpha$ . In this terminology, Lemma 15 says that if the separant  $S_Q$  is non-zero at the point  $(\alpha, \mathbf{b})$  then there can be at most one solution to the differential equation  $Q$  with degree smaller than the characteristic, satisfying the initial conditions  $(\alpha, \mathbf{b})$ .

## 4 Reconstruction with the Polynomial Method

In this section we present a “de-condensing” procedure that given  $\Gamma : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^{s+2}$  and a set  $W \subseteq \mathbb{F}_q^{s+2}$  outputs  $\text{LIST}(W)$ . Throughout this section we assume that  $n \leq \text{char}(\mathbb{F}_q)$ . The de-condensing algorithm works as follows. Given  $W$  we first find a low-degree polynomial  $Q$  that vanishes over  $W$ , namely,

$\triangleright$  **Claim 16.** There exists a non-zero polynomial  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  with

$$\deg_{(1, n, \dots, n-s)} Q \leq D = \left\lceil n \cdot \left[ |W| \cdot (s+2)! \right]^{\frac{1}{s+2}} \right\rceil$$

that vanishes on  $W$ .

**Proof.** By Lemma 4 the number of monomials in  $\mathbb{F}_q[X, Y_0, \dots, Y_s]$  with  $(1, n, n-1, \dots, n-s)$ -weighted degree at most  $D$  is some value  $F$  such that

$$F \geq \frac{D^{s+2}}{(s+2)! \cdot \prod_{j=0}^s (n-j)} > |W|.$$

To find a polynomial  $Q$  that vanishes on  $W$ , we write a homogeneous linear system over  $\mathbb{F}_q$  where the variables are the coefficients of the above monomials, and for every  $w \in W$  we have a linear equation forcing that the polynomial vanishes on  $w$ . As the number of variables is larger than the number of constraints, there is a non-zero solution.  $\triangleleft$

## 12:10 Unbalanced Expanders from Multiplicity Codes

It then follows that every  $f \in \mathbb{F}_q^{<n}[T]$  with  $\Gamma(f) \subseteq W$  satisfies the differential equation  $Q(x, f(x), \dots, f^{(s)}(x)) = 0$ . Formally,

▷ **Claim 17.** If  $f \in \text{LIST}(W)$ , and  $q > D$ , then

$$R_f(T) = Q(T, f(T), \dots, f^{(s)}(T)) \in \mathbb{F}_q[T]$$

is the zero polynomial.

**Proof.** As  $\deg_{(1,n,\dots,n-s)}(Q) \leq D$  and  $\deg(f^{(i)}) < n - i$ ,  $R_f$  has degree at most  $D$ . Also, for every  $\alpha \in \mathbb{F}_q$ ,

$$R_f(\alpha) = Q(\alpha, f(\alpha), \dots, f^{(s)}(\alpha)) = 0.$$

As  $q > D$  we must have  $R_f = 0$  in  $\mathbb{F}_q[T]$ . ◁

The main challenge is proving the number of low-degree solutions to the differential equation  $Q$  with starting conditions  $W$  is small, and designing an algorithm finding all such solutions. For that we define algorithm *Solve*. The input to the algorithm is a polynomial  $\dot{Q} \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  and  $\dot{W} \subseteq \mathbb{F}_q^{s+2}$ . The output contains all polynomials  $f \in \mathbb{F}_q^{<n}[X]$  such that  $\Gamma(f) \subseteq \dot{W}$  and  $\dot{Q} \circ \overline{df} = 0$ . The algorithm works as follows:

■ **Algorithm 1** *Solve*( $\dot{Q}, \dot{W}$ ).

- 
- 1 If  $\dot{Q}$  does not depend on  $Y_0, \dots, Y_s$  return  $\emptyset$ .
  - 2 Let  $s^*$  be the largest  $j \in \{0, \dots, s\}$  for which  $\dot{Q}$  depends on  $Y_j$ .
  - 3 Set  $\mathcal{L}_1 \leftarrow \emptyset$  and

$$\dot{W}_1 \leftarrow \left\{ w \in \dot{W} \mid \frac{\partial \dot{Q}}{\partial Y_{s^*}}(w) \neq 0 \right\}.$$

- 4 **for**  $w = (\alpha, w_0, \dots, w_s) \in \dot{W}_1$  **do**
- 5     Assuming there exists some polynomial  $g \in \mathbb{F}_q[X]$  such that  $\dot{Q} \circ \overline{dg} = 0 \in \mathbb{F}_q[X]$  and  $g^{(i)}(\alpha) = w_i$  for all  $0 \leq i \leq s$ , find the unique values  $w_{s+1}, \dots, w_{n-1}$  such that  $g^{(i)}(\alpha) = w_i$  for all  $0 \leq i < n$ . Such a unique solution exists by Lemma 15.
- 6     Define
 
$$f(x) = \sum_{i=0}^{n-1} \frac{w_i}{i!} (x - \alpha)^i.$$
- 7     If  $\Gamma(f) \subseteq \dot{W}$  add  $f$  to  $\mathcal{L}_1$ .
- 8 **Set**

$$\dot{W}_0 \leftarrow \left\{ w \in \dot{W} \mid \frac{\partial \dot{Q}}{\partial Y_{s^*}}(w) = 0 \right\}.$$

- 9  $\mathcal{L}_0 \leftarrow \text{Solve}(\frac{\partial \dot{Q}}{\partial Y_{s^*}}, \dot{W}_0)$

10 **return**  $\mathcal{L}_0 \cup \mathcal{L}_1$

---



With that the de-condensing algorithm is:

■ **Algorithm 2** Decondensing.

---

**Input:** Parameters  $q, s, n$ , the condenser  $\Gamma : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^{(s+2)}$ , and a set  $W \subseteq \mathbb{F}_q^{s+2}$

**Output:** All  $f \in \mathbb{F}_q^{<n}[X]$  such that  $\Gamma(f) \subseteq W$

---

- 1 Set  $D \leftarrow \left\lceil n \cdot \left[ |W| \cdot (s+2)! \right]^{\frac{1}{s+2}} \right\rceil$
- 2 Construct a non-zero polynomial  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  with

$$\deg_{(1,n,\dots,n-s)} Q \leq D$$

that vanishes on  $W$ .

- 3 **return**  $Solve(Q, W)$
- 

#### 4.1 Analysis of *Solve*

► **Lemma 18** (Correctness of *Solve*). *Fix a non-zero polynomial  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  such that  $\deg_{(1,n,\dots,n-s)}(Q) < q$ , and  $W \subseteq \mathbb{F}_q^{s+2}$ . Every  $f \in \mathbb{F}_q^{<n}[T]$  for which  $\Gamma(f) \subseteq W$  and  $Q(x, f(x), \dots, f^{(s)}(x)) = 0$  appears in the output of  $Solve(Q, W)$ .*

**Proof.** The proof is by induction on the degree of  $Q$  as a polynomial in  $Y_0, \dots, Y_s$ , i.e.,  $\deg_{(0,1,\dots,1)}(Q)$ . In the base case  $Q$  depends only on  $X$ , thus  $Q = Q(X)$ . As  $Q \neq 0$ , there are no solutions to  $Q(T, f(T), \dots, f^{(s)}(T)) = Q(T) = 0$  and  $\mathcal{L} = \emptyset$  is the correct output.

Now let  $f(T) \in \mathbb{F}_q^{<n}[T]$  such that  $\Gamma(f) \subseteq W$  and  $Q(T, f(T), \dots, f^{(s)}(T)) = 0$ . We have two cases:

1.  $\frac{\partial Q}{\partial Y_{s^*}}(T, f(T), \dots, f^{(s)}(T)) \neq 0$ . Note that

$$\begin{aligned} \deg \left( \frac{\partial Q}{\partial Y_{s^*}}(T, f(T), \dots, f^{(s)}(T)) \right) &\leq \deg_{(1,n,\dots,n-s)} \left( \frac{\partial Q}{\partial Y_{s^*}}(X, Y_0, \dots, Y_s) \right) \\ &\leq \deg_{(1,n,\dots,n-s)}(Q(X, Y_0, \dots, Y_s)) < q. \end{aligned}$$

Therefore there must be some  $\alpha \in \mathbb{F}_q$  for which

$$\frac{\partial Q}{\partial Y_{s^*}}(\alpha, f(\alpha), \dots, f^{(s)}(\alpha)) \neq 0.$$

As  $(\alpha, f(\alpha), \dots, f^{(s)}(\alpha)) \in \Gamma(f) \subseteq W$ , in the for loop we iterate over this vector and therefore in line 5 we find the unique solution of the ODE with these initial conditions, and because of the uniqueness this solution must be  $f$ . As  $\Gamma(f) \subseteq W$  we add it to the list  $\mathcal{L}$  in line 7.

2.  $\frac{\partial Q}{\partial Y_{s^*}}(T, f(T), \dots, f^{(s)}(T)) = 0$ . We notice that in this case  $\Gamma(f) \subseteq W_0$ , as for every  $\alpha \in \mathbb{F}_q$  we have  $\frac{\partial Q}{\partial Y_{s^*}}(\alpha, f(\alpha), \dots, f^{(s)}(\alpha)) = 0$ . Also  $\deg_{(0,1,\dots,1)}(\frac{\partial Q}{\partial Y_{s^*}}) < \deg_{(0,1,\dots,1)}(Q)$ , hence by induction  $f \in \mathcal{L}_0$ . ◀

► **Lemma 19** (List size of *Solve*). *For every non-zero  $Q \in \mathbb{F}_q[X, Y_0, \dots, Y_s]$  with  $\deg_{(1,n,n-1,\dots,n-s)}(Q) \leq D < q$  and every  $W \subseteq \mathbb{F}_q^{s+2}$ , the size of the output of  $Solve(Q, W)$  is at most  $\frac{|W|}{q-D}$ .*

## 12:12 Unbalanced Expanders from Multiplicity Codes

**Proof.** We prove by induction on the  $(0, 1, \dots, 1)$ -degree of  $Q$ . If  $\deg_{(0,1,\dots,1)}(Q)$  is zero, the list is empty, the list size is zero and the claim holds. We next prove the induction step.

For every  $w = (\alpha, w_0, \dots, w_s) \in W_1$ , there exists a unique  $f$  that may be joined to the list. Furthermore, since  $w \in W_1$  we have that:

$$\frac{\partial Q}{\partial Y_{s^*}}(\alpha, f(\alpha), \dots, f^{(s)}(\alpha)) = \frac{\partial Q}{\partial Y_{s^*}}(\alpha, w_0, \dots, w_s) \neq 0,$$

thus  $\frac{\partial Q}{\partial Y_{s^*}}(T, f(T), \dots, f^{(s)}(T)) \neq 0$ , and its degree is at most  $D$ , meaning that it equals 0 for at most  $D$  values of  $T$ , hence it is non-zero for at least  $q - D$  values of  $T \in \mathbb{F}_q$ . Also, if  $f$  appears in the list then  $\Gamma(f) \subseteq W$ . Hence, each of those  $q - D$  values lies in  $W$  (and therefore in  $W_1$ ) and reconstructs  $f$ . We conclude that  $f$  is reconstructed from at least  $q - D$  different points in  $W_1$ , thus  $|\mathcal{L}_1| \leq \frac{|W_1|}{q-D}$ .

We remain with the list size of  $\mathcal{L}_0$  which is obtained from  $\text{Solve}(\frac{\partial Q}{\partial Y_{s^*}}, W_0)$ . Since  $\deg_{(0,1,\dots,1)}(\frac{\partial Q}{\partial Y_{s^*}}) < \deg_{(0,1,\dots,1)}(Q)$ , and the  $(1, n, \dots, n-s)$ -weighted degree of  $\frac{\partial Q}{\partial Y_{s^*}}$  is at most  $D$ , we know by induction that  $|\mathcal{L}_0| \leq \frac{|W_0|}{q-D}$ . Altogether,  $|\mathcal{L}| \leq \frac{|W_1|}{q-D} + \frac{|W_0|}{q-D} = \frac{|W|}{q-D}$ . ◀

### 4.2 Putting it together

**Proof of Theorem 3.** By Lemma 7 it is enough to prove that for every  $W \subseteq \mathbb{F}_q^{s+2}$  of size at most  $AK - 1$  we have  $|\text{LIST}(W)| < K$ . Fix a set  $W \subseteq \mathbb{F}_q^{s+2}$  of size  $AK - 1 < qK$ . Let  $Q$  be as in Claim 16, with

$$\begin{aligned} D &= \left\lceil n \cdot \left[ qK \cdot (s+2)! \right]^{\frac{1}{s+2}} \right\rceil \leq n \cdot (qK)^{\frac{1}{s+2}} \cdot (((s+2)!)^{\frac{1}{s+2}} + 1) \\ &\leq \frac{n(s+2)}{2} \cdot (qK)^{\frac{1}{s+2}} = q - A \end{aligned}$$

Where the second to last inequality is due to the fact that  $(k!)^{1/k} + 1 \leq \frac{k}{2}$  for every  $k \geq 15$ . Let  $\mathcal{L}$  be the output list of  $\text{Solve}(Q, W)$ . Then,

$$|\text{LIST}(W)| \leq |\mathcal{L}| \leq \frac{|W|}{q-D} \leq \frac{AK-1}{q-D} < K,$$

where the first inequality is by Lemma 18, the second by Lemma 19 and the last inequality by using the fact that  $A \leq q - D$ . ◀

By choosing the parameters of in the same way as done in [7, Theorem 3.5] we get the following expander

► **Theorem 20.** *For every positive integers  $N$ ,  $K_{\max} \leq N$ , all  $\epsilon > 0$ , and  $\frac{16 \log(\frac{\log N}{\epsilon})}{\sqrt{\log K_{\max}}} \leq \alpha \leq 1$ , there is an  $M \leq D \cdot K_{\max}^{1+\alpha}$  and an explicit  $(\leq K_{\max}, (1-\epsilon)D)$  expander  $\Gamma : [N] \times [D] \rightarrow [M]$  with degree  $D = O((\log N(\log K_{\max}))/\epsilon)^{1+1/\alpha}$ .*

For completeness we repeat the proof from [7].

**Proof.** Let  $n = \log N$  and  $k = \log K_{\max}$ . Let  $h_0 = (2nk/\epsilon)^{1/\alpha}$ ,  $h = \lceil h_0 \rceil$ , and let  $q$  be a prime in the interval  $(h^{1+\alpha}/2, h^{1+\alpha}]$ .

Set  $s+2 = \lceil k/\log h \rceil$ , so that  $h^{s+1} \leq K_{\max} \leq h^{s+2}$ . As  $15 \leq s+2 \leq n \leq q = \text{char}(\mathbb{F}_q)$ , by Theorem 3, the graph  $\Gamma : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^{s+2}$  is a  $(\leq h^{s+2}, A)$  expander for  $A = q - \frac{n(s+2)}{2} \cdot (qK)^{\frac{1}{s+2}}$ , because  $K_{\max} \leq h^{s+2}$ , it is also a  $(\leq K_{\max}, A)$  expander.

Note that the number of left-vertices in  $\Gamma$  is  $q^n \geq N$ , and the number of right-vertices is

$$M = q^{s+2} \leq q \cdot h^{(1+\alpha)(s+1)} \leq q \cdot K_{max}^{1+\alpha}$$

The degree is

$$\begin{aligned} D = q &\leq h^{1+\alpha} \leq (h_0 + 1)^{1+\alpha} \\ &= O(h_0^{1+\alpha}) = O((nk/\epsilon)^{1+1/\alpha}) \end{aligned}$$

Lastly, we consider the expansion factor,  $A = q - \frac{n(s+2)}{2} \cdot (qK)^{\frac{1}{s+2}} \geq q - \frac{nkqh^{\frac{1}{s+2}}}{2}$ , of the graph, first notice

$$nk h \leq \epsilon \frac{h^{1+\alpha}}{2} \leq \epsilon q$$

where the first equality is due to the fact that  $nk/\epsilon \leq h^\alpha/2$ . Secondly, we can convert our lower bound on  $\alpha$  to a lower bound on  $k$

$$k \geq \frac{256}{\alpha^2} \log\left(\frac{n}{\epsilon}\right)$$

and by using it we get

$$\begin{aligned} s+2 &\geq \frac{k}{\log h} \geq \frac{\frac{256}{\alpha^2} \log^2\left(\frac{n}{\epsilon}\right)}{\log h} \geq \frac{\frac{64}{\alpha^2} \log^2\left(\frac{nk}{\epsilon}\right)}{\log h} \geq \frac{\frac{16}{\alpha^2} \log^2\left(2\frac{nk}{\epsilon}\right)}{\log h} \\ &= \frac{16 \log^2 h_0}{\log h} \geq \frac{4 \log^2 h}{\log h} = 4 \log h \geq (1+\alpha) \log h \geq \log q \end{aligned}$$

by combining the two inequalities

$$\frac{nkqh^{1/(s+2)}}{2} = nk h \cdot \frac{q^{1/(s+2)}}{2} \leq \epsilon q.$$

By substituting back to  $A$  we get  $A \geq (1-\epsilon)q = (1-\epsilon)D$ , which concludes the proof. ◀

Finally, Theorem 2 is an immediate consequence of Lemma 8 applied to Theorem 20.


---

## References


- 1 Nir Aviv and Amnon Ta-Shma. On the entropy loss and gap of condensers. *ACM Trans. Comput. Theory*, 11(3):15:1–15:14, 2019. doi:10.1145/3317691.
- 2 Aharon Gavriel Bege-Dov. Lower and upper bounds for the number of lattice points in a simplex. *SIAM Journal on Applied Mathematics*, 22(1):106–108, 1972. doi:10.1137/0122012.
- 3 Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2014. doi:10.1007/978-3-642-55220-5\_6.
- 4 Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. *SIAM J. Comput.*, 42(6):2305–2328, 2013. doi:10.1137/100783704.
- 5 Sebastian Falkensteiner, Yi Zhang, and Thieu N. Vo. On existence and uniqueness of formal power series solutions of algebraic ordinary differential equations. *Mediterranean Journal of Mathematics*, 19(2):74, February 2022. doi:10.1007/s00009-022-01984-w.

- 6    Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Comb.*, 28(4):415–440, 2008. doi:10.1007/s00493-008-2259-3.
- 7    Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. doi:10.1145/1538902.1538904.
- 8    Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Trans. Inf. Theory*, 59(6):3257–3268, 2013. doi:10.1109/TIT.2013.2246813.
- 9    Swastik Kopparty. List-decoding multiplicity codes. *Theory Comput.*, 11:149–182, 2015. doi:10.4086/toc.2015.v011a005.
- 10   Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014. doi:10.1145/2629416.
- 11   Maksim Amiryanyovich Limonov. Generalized separants of differential polynomials. *Moscow University Mathematics Bulletin*, 70(6):248–252, 2015. doi:10.3103/S0027132215060029.
- 12   Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 285–294. IEEE Computer Society, 2005. doi:10.1109/SFCS.2005.29.
- 13   Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discret. Math.*, 13(1):2–24, 2000. doi:10.1137/S0895480197329508.
- 14   Joseph Fels Ritt. *Differential algebra*, volume 33. American Mathematical Soc., 1950.
- 15   Amnon Ta-Shma and Christopher Umans. Better condensers and new extractors from parvaresh-vardy codes. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 309–315. IEEE Computer Society, 2012. doi:10.1109/CCC.2012.25.
- 16   Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Comb.*, 27(2):213–240, 2007. doi:10.1007/s00493-007-0053-2.
- 17   Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004. doi:10.1109/TIT.2004.838377.

# Streaming Algorithms with Large Approximation Factors

Yi Li 


Division of Mathematical Sciences, Nanyang Technological University, Singapore

Honghao Lin 

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

David P. Woodruff 

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Yuheng Zhang 

Zhiyuan College, Shanghai Jiao Tong University, China

---

## Abstract

We initiate a broad study of classical problems in the streaming model with insertions and deletions in the setting where we allow the approximation factor  $\alpha$  to be much larger than 1. Such algorithms can use significantly less memory than the usual setting for which  $\alpha = 1 + \epsilon$  for an  $\epsilon \in (0, 1)$ . We study large approximations for a number of problems in sketching and streaming, assuming that the underlying  $n$ -dimensional vector has all coordinates bounded by  $M$  throughout the data stream:

1. For the  $\ell_p$  norm/quasi-norm,  $0 < p \leq 2$ , we show that obtaining a  $\text{poly}(n)$ -approximation requires the same amount of memory as obtaining an  $O(1)$ -approximation for any  $M = n^{\Theta(1)}$ , which holds even for randomly ordered streams or for streams in the bounded deletion model.
2. For estimating the  $\ell_p$  norm,  $p > 2$ , we show an upper bound of  $O(n^{1-2/p}(\log n \log M)/\alpha^2)$  bits for an  $\alpha$ -approximation, and give a matching lower bound for linear sketches.
3. For the  $\ell_2$ -heavy hitters problem, we show that the known lower bound of  $\Omega(k \log n \log M)$  bits for identifying  $(1/k)$ -heavy hitters holds even if we are allowed to output items that are  $1/(\alpha k)$ -heavy, provided the algorithm succeeds with probability  $1 - O(1/n)$ . We also obtain a lower bound for linear sketches that is tight even for constant failure probability algorithms.
4. For estimating the number  $\ell_0$  of distinct elements, we give an  $n^{1/t}$ -approximation algorithm using  $O(t \log \log M)$  bits of space, as well as a lower bound of  $\Omega(t)$  bits, both excluding the storage of random bits, where  $n$  is the dimension of the underlying frequency vector and  $M$  is an upper bound on the magnitude of its coordinates.
5. For  $\alpha$ -approximation to the Schatten- $p$  norm, we give near-optimal  $\tilde{O}(n^{2-4/p}/\alpha^4)$  sketching dimension for every even integer  $p$  and every  $\alpha \geq 1$ , while for  $p$  not an even integer we obtain near-optimal sketching dimension once  $\alpha = \Omega(n^{1/q-1/p})$ , where  $q$  is the largest even integer less than  $p$ . The latter is surprising as it is unknown what the complexity of Schatten- $p$  norm estimation is for constant approximation; we show once the approximation factor is at least  $n^{1/q-1/p}$ , we can obtain near-optimal sketching bounds.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** streaming algorithms,  $\ell_p$  norm, heavy hitters, distinct elements

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.13

**Category** RANDOM

**Related Version** Full Version: <https://arxiv.org/abs/2207.08075>

**Funding** Yi Li: Supported in part by a Singapore Ministry of Education (MOE) AcRF Tier 1 grant RG75/21.

Honghao Lin: Supported in part by National Science Foundation (NSF) grant No. CCF-1815840.

David P. Woodruff: Supported in part by National Science Foundation (NSF) grant No. CCF-1815840.



© Yi Li, Honghao Lin, David P. Woodruff, and Yuheng Zhang;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 13; pp. 13:1–13:23



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

The data stream model is an important model for analyzing massive datasets, where the sheer size of the input imposes severe restrictions on the resources available to an algorithm. Such algorithms have only a small amount of memory and can only make a few passes over the data. Given a stream of elements from some universe, the algorithm maintains a short sketch, or summary, of what it has seen. Often such sketches are linear, which has multiple benefits, e.g., (1) the sketches can handle both insertions and deletions of items, and (2) the sketches are mergeable, meaning that given the sketch of a stream  $S$  and the sketch of a stream  $S'$ , the sketch of the concatenation of streams  $S$  and  $S'$  is the sum of the two sketches.

Many streaming algorithms have been developed to study fundamental problems in databases, such as estimating the number  $\ell_0$  of distinct elements, which is useful for query optimization and data mining. Among other things, this statistic can be used for selecting a minimum cost query plan [47], the design of databases [24], OLAP [44, 48], data integration [17, 20], and data warehousing [1]. Other important streaming problems include finding the heavy hitters, also known as the top- $k$ , most popular items, frequent items, elephants, or iceberg queries. These can be used in association rules and frequent itemsets [2, 27, 28, 46, 50], and for iceberg queries and iceberg datacubes [11, 23, 26]. Other important applications include estimating the frequency moments  $F_p$  [3], which for  $p \geq 1$  correspond to the  $p$ -th power of the  $\ell_p$  norm of the vector of frequencies of items, where the frequency of an item is its number of occurrences in the stream. For  $p \geq 2$ ,  $F_p$  indicates the degree of skew of the data, which may determine the selection of algorithms for data partitioning [21]. The case  $p = 2$  is the self-join size, which is useful for algorithms involving joining a relation with itself. The frequency moments of a vector are special cases of the Schatten- $p$  norms of a matrix, and there is a large body of work in the data stream model studying these intriguing norms [42, 43, 41, 15, 16], as well as the related cascaded norms [19, 5, 31, 6].

Given that the memory of a data stream algorithm is often significantly sublinear in the size of a stream  $\mathcal{S}$ , such algorithms are usually both randomized and approximate, and very often come with a guarantee that for a function  $f(\mathcal{S})$ , the output  $X$  of the algorithm satisfies that  $(1 - \epsilon)f(\mathcal{S}) \leq X \leq (1 + \epsilon)f(\mathcal{S})$ , with probability at least  $2/3$  over the coin tosses of the algorithm, where  $\epsilon \in (0, 1)$  is a parameter of the algorithm. Here the  $2/3$  probability can typically be amplified to  $1 - \delta$  by repeating the algorithm  $O(\log(1/\delta))$  times independently and outputting the median estimate. While a large body of work in the last two decades has resolved the space complexity of many of the aforementioned problems for  $\epsilon \in (0, 1)$ , for certain applications the lower bounds on the space complexity may be too large to be useful. For such applications it is therefore natural to allow for a larger approximation factor  $\alpha > 1$ , in the hope of obtaining a smaller amount of memory. Namely, one could instead ask for the output  $X$  of the streaming algorithm to satisfy  $f(\mathcal{S}) \leq X \leq \alpha \cdot f(\mathcal{S})$ . This motivates our main question:

*What is the space complexity of classical streaming problems when the approximation factor  $\alpha$  is allowed to be much larger than 1?*

Perhaps surprisingly, this question does not seem to be well-understood, and is in fact open for all of the abovementioned problems in a data stream. There are a few related works, such as [18], which studies large approximation factors for *deterministically* estimating the number of distinct elements,  $\ell_p$ -estimation, entropy estimation, as well as maximum matching size in a graph stream; see also [7] for large approximation factor lower bounds for randomized algorithms for maximum matching. Other streaming problems where large approximation

factors were studied include dynamic time warping [14], maximum  $k$ -coverage [29] and the  $p$ -to- $q$  norms [37]. In contrast to [18], our focus is on tight bounds for randomized algorithms, for which significantly less memory can be achieved than deterministic algorithms, and for a wide range of fundamental problems in the data stream model that do not appear to have been studied before for large approximation factors.

## 1.1 Our Results

A summary of our upper and lower bounds for a number of data stream problems can be found in Tables 1 and 2.

For estimating the  $\ell_p$  norm for  $0 < p \leq 2$ , we show that obtaining a  $\text{poly}(n)$ -approximation requires the same amount of memory as obtaining an  $O(1)$ -approximation, under the common assumption that  $M = \text{poly}(n)$ . Namely, we show an  $\Omega(\log n)$  lower bound even with a random oracle for these problems. Previously, only an  $\Omega(1)$  lower bound was known for  $\text{poly}(n)$ -approximation in this setting. Our result also holds if the stream is randomly ordered, or in the bounded deletion model [30], in which deletions are allowed but the norm should not drop by more than a constant factor from what it was at a previous point in time. Our lower bound can also be extended to a wide class of statistical  $M$ -estimators. We also show a two-pass algorithm that uses less space than the best existing one-pass algorithm.

For estimating the  $\ell_p$  norm of an underlying  $n$ -dimensional vector,  $p > 2$ , we show an upper bound of  $O(n^{1-2/p}(\log n \log M)/\alpha^2)$  bits for  $\alpha$ -approximation for any  $\alpha > 1$ , and a matching lower bound for almost the full range of  $\alpha$  on the bit complexity of linear sketches, which gives a matching streaming lower bound under the conditions of [40], though these conditions can be restrictive, see, e.g., [34] for discussion. One important motivation for studying such norms is to data-augmented streaming algorithms. For example, it was shown in [32] that for estimating the  $\ell_p$  norm with a so-called learned oracle, one can achieve an  $O(1)$ -approximation using  $\tilde{O}(n^{1/2-1/p})$  bits of space. However, this requires a successfully trained oracle, which could have an arbitrarily bad approximation in the worst case. By instead running our worst-case  $\tilde{O}(n^{1/4-1/(2p)})$ -approximation algorithm for  $\ell_p$  estimation with  $\tilde{O}(n^{1/2-1/p})$  bits of memory in parallel, we can ensure that we do at least as well as the learned algorithm in the same amount of memory (up to a constant factor), but we can ensure we never return worse than an  $\tilde{O}(n^{1/4-1/(2p)})$ -approximation. Another important consequence of our  $\ell_p$ -estimation algorithm is that it can be used as a subroutine to obtain large approximations for the  $(p, q)$ -cascaded-norm ( $p \geq 1, q > 2$ ) and rectangle  $\ell_p$  ( $p > 2$ ) problems, showing that the previous space bounds can be reduced by an  $\alpha^2$  factor for an  $\alpha$ -approximation. These results are shown in Sections E and F.

In the  $\ell_2$ -heavy hitters problem, the goal is to output a subset  $S$  of  $\{1, 2, \dots, n\}$  which contains every  $i$  for which  $x_i^2 \geq \frac{1}{k}\|x\|_2^2$ , and no  $i$  for which  $x_i^2 < \frac{1}{2k}\|x\|_2^2$ . It is known [8, 33] that the space complexity of this problem is  $\Theta(k \log n \log M)$  bits, if we are promised that  $x \in \{-M, \dots, M\}^n$ . A natural relaxation would be to instead require only that  $S$  contains every index  $i$  for which  $x_i^2 \geq \frac{1}{k}\|x\|_2^2$  and no  $i$  for which  $x_i^2 < \frac{1}{\alpha k}\|x\|_2^2$ . We show a strong negative result, that for any  $\alpha = O((n/k)(\log \log n)^2/(\log n)^2)$ , this problem still requires  $\Omega(k \log n \log M)$  bits of memory for any linear sketch, which gives a matching streaming lower bound under the conditions of [40]. For our bit complexity lower bound we assume the algorithm succeeds with probability  $1 - O(1/n)$ , while our sketching dimension lower bound only requires that the algorithm succeeds with constant probability. Interestingly, the proofs of our lower bounds do not use the usual hard instances for finding  $\ell_2$ -heavy hitters [8, 33], and instead use a hard instance for  $\ell_p$ -estimation in [51].



For estimating the number  $\ell_0$  of distinct elements, we show that to obtain an  $\alpha = n^{1/t}$ -approximation, an upper bound of  $O(t \log \log M)$  bits is possible and there is a lower bound of  $\Omega(t)$  bits, where  $n$  denotes the dimension of the underlying frequency vector and  $M$  is an upper bound on the absolute value of its coordinates. We state our results in the random oracle model, where a public random string is known to the algorithm. Without such a random string, a simple reduction from the Equality communication problem gives an  $\Omega(\log n)$  bit lower bound for any multiplicative approximation, see, e.g., [3] for similar arguments<sup>1</sup>. Nevertheless, our results are still interesting outside of the random oracle model, since in the common setting of  $M \leq \text{poly}(n)$ , setting  $t = (\log n) / \log \log n$  gives us an  $O(\log n)$ -approximation with  $O(\log n)$  bits of memory, and since  $O(\log n)$  bits of randomness is also sufficient, this matches the  $\Omega(\log n)$  bit lower bound from the Equality problem. The previous best algorithm [36] required at least  $O(\log n \log \log M)$  bits for any multiplicative approximation factor. We also study estimating the number of distinct elements in two and three passes, showing a separation for the problem between one and two passes and a near-optimal three-pass algorithm.

The Schatten- $p$  norm of an  $n \times n$  input matrix  $A$  is just the  $\ell_p$ -norm of the vector of singular values of  $A$ . For  $\alpha$ -approximation to the Schatten- $p$  norm, we give a linear sketch of dimension  $\tilde{O}(n^{2-4/p}/\alpha^4)$ , which is optimal up to logarithmic factors, for every even integer  $p$  and every  $\alpha \geq 1$ , while for  $p$  not an even integer we obtain a near-optimal sketch dimension of  $\tilde{O}(n^{2-4/p}/\alpha^4)$  once  $\alpha = \Omega(n^{1/q-1/p})$ , where  $q$  is the largest even integer less than  $p$ . Interestingly, we obtain the first near-optimal multiplicative approximations for Schatten- $p$  norms for non-integer  $p$  for a wide range of non-trivial approximation factors  $\alpha$ , whereas it is still unknown and a major open question (see, e.g., [41] for discussion) to obtain optimal multiplicative approximations for Schatten- $p$  norms for non-integer  $p$  when  $\alpha = O(1)$ . Our work highlights that surprisingly, the difficulty of this problem stems from the approximation factor rather than the problem being hard for every approximation factor.

## 1.2 Our Techniques

For our lower bound for estimating  $\ell_p$ -norms for  $0 < p \leq 2$  (or more generally for  $M$ -estimators), we give a reduction from the coin problem introduced in [12] and strengthened in [13]. Consider a sequence of independent coin flips with either a heads probability of  $1/2 + \beta$  or a heads probability of  $1/2 - \beta$ . The coin problem asks us to distinguish between the two cases with the fewest number of flips. Given a sequence of  $n$  coin flips, for an underlying vector  $x$  in a stream we can perform  $x_1 \leftarrow x_1 + 1$  or  $x_1 \leftarrow x_1 - 1$ , depending on whether the coin is a heads or a tail. To ensure a bounded deletion stream, we initialize  $x = (2n\beta, 0, \dots, 0)$ . Then, with constant probability, we have  $x_1 = 4n\beta \pm O(\sqrt{n})$  in one case and  $x_1 = \pm O(\sqrt{n})$  in the other, resulting in an  $\alpha$ -factor difference in the  $\ell_p$ -norm when  $4n\beta = \Omega(\alpha\sqrt{n})$ . Note that our goal is to obtain a lower bound for  $\alpha = \omega(1)$ . The earlier lower bound for the coin problem [12] instead considers  $\beta \sim 1/\sqrt{n}$ , which only translates into  $\alpha = \Theta(1)$  at best, for which we know an upper bound of  $O(\log n)$  words exists. The newer result [13] shows an  $O(\log n)$  bit lower bound for  $\beta < n^{1/3-\epsilon}$ . Such a  $\beta$  translates into  $\alpha = n^{\Omega(1)}$ , as desired. This is also the first application of the newer result [13] to data streams.

<sup>1</sup> Briefly, Alice has  $x \in \{0, 1\}^n$  and inserts  $i$  for which  $x_i = 1$ . Bob has  $y \in \{0, 1\}^n$  and deletes  $i$  for which  $y_i = 1$ . If  $x = y$  then  $\ell_0 = 0$ , otherwise it is non-zero, and the private coin randomized communication complexity of Equality is  $\Omega(\log n)$  bits.



■ **Table 1** Summary of previous results and the results obtained in this work. We assume that  $M = \text{poly}(n)$  and  $p$  is constant. The reported space bounds are measured in bits except for the Schatten- $p$  norm, where we consider the sketching dimension. In this table,  $\tilde{\Omega}(f)$  denotes  $\Omega(f \text{ poly log } f)$  and, for the rectangle  $\ell_p$  estimation problem,  $O^*(f)$  denotes  $f \cdot \text{poly}(d, \log(mn/\delta))$ . For the  $\ell_2$ -heavy hitters problem, both our lower bound and our upper bound for bit complexity assume that the success probability is at least  $1 - O(1/n)$ , while for the sketching dimension results we assume constant success probability.

| Problem                                         |                  | Large Approx. Ratio                       |            | Constant Approx. Ratio                       |
|-------------------------------------------------|------------------|-------------------------------------------|------------|----------------------------------------------|
| $\ell_p$ Estimation ( $0 < p \leq 2$ )          | $\text{poly}(n)$ | $\Omega(\log n)$                          | Thm 7      | $O(\log n)$ [35]<br>$\Omega(\log n)$ [35]    |
| $\ell_p$ Estimation ( $p > 2$ )                 | $\alpha$         | $\tilde{O}(n^{1-2/p}/\alpha^2)$           | Thm 12     | $\tilde{O}(n^{1-2/p})$ e.g. [6]              |
|                                                 |                  | $\tilde{\Omega}(n^{1-2/p}/\alpha^2)$      | Thm 18     | $\tilde{\Omega}(n^{1-2/p})$ e.g. [51]        |
| $\ell_2$ Heavy Hitters                          | $\tilde{O}(n/k)$ | $\Omega(k \log^2 n)$                      | Thm 21     | $O(k \log^2 n)$<br>$\Omega(k \log^2 n)$ [33] |
| $\ell_2$ Heavy Hitters<br>(Sketching Dimension) | $\tilde{O}(n/k)$ | $\Omega(k \log n)$                        | Thm 24     | $O(k \log n)$<br>$\Omega(k \log n)$ [45]     |
| Distinct Elements                               | $n^{1/t}$        | $O(t \log \log n)$                        | Thm 25     | $O(\log n \log \log n)$ [36]                 |
|                                                 |                  | $\Omega(t)$                               | Thm 29     | $\Omega(\log n \log \log n)$ [52]            |
| Schatten- $p$ Norm                              | $\alpha$         | $\tilde{O}(n^{2-4/p}/\alpha^4)$           | Thm 35, 36 | $O(n^{2-4/p})$ even $p$ [41]                 |
|                                                 |                  | $\Omega(n^{2-4/p}/\alpha^4)$              | Thm 38     | $\Omega(n^{2-4/p})$ [41]                     |
| Cascaded Norm<br>( $p, q > 2$ )                 | $\alpha$         | $\tilde{O}(n^{1-2/p} d^{1-2/q}/\alpha^2)$ | Thm 39     | $\tilde{O}(n^{1-2/p} d^{1-2/q})$ [6]         |
|                                                 |                  | $\Omega(n^{1-2/p} d^{1-2/q}/\alpha^2)$    | Thm 40     | $\Omega(n^{1-2/p} d^{1-2/q})$ [31]           |
| Cascaded Norm<br>( $1 \leq p < 2, q > 2$ )      | $\alpha$         | $\tilde{O}(d^{1-2/q}/\alpha^2)$           | Thm 39     | $\tilde{O}(d^{1-2/q})$ [6]                   |
|                                                 |                  | $\tilde{\Omega}(d^{1-2/q}/\alpha^2)$      | Thm 40     | $\Omega(d^{1-2/q})$ [31]                     |
| Rectangle $F_p$<br>Estimation ( $p > 2$ )       | $\alpha$         | $O^*(n^{d(1-2/p)}/\alpha^2)$              | Thm 41     | $O^*(n^{d(1-2/p)})$ [49]                     |
|                                                 |                  | $\Omega(n^{d(1-2/p)}/\alpha^2)$           | Thm 18     | $\Omega(n^{d(1-2/p)})$ [49]                  |

■ **Table 2** Summary of previous results and the results obtained in this work to obtain a  $(1 \pm \varepsilon)$ -approximation. The reported space bounds are measured in bit complexity.

| Problem                        | Type   | New Alg                                                                                      | Previous 1-pass Alg                                    |
|--------------------------------|--------|----------------------------------------------------------------------------------------------|--------------------------------------------------------|
| Distinct Elements              | 2-pass | $O(\log n + \varepsilon^{-2} \log \log M (\log(1/\varepsilon) + \log \log M))$<br>Theorem 30 | $O(\varepsilon^{-2} \log n \log \log nM)$<br>[36]      |
|                                | 3-pass | $O(\log n + \varepsilon^{-2} (\log(1/\varepsilon) + \log \log M))$<br>Theorem 31             | $\Omega(\varepsilon^{-2} \log n \log \log nM)$<br>[52] |
| $\ell_p$ Moment ( $p \leq 2$ ) | 2-pass | $O(\log n + \varepsilon^{-2} (\log M + \log 1/\varepsilon))$<br>Theorem 32                   | $O(\varepsilon^{-2} \log nM)$<br>[35]                  |

For our upper bound for estimating  $\ell_p$ -norms for  $p > 2$ , we connect the problem to an instance of the same problem with a different parameter. Namely, suppose that  $q$  is such that  $n^{1-2/q} = \Theta(n^{1-2/p}/\alpha^2)$ , where  $\alpha$  is the approximation factor. Then from relationships between norms we have  $\|x\|_p \leq \|x\|_q \leq \alpha \|x\|_p$ . Hence, a constant factor approximation to the  $\ell_q$  norm actually gives an  $\alpha$  approximation to the  $\ell_p$  norm. This “self-reduction” from an instance of the problem under one norm to an instance of the same problem under a different norm also helps us derive our algorithm for estimating the Schatten- $p$  norms of a matrix when  $\alpha = \Omega(n^{1/q-1/p})$ , where  $q$  is the largest even integer less than  $p$ . For our lower bound for  $\ell_p$ -norm estimation for  $p > 2$ , we consider the multiparty disjointness ( $\text{DISJ}_s^n$ ) problem in the public-coin simultaneous message passing model, which was initially proposed in [51]. We show that the hard instance can still give a matching lower bound for  $\alpha$ -approximation if we set the number of players appropriately.

For the  $\ell_2$  heavy hitters problem, the usual hard instances for this problem (see, e.g., [33] and [8]) fail to give an extra  $\log n$  factor for large approximations. The reason is that when reducing from the so-called Augmented Indexing problem, to make the two cases distinguishable for an  $\alpha$ -approximation, one would need to partition the vector into  $\log_\alpha(n)$  levels, which for  $\alpha = n^{\Omega(1)}$ , is only  $O(1)$ . In contrast, we consider the same multiparty disjointness problem we use for the  $\ell_p$  norm estimation problem and show that a similar hard instance gives a matching lower bound for the  $\ell_2$  heavy hitters problem with a large approximation factor. Thus, we use a fundamentally different hard instance for this problem.

For our upper bound for estimating the number  $\ell_0$  of distinct elements, suppose that the approximation factor  $\alpha = n^{1/t}$ . We sub-sample the input coordinates into  $t$  levels, with a geometrically decreasing sampling probability. In each level, the surviving coordinates are hashed into a constant number of buckets. If the  $\ell_0$  of the sub-sampled vector in a level is at most a constant, then only a small number of these buckets will be occupied. Based on this, we can find the specific level  $j^*$  for which the  $\ell_0$  norm in this level is between 0 and  $n^{1/t}$  and show that after rescaling it is a good estimator to the overall  $\ell_0$  of the original vector. To use less memory in each bucket, we choose a random prime  $p = \text{poly}(\log M)$  and only store each counter mod  $p$ . Our lower bound is based on a reduction from the Augmented Indexing problem mentioned above, which in more detail is a two player communication problem in which Alice holds a binary vector  $u \in \{0, 1\}^t$  and asks for Bob to recover  $u_i$  given  $u_{i+1}, \dots, u_t$ . We divide the vector  $x$  into  $l = \Theta(t)$  segments, where the  $i$ -th segment has length  $\Theta(n^{i/l})$ , and fill the  $i$ -th segment with all 1s if and only if  $u_i = 1$ . Then  $\|x\|_0$  differs by a factor of  $\Theta(n^{1/t})$  between the cases of  $u_i = 0$  and  $u_i = 1$ , whence an  $\Omega(t)$  lower bound follows. Despite the fact that a  $\log(1/\varepsilon)$ -factor gap remains in the upper and lower bounds for  $(1 \pm \varepsilon)$ -approximation for  $\ell_0$  (see, e.g., [22] for discussion), we obtain a tight  $\Theta(\log n)$  space bound for  $\alpha = \Theta(\log n)$  approximation, for example. Our bounds also show a separation between the estimation of the  $\ell_p$ -norm ( $0 < p \leq 2$ ) and the  $\ell_0$ -norm with an  $n^{\Theta(1)}$ -approximation factor, since we show an  $\Omega(\log n)$  lower bound via the coin problem for  $p > 0$  and  $n^{\Omega(1)}$  approximation, while we have an  $O(\log \log n)$  upper bound for  $p = 0$  and  $n^{O(1)}$  approximation.

We also consider multi-pass algorithms for  $\ell_0$  and  $\ell_p$  ( $0 < p \leq 2$ ) estimation. For the  $\ell_0$  estimation problem, we show that if we obtain an  $O(\log n)$ -approximation in the first pass, then we can obtain a  $(1 \pm \varepsilon)$ -approximation in the second pass using  $O(\varepsilon^{-2} \log \log M (\log(1/\varepsilon) + \log \log M))$  bits of space. This can be further reduced to  $O(\varepsilon^{-2} (\log(1/\varepsilon) + \log \log M))$  bits of space using a third pass. For  $\ell_p$  estimation, we show that if we can obtain a constant approximation  $Z$  in the first pass, then in the second pass, we can sample the coordinates with probability  $O(\varepsilon^{-2} M^p / Z)$ . Hence, we only need to generate certain  $p$ -stable random variables used in our algorithm with precision  $(M/\varepsilon)^{O(1)}$ , from which we obtain an  $O(\varepsilon^{-2} (\log M + \log(1/\varepsilon)))$  bits of space algorithm in the second pass, which is better than the previous result of  $O(\varepsilon^{-2} \log n M)$  when  $M$  is small.

## 2 Preliminaries

**Notation.** For a vector  $x \in \mathbb{R}^n$ , its  $\ell_p$  norm is  $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ , where  $p \geq 1$ . We also write  $F_p(x) = \|x\|_p^p$ . We also define  $\|x\|_\infty = \max_i |x_i|$ . When  $p < 1$ , the quantity  $\|x\|_p$  is not a norm though it is a well-defined quantity and  $\|x\|_p^p$  tends to the number of nonzero entries of  $x$  as  $p \rightarrow 0^+$ . In view of this limit, we denote the number of nonzero entries of  $x$  by  $\|x\|_0$  and also refer to it as  $\ell_0$ .

For a matrix  $A \in \mathbb{R}^{m \times n}$ , we define its Schatten- $p$  norm to be  $\|A\|_p = (\sum_{i=1}^{\min\{m,n\}} (\sigma_i(A))^p)^{\frac{1}{p}}$  for each  $p \geq 1$ , where  $\sigma_1(A) \geq \sigma_2(A) \geq \dots$  are the singular values of  $A$ . We also define the  $(p, q)$ -cascaded norm of  $A$  to be  $\|A\|_{p,q} = (\sum_i (\sum_j |A_{i,j}|^q)^{\frac{p}{q}})^{\frac{1}{p}}$  for  $p, q \geq 1$ .

**Turnstile streaming model.** In the turnstile model of data streams, there is an underlying  $n$ -dimensional vector  $x$  which is initialized to 0 and keeps receiving updates of the form  $(i, \Delta) \in [n] \times \mathbb{R}$ , which represents  $x_i \leftarrow x_i + \Delta$ . Here  $\Delta$  can be either positive or negative. In this paper we assume that the underlying vector is guaranteed to be bounded by  $M$ , i.e., it always holds that  $\|x\|_\infty \leq M$  throughout the data stream. The length of the stream is denoted by  $m$ . When the vector  $x$  is given by a stream  $\mathcal{S}$  in the turnstile model, we abuse notation and also write  $\ell_p(\mathcal{S})$  for  $\|x\|_p$ .

When the input describes a matrix  $A \in \mathbb{R}^{m \times n}$ , we can view the matrix as an  $mn$ -dimensional vector and each item in the stream updates an entry of the matrix.

A variant of the streaming model for a matrix  $A$  concerns rectangular updates. Here  $x$  is a tensor indexed by  $[n]^d$  and each update has the form  $(R, \Delta)$ , where  $R \subseteq [n]^d$  is a rectangle, representing the update  $x_i \leftarrow x_i + \Delta$  for all  $i \in R$ . The rectangle  $\ell_p$  problem is considered under this model (see, e.g., [49]), which asks to estimate  $\|A\|_p = (\sum_{i \in [n]^d} |x_i|^p)^{1/p}$ .

**Subspace Embeddings.** Suppose that  $A \in \mathbb{R}^{n \times d}$ . A matrix  $S \in \mathbb{R}^{m \times n}$  is called an  $(\varepsilon, \delta)$ -subspace-embedding for  $A$  if it holds with probability at least  $1 - \delta$  that  $(1 - \varepsilon) \|Ax\|_2 \leq \|SAx\|_2 \leq (1 + \varepsilon) \|Ax\|_2$  for all  $x \in \mathbb{R}^d$  simultaneously. A classical construction is to take  $S$  to be a Gaussian random matrix of i.i.d.  $N(0, 1/m)$  entries, where  $m = O((d + \log(1/\delta)/\varepsilon^2))$ . Recall the minimax characterization of singular values of a matrix  $A$ :  $\sigma_i(A) = \sup_H \inf_{x \in H: \|x\|_2=1} \|Ax\|_2$ , where the supremum is taken over all subspaces  $H$  such that  $\dim(H) = i$ . This implies (see e.g., Lemma 7.2 of [41]) that  $(1 - \varepsilon)\sigma_i(A) \leq \sigma_i(SA) \leq (1 + \varepsilon)\sigma_i(A)$  with probability at least  $1 - \delta$ , for all  $i = 1, \dots, \min\{m, n\}$ , i.e.,  $S$  preserves all singular values of  $A$  if  $S$  is an  $(\varepsilon, \delta)$ -subspace-embedding for  $A$ .

### 3 Lower Bound for $M$ -Estimators

We start by giving a very general lower bound for  $M$ -estimator estimation with a large approximation factor.  $M$ -estimators can be seen as generalizations of the  $p$ -th frequency moments of the underlying vector  $x$ . We first show this lower bound in the turnstile streaming model and later we will show that it still holds even in the bounded deletion and random order models.

► **Definition 1** ( $M$ -estimator with parameter  $\gamma$ ). Suppose  $G: \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$  is a function. We say  $\|y\|_G = \sum_i G(y_i)$  is an  $M$ -estimator with parameter  $\gamma$  if  $G$  satisfies the following conditions:

- $G(0) = 0$ ;
- $G(x) = G(-x)$ ;
- $G(x)$  is non-decreasing in  $|x|$ ;
- For all  $x, y$  with  $|y| > |x| > 0$ ,  $\frac{G(y)}{G(x)} \geq \left(\frac{y}{x}\right)^\gamma$ .

We will give a reduction from the following coin problem. In [13], the authors show an  $\Omega(\log n)$  lower bound even when the parameter  $\beta$  is allowed to be very small:

► **Definition 2** (Coin Problem). Let  $X_1, \dots, X_n$  be a stream of i.i.d. random bits, which either (1) comes from a distribution with heads probability  $\frac{1}{2} + \beta$  or (2) comes from a distribution with heads probability  $\frac{1}{2} - \beta$ . We are asked to distinguish these two cases at the end of the stream, with probability  $2/3$ .

► **Theorem 3** ([13]). *For all constant  $\varepsilon > 0$ , any length- $n$  Read-Once Branching Program that solves the coin problem for bias  $\beta < n^{-1/3-\varepsilon}$ , requires  $n^{\Omega(\varepsilon)}$  width.*

► **Corollary 4.** *For all constants  $\varepsilon > 0$ , any randomized streaming algorithm that solves the coin problem with bias  $\beta < n^{-1/3-\varepsilon}$  requires  $\Omega(\log n)$  space. This holds even if we give the algorithm access to an arbitrarily long random tape.*

Suppose that we are given an  $M$ -estimator with parameter  $\gamma$ . We define a distribution  $\mathcal{D}$  on the sequences of  $n$  random bits as follows: suppose that  $\beta = n^{-1/3-\varepsilon}$  for a small constant  $\varepsilon$ . Let  $X_1, \dots, X_n$  be a binary sequence coming from a distribution with heads probability  $\frac{1}{2}$  or a distribution with heads probability  $\frac{1}{2} + \beta$ , where  $X_i = 1$  if the  $i$ -th coin is a head and  $X_i = 0$  if the  $i$ -th coin is a tail. Let  $x$  be the underlying vector in the streaming algorithm. During the stream we perform updates  $x_1 \leftarrow x_1 + 1$  if  $X_i = 1$ , or  $x_1 \leftarrow x_1 - 1$  otherwise. We will show that any streaming algorithm that gives an  $O(n^{(1/6-\varepsilon)\gamma})$ -approximation for  $\|x\|_G$  can distinguish the above two distributions with large constant probability. We first analyze the sum  $|\sum_{i=1}^n X_i|$  for these two distributions. The following two lemmas can be easily proved using Chebyshev's inequality and thus the proofs are omitted.

► **Lemma 5.** *Suppose that the sequence  $(X_1, \dots, X_n) \in \{\pm 1\}^n$  comes from the distribution with heads probability  $\frac{1}{2}$ . Then with probability at least  $1 - 1/(4k)$ , we have  $|\sum_{i=1}^n X_i| \leq \sqrt{kn}$ .*

► **Lemma 6.** *Suppose that the sequence  $(X_1, \dots, X_n) \in \{\pm 1\}^n$  comes from the distribution with heads probability  $\frac{1}{2} + n^{-1/3-\varepsilon}$ . Then with probability at least  $1 - 1/(4k)$ , we have  $\sum_{i=1}^n X_i \geq 2n^{2/3-\varepsilon} - \sqrt{kn}$ .*

We are now ready to give our lower bound.

► **Theorem 7.** *Suppose that  $G$  is an  $M$ -estimator with parameter  $\gamma$ . Then any randomized streaming algorithm which outputs an  $O(M^{(1/6-\varepsilon)\gamma})$ -approximation to  $\|x\|_G$  with probability at least  $2/3$  requires  $\Omega(\log M)$  bits of space, excluding the storage for random bits.*

**Proof.** Suppose that we have a streaming algorithm which outputs an  $O(M^{(1/6-\varepsilon)\gamma})$ -approximation to  $\|x\|_G$ . We shall show that we can distinguish the two distributions with bias  $\beta$  in the coin problem with large constant probability.

We initialize the vector  $x = (0, 0, \dots, 0)$ . Suppose we have a stream of bits  $X_1, \dots, X_M$  coming from the distribution with heads probability  $\frac{1}{2}$  or with heads probability  $\frac{1}{2} + \beta = \frac{1}{2} + M^{-1/3-\varepsilon}$ . Then, during the stream, we perform the update  $x_1 \leftarrow x_1 + 1$  if  $X_i = 1$  and  $x_1 \leftarrow x_1 - 1$  otherwise.

Let  $x^{(0)}$  be the underlying vector if the heads probability for the distribution is  $\frac{1}{2}$  and  $x^{(1)}$  be the underlying vector if the heads probability is  $\frac{1}{2} + \beta$ . From Lemmas 5 and 6 we have that with probability at least  $9/10$ ,  $|x_1^{(0)}| = O(\sqrt{M})$  at the end of the stream, while  $|x_1^{(1)}| = \Omega(M^{2/3-\varepsilon})$  in the second case. It follows from the definition of  $\gamma$  that  $\frac{\|x^{(1)}\|_G}{\|x^{(0)}\|_G} = \Omega(M^{(1/6-\varepsilon)\gamma})$  for the two cases. This implies that if the streaming algorithm can output an  $O(n^{(1/6-\varepsilon)\gamma})$ -approximation to  $\|x\|_G$ , then we can distinguish the two distributions with bias  $\beta$  in the coin problem. From Corollary 4, such a streaming algorithm needs  $\Omega(\log M)$  bits of space. ◀

► **Corollary 8.** *Any randomized streaming algorithm outputting an  $O(M^{1/6-\varepsilon})$ -approximation to  $\|x\|_p^p$  requires  $\Omega(p \cdot \log M)$  bits of space, excluding the storage for random bits. Moreover, under the assumption that  $M = \text{poly}(n)$ , any randomized streaming algorithm outputting a  $\text{poly}(n)$ -approximation to  $\|x\|_p^p$  requires  $\Omega(p \cdot \log n)$  bits of space.*

### 3.1 Lower Bound in Other Streaming Models

**Bounded Deletions.** Our  $\Omega(\log M)$  lower bound still holds even with the assumption of bounded deletions. In this model, the updates  $\Delta_j$  can be positive or negative, but one is promised that the norm  $\|x\|_2$  never drops by more than an  $\alpha$ -fraction of what it was at any earlier point in the stream, for a constant parameter  $\alpha$ . We only state the theorem below, whose proof can be found in the full version.

► **Theorem 9.** *Suppose  $G$  is an  $M$ -estimator with parameter  $\gamma$ . Then any randomized streaming algorithm outputting an  $O(M^{(1/6-\varepsilon)\gamma})$ -approximation of  $\|x\|_G$  in the bounded deletion model needs  $\Omega(\log M)$  bits, excluding the storage for random bits.*

**Random Order.** In the random order model, we assume the updates  $\Delta_j$  come in a random order. We note that the updates for the distribution in Theorem 7 are a sequence of random  $\pm 1$  variables. Hence it satisfies the random order assumption automatically, which means we obtain the following theorem.

► **Theorem 10.** *Suppose  $G$  is an  $M$ -estimator with parameter  $\gamma$ . Then any randomized streaming algorithm which outputs an  $O(M^{(1/6-\varepsilon)\gamma})$ -approximation to  $\|x\|_G$  in the random order model requires  $\Omega(\log M)$  bits of space, excluding the storage for its random bits.*

## 4 $\ell_p$ Estimation $p > 2$

In this section, we consider the problem of estimating  $\|x\|_p$  with a large approximation factor when  $p > 2$ . We present an algorithm that gives an  $\alpha$ -approximation to  $\|x\|_p$  using  $\tilde{O}(n^{1-2/p}/\alpha^2)$  bits of space. We will also give a matching lower bound for this problem.

**Upper bound.** Suppose that we want an  $\alpha$ -approximation where  $n^{1-2/p}/\alpha^2 = \Omega(1)$  (otherwise there is a trivial  $\Omega(1)$  lower bound) and let  $q$  be the number such that  $n^{1-2/q} = \Theta(n^{1-2/p}/\alpha^2)$ . Then we have  $2 \leq q < p$ . The following lemma shows that  $\|x\|_q$  is an  $\alpha$ -approximation to  $\|x\|_p$ .

► **Lemma 11.** *Suppose that  $p \geq q \geq 2$  and  $\alpha \geq 1$  satisfies  $n^{1-2/q} = n^{1-2/p}/\alpha^2 = \Omega(1)$ . Then it holds that  $\|x\|_p \leq \|x\|_q \leq \alpha \|x\|_p$ .*

**Proof.** From our choice of  $q$ , we have that  $\alpha = n^{1/q-1/p}$ . The  $\ell_p$  norm is decreasing in  $p$ , and thus  $\|x\|_q \geq \|x\|_p$ . By Hölder's inequality, it also holds that  $\|x\|_q \leq n^{1/q-1/p} \|x\|_p = \alpha \|x\|_p$ . ◀

The preceding lemma shows that we can use any  $O(1)$ -approximation algorithm for  $\ell_q$  to obtain an  $\alpha$ -approximation to the  $\ell_p$  norm. For example, we can use the  $O(n^{1-2/q} \log^2 n)$ -bit algorithm of [4], or the algorithm of [25]. Our theorem follows immediately.

► **Theorem 12.** *Suppose that  $p > 2$  is a constant. There is an algorithm whose output is  $Z$ , which satisfies that  $\|x\|_p \leq Z \leq \alpha \|x\|_p$  with probability at least 0.9. Furthermore, the algorithm uses  $O(n^{1-2/p} \log n \log M/\alpha^2)$  bits of space.*

**Application to Data-augmented Algorithm Design.** One important motivation for  $\ell_p$  estimation with large approximation is worst-case guarantees for learning-augmented data stream algorithm design. In [32], it was shown that given a heavy hitter oracle which can decide, for each input  $i$ , whether or not  $|x_i| \geq n^{-p/2} \|x\|_p$ , one can estimate  $\|x\|_p$  up to

a constant factor with probability at least 0.9 using  $O(n^{1/2-1/p} \log n \log M)$  bits of space. In this case, we say the oracle is successful. However, when the oracle is not successful, there is no worst-case guarantee on the quality of approximation. An observation here is that when the oracle is not successful, the estimation will be an under-estimate with high probability. Letting  $\alpha = \Theta(n^{1/4-1/(2p)})$  in the preceding theorem, we obtain an  $\alpha$ -approximation algorithm whose output  $Z$  satisfies  $\frac{1}{\alpha} \|x\|_p \leq Z \leq \|x\|_p$  with probability at least 0.9 using the same  $O(n^{1/2-1/p} \log n \log M)$  bits of space. Hence we can run our algorithm and the oracle algorithm in parallel and take a maximum. This guarantees an  $\alpha$ -approximation in  $O(n^{1/2-1/p} \log n \log M)$  bits of space with probability at least 0.8.

► **Theorem 13.** *Assuming a successful oracle, there is a streaming algorithm which runs in  $O(n^{1/2-1/p} \log M \log n)$  bits of space, and for which the output  $Z$  satisfies  $\|x\|_p \leq Z \leq 2 \|x\|_p$ . Moreover, even if the oracle is not successful, the output  $Z$  always satisfies  $\|x\|_p \leq Z \leq n^{1/4-1/(2p)} \|x\|_p$ .*

**Lower Bound.** We next show an  $\Omega(n^{1-2/p}(\log(M) \log(1/\delta))/\alpha^2)$  lower bound for obtaining an  $\alpha$ -approximation to  $\|x\|_p$ , or, equivalently, an  $\Omega(n^{1-2/p}(\log(M) \log(1/\delta))/\alpha^{2/p})$  lower bound for obtaining an  $\alpha$ -approximation of  $F_p(x)$ . We first note that it is easy to get an  $\Omega(n^{1-2/p}/\alpha^{2/p})$  lower bound from the following  $\ell_\infty^k$  communication problem in [9]: there are two parties, Alice and Bob, holding vectors  $x, y \in \mathbb{Z}^n$  respectively, and their goal is to decide if  $\|x - y\|_\infty \leq 1$  or  $\|x - y\|_\infty \geq k$ . This problem requires  $\Omega(n/k^2)$  bits of communication [9]. Let  $k = 2^{1/p} \alpha^{1/p} n^{1/p}$ . For the case where  $\|x - y\|_\infty \leq 1$ , we have  $\|x - y\|_p^p \leq n$ . For the case where  $\|x - y\|_\infty \geq k$ , we have  $\|x - y\|_p^p \geq k^p = 2\alpha n$ . Suppose there is an algorithm  $\mathcal{A}$  which can output a number  $Z$  such that  $\|x\|_p \leq Z \leq \alpha \|x\|_p$  with probability at least  $2/3$ . Then Alice can perform the update  $x$  to the algorithm  $\mathcal{A}$  and send the memory contents of  $\mathcal{A}$  to Bob. Bob then performs the update  $-y$  to  $\mathcal{A}$ . From the discussion above, Bob can determine which of the two cases it is with probability at least  $2/3$ .

To obtain a stronger lower bound, we consider the following version of multiparty disjointness ( $\text{DIS}_s^n$ ), coupled with an input distribution, in the public-coin simultaneous message passing model of communication (SMP), as proposed in [51]. In this setting, there are  $s$  players, each of whom has a bit string  $\mathbf{X}_i \in \{0, 1\}^n$  ( $i \in [s]$ ) as input. The inputs are generated according to the following distribution  $\eta$ .

► **Definition 14** (Distribution  $\eta$ ). *The distribution  $\eta$  is the joint distribution of  $(\mathbf{X}_1, \dots, \mathbf{X}_s) \in (\{0, 1\}^n)^s$ , generated as follows.*

- For each  $i \in [n], j \in [s]$ , set  $\mathbf{X}_{j,i} \sim B(1/s)$  independently at random.
- Pick a uniformly random coordinate  $I \in [n]$ .
- Pick a  $Z \in \{0, 1\}$ . If  $Z = 1$ , set  $\mathbf{X}_{j,I} = 1$  for all  $j \in [s]$ . (If  $Z = 0$ , keep all coordinates as before.)

We call the instance of the inputs  $\{\mathbf{X}_i\}_{i \in [s]}$  a “YES” instance when  $Z = 1$ , and a “NO” instance when  $Z = 0$ .

The players simultaneously send a message  $M_i(\mathbf{X}_i, R)$  to a referee, where  $R$  denotes the public coins shared among the players. The referee then decides, based on  $M_1(\mathbf{X}_1, R), \dots, M_s(\mathbf{X}_s, R)$  and  $R$ , whether  $\{\mathbf{X}_i\}_{i \in [s]}$  forms a YES instance or a NO instance. As observed in [51], if  $X \sim \text{Bin}(s, 1/s)$ , then  $\Pr[X > \ell] \leq (e/\ell)^\ell$ . Hence, by a union bound for all coordinates  $i \in [n]$ , it holds in a NO instance, with probability at least  $1 - 1/\text{poly}(n)$ , that  $\sum_{j=1}^s \mathbf{X}_{j,i} \leq c \log n / (\log \log n)$  for all  $i \in [n]$ . On the other hand, in a YES instance it always holds that  $\sum_{j=1}^s \mathbf{X}_{j,I} = s$ . Thus, YES and NO instances are distinguishable for  $s = \Omega(\log n / \log \log n)$ .



The following is an augmented version of this problem.

► **Definition 15** (Aug-DISJ( $r, s, \delta$ )). *The augmented disjointness problem Aug-DISJ( $r, s, \delta$ ) is the following  $s$ -party communication problem. The players receive  $r$  instances of  $\text{DISJ}_s^n(\mathbf{X}_1^1, \dots, \mathbf{X}_s^1), (\mathbf{X}_1^2, \dots, \mathbf{X}_s^2), \dots, (\mathbf{X}_1^r, \dots, \mathbf{X}_s^r)$  and the referee, in addition, receives an index  $T \in [r]$  which is unknown to the players, along with the last  $(r - T)$  inputs  $\{(\mathbf{X}_1^t, \dots, \mathbf{X}_s^t)\}_{t=T+1}^r$ . The inputs are generated according to the following distribution:*

- (i)  $T$  is chosen uniformly at random from  $[r]$ ;
- (ii)  $(\mathbf{X}_1^T, \dots, \mathbf{X}_s^T) \sim \eta$ ;
- (iii) For each  $t \neq T$ ,  $(\mathbf{X}_1^t, \dots, \mathbf{X}_s^t) \sim \eta_0$  independently, where  $\eta_0$  is the conditional distribution of  $\eta$  given  $Z = 0$ .

At the end of the protocol, the referee should output whether the  $T$ -th instance  $(\mathbf{X}_1^T, \dots, \mathbf{X}_s^T)$  is a YES or a NO instance, i.e., the players need to solve  $\text{DISJ}_s^n(\mathbf{X}_1^T, \dots, \mathbf{X}_s^T)$ , with probability  $1 - \delta$ .

► **Theorem 16** ([51]). *Suppose that  $\delta \geq n \cdot 2^{-s}$ . Any deterministic protocol that solves Aug-DISJ( $r, s, \delta$ ) (as defined in Definition 15) requires  $\Omega(rn \min(\log \frac{1}{\delta}, \log s)/s)$  bits of total communication.*

*A Reduction to Streaming:* To lower bound the space complexity of a streaming algorithm we need a way of relating it to the communication cost of a protocol for this communication problem. In [51], the authors use a result of [40], showing under certain conditions that any streaming algorithm  $\mathcal{A}$  which solves the problem  $P$  with probability at least  $1 - \delta$  can be converted to a “path-independent” streaming algorithm  $\mathcal{B}$  which solves  $P$  with probability at least  $1 - 7\delta$ , and which uses the same space up to an additive  $(\log n + \log \log m + \log 1/\delta)$  factor. The latter then gives a protocol for the Aug-DISJ( $r, s, \delta$ ) problem. Here path-independence means that the output of the algorithm only depends on the initial state and the underlying frequency vector. In other words, the order of the updates of the same frequency vector will not cause different outputs to such an algorithm. We now assume that the algorithm  $\mathcal{A}$  we have enjoys this path-independence property. For a more detailed discussion, we refer the readers to Section 5 in [51].

Suppose there is a path-independent 1-pass streaming algorithm  $\mathcal{A}$  which gives an  $\alpha$ -approximation to  $\|x\|_p^p$  with probability  $1 - \delta$ . We shall use this to solve the Aug-DISJ( $r, s, \delta$ ) problem for  $s = \Theta(\alpha^{1/p} n^{1/p})$  and  $r = \log(M/s)$ , from which a space lower bound of  $\Omega(n^{1-2/p} \log(M) \log(1/\delta)/\alpha^{2/p})$  bits follows if  $M = \Omega((n\alpha)^{1/p+O(1)})$ .

We design the following protocol  $\pi$  between the players and the referee. For each  $i \in [s]$ , player  $i$  has  $r$  instances  $(\mathbf{X}_i^1, \mathbf{X}_i^2, \dots, \mathbf{X}_i^r)$ . Player  $i$  performs the update  $10^{j-1} \cdot \mathbf{X}_i^j$  to the algorithm  $\mathcal{A}$ , for each  $j \in [r]$ , and sends the memory contents of  $\mathcal{A}$  to the referee. Under the path-independence assumption, the referee can determine an equivalent frequency vector (i.e., leading to the same state of the algorithm) from each player and then add up the corresponding updates itself. After receiving  $T$  and  $\{(\mathbf{X}_1^t, \dots, \mathbf{X}_s^t)\}_{t=T+1}^r$ , the referee performs the update  $-10^{j-1} \cdot (\sum_{i=1}^s \mathbf{X}_i^j)$  to the algorithm  $\mathcal{A}$ , for each  $j \geq T+1$ . Suppose that  $\mathcal{A}$  outputs a set  $S$ . The referee will output YES if  $|S| = 1$  and NO if  $S = \emptyset$ .

Next we analyze correctness of the above protocol  $\pi$ . We recall that the referee needs to output the answer to the  $T$ -th instance. For simplicity, we define  $\mathbf{Y}^j = \sum_{i=1}^s \mathbf{X}_i^j$  for the  $j$ -th instance. After taking a union bound, for every instance  $j$ ,  $\|\mathbf{Y}^j\|_\infty \leq c \log n / \log \log n$  if it is a NO instance. Also from a Chernoff bound, it is easy to see that  $\|\mathbf{Y}^j\|_2^2 = \Omega(n)$  for all  $j$  with probability at least  $1 - e^{-\Omega(n)}$ . Note that the actual underlying vector that  $\mathcal{A}$  maintains has the same output as the frequency vector  $\mathbf{Y} = \sum_{t=1}^T 10^{t-1} \mathbf{Y}^t$  after the referee performs the updates. We need the following concentration bounds for  $\mathbf{Y}$  [51]. We note an omission in the proof in that paper and included a corrected one in Appendix A.

► **Lemma 17** ([51]). Let  $\sigma_r(\|\mathbf{Y}_{-I}\|_p^p) = (\mathbb{E}[\|\mathbf{Y}_{-I}\|_p^p - \mathbb{E}[\|\mathbf{Y}_{-I}\|_p^p]^r])^{1/r}$ . It holds that

$$\mathbb{E}[\|\mathbf{Y}_{-I}\|_p^p] \leq K_1^p p^p n \cdot 10^{pT}, \quad (1)$$

$$\sigma_r(\|\mathbf{Y}_{-I}\|_p^p) \leq K_2^p p^p \frac{r}{\ln r} \max\{2^p \sqrt{n}, r^p n^{1/r}\} \cdot 10^{pT}, \quad (2)$$

where  $r \geq 2$  is arbitrary and  $K_1, K_2 > 0$  are absolute constants.

Taking  $r = 3 \ln n$  in (2) gives that

$$\begin{aligned} & \Pr[|\|\mathbf{Y}_{-I}\|_p^p - \mathbb{E}[\|\mathbf{Y}_{-I}\|_p^p]| > 0.1n \cdot 10^{pT}] \\ & \leq \Pr[|\|\mathbf{Y}_{-I}\|_p^p - \mathbb{E}[\|\mathbf{Y}_{-I}\|_p^p]| > 2\sigma_{\ell}(\|\mathbf{Y}_{-I}\|_p^p)] \\ & \leq 2^{-r} \leq 1/n^2. \end{aligned} \quad (3)$$

We condition on all of the events above. Notice that in all cases, the value  $\|x\|_{\infty}$  of the underlying vector  $x$  the algorithm  $\mathcal{A}$  maintains is less than  $\left(\sum_{i=1}^{r-1} 10^{i-1} \cdot \frac{\log n}{\log \log n} + 10^{r-1} \cdot s\right) < 10^r \cdot s = O(M)$  for  $r = \log(M/\alpha^{1/p} n^{1/p})$ .

We first consider the case for which the  $T$ -th instance is a YES instance. In this case,  $\mathbf{Y}_I^T = s$  and thus,  $\|\mathbf{Y}\|_p^p \geq 10^{(T-1)p} \cdot s = \Omega(10^{(T-1)p} \cdot \alpha n)$ .

Next consider the case in which the  $T$ -th instance is a NO instance. In this case, we have from (1) and (3) that  $\|\mathbf{Y}\|_p^p = \|\mathbf{Y}_{-I}\|_p^p + \mathbf{Y}_I^p \leq K_p \cdot 10^{pT} n + 10^{pT} (\frac{\log n}{\log \log n})^p \leq 1.1K_p \cdot 10^{pT} n$ , where  $K_p$  is a constant that depends only on  $p$ .

From the same argument in Section 5 we know that if there is an algorithm that can output a  $Z$  such that  $\|x\|_p^p \leq Z \leq K'_p \alpha \|x\|_p^p$ , we can use this algorithm to solve the Aug-DISJ( $r, s, \delta$ ) problem. From Theorem 16, we obtain the following theorem.

► **Theorem 18.** Suppose that  $p$  is a constant and  $M = \Omega((\alpha n)^{1/p+O(1)})$ . Then, for  $\delta \geq 2^{-\Theta((n\alpha)^{1/p})}$ , any one-pass streaming algorithm which outputs a number  $Z$  for which  $\|x\|_p^p \leq Z \leq \alpha \|x\|_p^p$  with probability at least  $1 - \delta$  requires  $\Omega(n^{1-2/p} \log(M) \log(1/\delta)/\alpha^{2/p})$  bits of space. In particular, when  $\delta = \Theta(1/n)$ , any one-pass streaming algorithm requires  $\Omega(n^{1-2/p} \log(M) \log(n)/\alpha^{2/p})$  bits of space.

## 5 $\ell_2$ Heavy Hitters

In the heavy hitters problem, we want to find a set  $S \subseteq [n]$  of indices for the underlying vector  $x$  such that:

- (i)  $S$  contains every  $i$  such that  $|x_i|^2 \geq \frac{1}{k} \|x\|_2^2$ ;
- (ii)  $S$  does not contain any  $i$  such that  $|x_i|^2 < \frac{1}{2k} \|x\|_2^2$ .

We call  $S$  a  $(1/k)$ -heavy set of  $x$  if  $S$  satisfies the above conditions. Using the classical Count-Sketch, we can solve the above problem in  $O(k \log n \log M)$  bits of space with high probability.

► **Lemma 19.** There is a randomized one-pass streaming algorithm which can be implemented in  $O(k \log n \log M)$  bits of space such that with probability  $1 - 1/\text{poly}(n)$ , it can output a  $\frac{1}{k}$ -heavy set  $S$  of  $x$ .

In this section, we consider the following relaxation of the heavy hitters problem, where we want to find a set  $S$  of indices such that:

- (i)  $S$  contains every  $i$  such that  $|x_i|^2 \geq \frac{1}{k} \|x\|_2^2$ ;
- (ii)  $S$  does not contain any  $i$  such that  $|x_i|^2 < \frac{1}{\alpha k} \|x\|_2^2$ .



We call such a set  $S$  a  $(\frac{1}{k}, \alpha)$ -heavy set of  $S$ . Our result is negative, where we show that any one-pass streaming algorithm outputting a  $(\frac{1}{k}, \alpha)$ -heavy set of  $x$  with probability at least  $1 - 1/n$  still requires  $\Omega(k \log n \log M)$  bits of space if  $\alpha = O((n/k)(\log \log n)^2/(\log n)^2)$ .

We  
tion 15.

Suppose that there is a path-independent one-pass streaming algorithm  $\mathcal{A}$  which can solve the  $(\frac{1}{k}, \alpha)$ -heavy hitters problem with probability  $1 - O(1/n)$ , where  $\alpha = O(n/k \cdot (\log \log n / \log n)^2)$ . Then we can use it to solve the  $\text{Aug-DISJ}(r, s, \delta)$  problem for  $r = \log(M/n^{1/2})$ ,  $s = \Theta(\sqrt{n/k})$ ,  $\delta = 1/n$ , from which a space lower bound of  $\Omega(k \log n \log M)$  bits follows if  $M = \Omega(n^{1/2+O(1)})$ .

We design the following protocol  $\pi$  between the players and referee. For each  $i \in [s]$ , player  $i$  has the  $r$  instances  $(\mathbf{X}_i^1, \mathbf{X}_i^2, \dots, \mathbf{X}_i^r)$ . Player  $i$  then performs the update  $10^{j-1} \cdot \mathbf{X}_i^j$  to the algorithm  $\mathcal{A}$  for each  $j \in [r]$  and sends the memory of  $\mathcal{A}$  to the referee. Under the path-independence assumption, the referee can determine an equivalent frequency vector (i.e., leading to the same state of the algorithm) from each player and then add up the corresponding updates. After receiving  $T$  and  $\{(\mathbf{X}_1^t, \dots, \mathbf{X}_s^t)\}_{t=T+1}^r$ , the referee performs the update  $-10^{j-1} \cdot (\sum_{i=1}^s \mathbf{X}_i^j)$  to the algorithm  $\mathcal{A}$  for each  $j \geq T+1$ . Suppose that  $\mathcal{A}$  outputs a set  $S$ . The referee will output YES if  $|S| = 1$  and NO if  $S = \emptyset$ .

Now we analyze the correctness of the above protocol  $\pi$ . We recall that the referee needs to output the answer to the  $T$ -th instance. For simplicity, we define  $\mathbf{Y}^j = \sum_{i=1}^s \mathbf{X}_i^j$  for the  $j$ -th instance. Recall that after taking a union bound, for every instance  $j$ ,  $\|\mathbf{Y}^j\|_\infty \leq c \log n / \log \log n$  if it is a NO instance. Also from a Chernoff bound, it is easy to see that  $\|\mathbf{Y}^j\|_2^2 = \Omega(n)$  for all  $j$  with probability at least  $1 - e^{-\Omega(n)}$ . Note that the actual underlying vector that algorithm  $\mathcal{A}$  maintains has the same output as the frequency vector  $\mathbf{Y} = \sum_{t=1}^T 10^{t-1} \mathbf{Y}^t$  after the referee performs the updates. We need the following concentration bounds, which are a special case of Lemma 17 with  $p = 2$ .

► **Lemma 20** ([51], special case of Claim 6.2). *It holds that*

$$\mathbb{E} \left[ \|\mathbf{Y}_{-I}\|_2^2 \right] \leq K_1 n \cdot 10^{2T}, \quad (4)$$

$$\sigma_\ell \left( \|\mathbf{Y}_{-I}\|_2^2 \right) \leq K_2 \frac{\ell}{\ln \ell} \max\{4\sqrt{n}, \ell^2 n^{1/\ell}\} \cdot 10^{2T}, \quad \forall \ell \geq 2, \quad (5)$$

where  $K_1, K_2 > 0$  are absolute constants.

Taking  $\ell = 3 \ln n$  in (5) gives that

$$\begin{aligned} & \Pr \left[ \left| \|\mathbf{Y}_{-I}\|_2^2 - \mathbb{E} \left[ \|\mathbf{Y}_{-I}\|_2^2 \right] \right| > 0.1n \cdot 10^{2T} \right] \\ & \leq \Pr \left[ \left| \|\mathbf{Y}_{-I}\|_2^2 - \mathbb{E} \left[ \|\mathbf{Y}_{-I}\|_2^2 \right] \right| > 2\sigma_\ell \left( \|\mathbf{Y}_{-I}\|_2^2 \right) \right] \\ & \leq 2^{-\ell} \leq 1/n^2. \end{aligned} \quad (6)$$

Condition on all of the events above occurring. In all cases, the value  $\|x\|_\infty$  of the underlying vector  $x$  that algorithm  $\mathcal{A}$  maintains is less than  $\left( \sum_{i=1}^{r-1} 10^{i-1} \cdot \frac{\log n}{\log \log n} + 10^{r-1} \cdot \sqrt{n/k} \right) < 10^r \cdot \sqrt{n} = O(M)$  for  $r = \log(M/n^{1/2})$ .

We first consider the case in which the  $T$ -th instance is a YES instance. In this case,  $\mathbf{Y}_I^T = s$ , and thus

$$\mathbf{Y}_I \geq 10^{T-1} \cdot s.$$

## 13:14 Streaming Algorithms with Large Approximation Factors

Meanwhile, for all  $j \neq I$ ,

$$\mathbf{Y}_j \leq c \sum_{t=1}^T 10^{t-1} \frac{\log n}{\log \log n} < c \cdot 10^T \frac{\log n}{\log \log n}. \quad (7)$$

It follows from (4) and (6) that

$$\Omega(10^{2T} \cdot n) \leq \|\mathbf{Y}\|_2^2 = \|\mathbf{Y}_{-I}\|_2^2 + \mathbf{Y}_I^2 \leq (K_1 + 0.1)n \cdot 10^{2T} + s^2.$$

It thus holds that  $\mathbf{Y}_I^2 \geq (1/k) \|\mathbf{Y}\|_2^2$ , or equivalently,  $s^2/100 \geq s^2/k + (K_1 + 0.1)n/k$  when  $k > 100$  and  $s = \Omega(\sqrt{n/k})$ . Furthermore, for  $j \neq I$ ,  $\mathbf{Y}_j^2 \leq \|\mathbf{Y}\|_2^2/(\alpha k)$  when  $\alpha = O((n/k)(\log \log n / \log n)^2)$ . Therefore, our choices of  $k$ ,  $s$  and  $\alpha$  imply that the set  $S = \{I\}$ .

Now we consider the case when the  $T$ -th instance is a NO instance. In this case, (7) holds for all  $j \in [n]$ . Since  $\|\mathbf{Y}\|_2^2 \geq \Omega(10^{2T} \cdot n)$ , it follows that  $\mathbf{Y}_j^2 \leq \|\mathbf{Y}\|_2^2/(\alpha k)$  for all  $j$ , provided that  $\alpha = O((n/k)(\log \log n / \log n)^2)$ . It follows that  $S = \emptyset$ .

To conclude, we have proved the following theorem.

► **Theorem 21.** *Suppose that  $k = \Omega(1)$ ,  $\alpha = O(\frac{n}{k}(\frac{\log \log n}{\log n})^2)$  and  $M = \Omega(n^{1/2+O(1)})$ . Then, any one-pass streaming algorithm that solves the  $(1/k, \alpha)$ -heavy hitters problem with failure probability  $O(1/n)$  requires  $\Omega(k \log n \log M)$  bits of space, where the algorithm can store any number of random bits.*

**Sketching dimension lower bound.** One limitation of the above theorem is that it requires the algorithm  $\mathcal{A}$  to succeed with high probability. Below we show that any algorithm  $\mathcal{A}$  using a linear sketch to solve the  $(1/k, \alpha)$ -heavy hitters problem with constant probability requires the sketching dimension to be  $O(k \log(n/k))$  if  $\alpha = O(n/(k \log n))$ .

We will consider the following communication game in [45]. Let  $\mathcal{F} \subset \{S \subset [n] \mid |S| = k/2\}$  be a family of  $k$ -sparse supports such that:

- $|S \Delta S'| \geq k$  for  $S \neq S' \in \mathcal{F}$ ,
- $\Pr_{S \in \mathcal{F}}[i \in S] = k/(2n)$  for all  $i \in [n]$ , and
- $\log |\mathcal{F}| = \Omega(k \log(n/k))$ .

Let  $X = \{x \in \{0, \pm 2\sqrt{n/k}\}^n \mid \text{supp}(x) \in \mathcal{F}\}$ . Let  $w \sim \mathcal{N}(0, I_n)$ . Consider the following process. First, Alice chooses  $S \in \mathcal{F}$  uniformly at random. Then  $x \in X$  is uniformly at random subject to  $\text{supp}(x) = S$ , and then  $w \sim \mathcal{N}(0, I_n)$ . Then, Alice computes  $y = Az = A(x + w)$ , where  $A \in \mathbb{R}^{m \times n}$  is the sketching matrix in  $\mathcal{A}$ , and Alice sends  $y$  to Bob. Then Bob needs to recover  $S$  from  $y$ .

► **Theorem 22 ([45]).** *Suppose that Bob can recover  $S$  with probability at least  $2/3$ . Then  $m = \Omega(k \log(n/k))$ .*

Next we will show that Alice and Bob can use a  $(\frac{1}{k}, \alpha)$ -heavy hitters algorithm to solve the communication game above if  $\alpha = O(n/(k \log n))$ . To show correctness, we need the following bounds for  $w$ .

► **Lemma 23 (folklore).** *Suppose that  $w \sim \mathcal{N}(0, I_n)$ . Then with probability  $9/10$  we have the following:*

- (i)  $0.9n \leq \|w\|_2^2 \leq 1.1n$ ;
- (ii)  $\|w\|_\infty \leq c \cdot \sqrt{\log n}$ .

Condition on the events above. For each  $i \in [n]$ , we have

$$\begin{aligned} z_i &\geq 2\sqrt{n/k} - c\sqrt{\log n} \geq 1.9\sqrt{n/k}, & i \in S, \\ z_i &\leq c\sqrt{\log n}, & i \notin S. \end{aligned}$$

We also have from Lemma 23 that

$$0.9n < \|w\|_2^2 < \|z\|_2^2 \leq \|w\|_2^2 + \|x\|_2^2 + 4ck\sqrt{\log n}\sqrt{n/k} < 4n.$$

It follows that any  $(\frac{1}{k}, \alpha)$ -heavy set  $T$  will exactly be the support set  $S$  if  $\alpha = O(n/(k \log n))$ . The following theorem is immediate.

► **Theorem 24.** *Suppose that  $\alpha = O(n/(k \log n))$ . Then, any linear sketching algorithm that solves the  $(1/k, \alpha)$ -heavy hitters problem with constant probability requires a sketching dimension of  $\Omega(k \log(n/k))$ .*

## 6 Additional Results

We present improved one-pass and multipass algorithms for the  $\ell_0$  estimation problem in Section B.1, our two-pass algorithm for the  $F_p$  estimation ( $0 < p \leq 2$ ) problem in Section C, our results for Schatten- $p$  norm estimation in Section D, and finally, our results for cascaded norms and rectangle-efficient  $F_p$  estimation in Section E and Section F, respectively. All proofs are omitted and can be found in the full version of this paper.

---

## References

- 1 Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. The aqua approximate query answering system. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 574–576, June 1–3, 1999, Philadelphia, Pennsylvania, USA, 1999. ACM Press.
- 2 Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.
- 3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.
- 4 Alexandr Andoni. High frequency moments via max-stability. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5–9, 2017*, pages 6364–6368. IEEE, 2017. doi:10.1109/ICASSP.2017.7953381.
- 5 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David P. Woodruff. Efficient sketches for earth-mover distance, with applications. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25–27, 2009, Atlanta, Georgia, USA*, pages 324–330. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.25.
- 6 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 363–372. IEEE, 2011.
- 7 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742. SIAM, 2017.

- 8 Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1190–1197. SIAM, 2010. doi:10.1137/1.9781611973075.95.
- 9 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.
- 10 Ziv Bar-Yossef, Thathachar S Jayram, Robert Krauthgamer, and Ravi Kumar. The sketching complexity of pattern matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 261–272. Springer, 2004.
- 11 Kevin S. Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and iceberg CUBEs. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 359–370, 1999.
- 12 Mark Braverman, Sumegha Garg, and David P Woodruff. The coin problem with applications to data streams. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 318–329. IEEE, 2020.
- 13 Mark Braverman, Sumegha Garg, and Or Zamir. Tight space complexity of the coin problem. *Electron. Colloquium Comput. Complex.*, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/083>.
- 14 Vladimir Braverman, Moses Charikar, William Kuszmaul, David P. Woodruff, and Lin F. Yang. The one-way communication complexity of dynamic time warping distance. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, USA*, volume 129 of *LIPIcs*, pages 16:1–16:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 15 Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, Yi Li, David P. Woodruff, and Lin F. Yang. Matrix norms in data streams: Faster, multi-pass and row-order. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 648–657, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, 2018. PMLR.
- 16 Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Roi Sinoff. Schatten norms in matrix streams: Hello sparsity, goodbye dimension. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 1100–1110, Virtual Event, 2020. PMLR.
- 17 Paul Brown, Peter Haas, Jussi Myllymaki, Hamid Pirahesh, Berthold Reinwald, and Yannis Sismanis. Toward automated large-scale information integration and discovery. In *Data Management in a Connected World*, pages 161–180. Springer, 2005.
- 18 Amit Chakrabarti and Sagar Kale. Strong fooling sets for multi-player communication with applications to deterministic estimation of stream statistics. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 41–50. IEEE, 2016.
- 19 Graham Cormode and S Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 271–282, 2005.
- 20 Tamraparni Dasu, Theodore Johnson, Shanmugaelayut Muthukrishnan, and Vladislav Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 240–251, 2002.
- 21 David J DeWitt, Jeffrey F Naughton, Donovan A Schneider, and Srinivasan Seshadri. Practical skew handling in parallel joins. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1992.
- 22 Elbert Du, Michael Mitzenmacher, David P. Woodruff, and Guang Yang. Separating k-Player from t-Player One-Way Communication, with Applications to Data Streams. *arXiv e-prints*, page arXiv:1905.07135, May 2019. arXiv:1905.07135.

- 23 Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing iceberg queries efficiently. In *Proc. 24rd International Conference on Very Large Data Bases*, pages 299–310, 1998.
- 24 Schkolnick Finkelstein, Mario Schkolnick, and Paolo Tiberio. Physical database design for relational databases. *ACM Transactions on Database Systems (TODS)*, 13(1):91–128, 1988.
- 25 Sumit Ganguly and David P. Woodruff. High probability frequency moment sketches. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9–13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 58:1–58:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 26 Jiawei Han, Jian Pei, Guozhu Dong, and Ke Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. 2001 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2001.
- 27 Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.
- 28 Christian Hidber. Online association rule mining. In *SIGMOD 1999, Proc. ACM SIGMOD International Conference on Management of Data*, pages 145–156, 1999.
- 29 Piotr Indyk and Ali Vakilian. Tight trade-offs for the maximum k-coverage problem in the general streaming model. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 200–217. ACM, 2019.
- 30 Rajesh Jayaram and David P. Woodruff. Data streams with bounded deletions. *CoRR*, abs/1803.08777, 2018. [arXiv:1803.08777](https://arxiv.org/abs/1803.08777).
- 31 T. S. Jayram and David P. Woodruff. The data stream space complexity of cascaded norms. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25–27, 2009, Atlanta, Georgia, USA*, pages 765–774. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.82.
- 32 Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. In *International Conference on Learning Representations*, 2019.
- 33 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for  $L_p$  samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12–16, 2011, Athens, Greece*, pages 49–58. ACM, 2011. doi:10.1145/1989284.1989289.
- 34 John Kallaugher and Eric Price. Separations and equivalences between turnstile streaming and linear sketching. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22–26, 2020*, pages 1223–1236. ACM, 2020.
- 35 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17–19, 2010*, pages 1161–1178. SIAM, 2010.
- 36 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In Jan Paredaens and Dirk Van Gucht, editors, *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6–11, 2010, Indianapolis, Indiana, USA*, pages 41–52. ACM, 2010. doi:10.1145/1807085.1807094.
- 37 Aditya Krishnan, Sidhanth Mohanty, and David P. Woodruff. On sketching the q to p norms. *CoRR*, abs/1806.06429, 2018. [arXiv:1806.06429](https://arxiv.org/abs/1806.06429).

- 38 Rafał Łatała. Estimation of moments of sums of independent real random variables. *The Annals of Probability*, 25(3):1502–1513, 1997. doi:10.1214/aop/1024404522.
- 39 Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, 1991.
- 40 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 174–183. ACM, 2014.
- 41 Yi Li, Huy L. Nguyen, and David P. Woodruff. On approximating matrix norms in data streams. *SIAM J. Comput.*, 48(6):1643–1697, 2019. doi:10.1137/17M1152255.
- 42 Yi Li and David P. Woodruff. Tight bounds for sketching the operator norm, Schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- 43 Yi Li and David P. Woodruff. Embeddings of Schatten norms with applications to data streams. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proceedings of ICALP*, volume 80 of *LIPICs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ICALP.2017.60.
- 44 Sriram Padmanabhan, Bishwaranjan Bhattacharjee, Tim Malkemus, Leslie Cranston, and Matthew Huras. Multi-dimensional clustering: A new data layout scheme in db2. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 637–641, 2003.
- 45 Eric Price and David P. Woodruff.  $(1 + \epsilon)$ -approximate sparse recovery. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 295–304. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.92.
- 46 Ashok Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. 21th International Conference on Very Large Data Bases*, pages 432–444, 1995.
- 47 P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. Access path selection in a relational database management system. In *Readings in Artificial Intelligence and Databases*, pages 511–522. Elsevier, 1989.
- 48 Amit Shukla, Prasad Deshpande, Jeffrey F Naughton, and Karthikeyan Ramasamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *VLDB*, volume 96, pages 522–531. Citeseer, 1996.
- 49 Srikanta Tirthapura and David Woodruff. Rectangle-efficient aggregation in spatial data streams. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 283–294. ACM, 2012.
- 50 Hannu Toivonen. Sampling large databases for association rules. In *Proc. 22th International Conference on Very Large Data Bases*, pages 134–145, 1996.
- 51 Omri Weinstein and David P. Woodruff. The simultaneous communication of disjointness with applications to data streams. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 1082–1093. Springer, 2015.
- 52 David P. Woodruff and Guang Yang. Separating  $k$ -player from  $t$ -player one-way communication, with applications to data streams. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPICs*, pages 97:1–97:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.



## A Proof of Lemma 17

The first result, Equation (1), was proved in [51]. Now we prove the second result.

By a standard symmetrization technique (see, e.g., [39, p153]),

$$\left(\mathbb{E} \left\| \mathbf{Y}_{-I} \right\|_p^p - \mathbb{E} \left[ \left\| \mathbf{Y}_{-I} \right\|_p^p \right] \right)^{\frac{1}{r}} = \left( \mathbb{E} \left| \sum_{i \neq I} (\mathbf{Y}_i^p - \mathbb{E} \mathbf{Y}_i^p) \right|^r \right)^{\frac{1}{r}} \leq 2 \left( \mathbb{E} \left| \sum_{i \neq I} \varepsilon_i \mathbf{Y}_i^p \right|^r \right)^{\frac{1}{r}}, \quad (8)$$

multiline where the  $\varepsilon_i$  are independent Rademacher variables.

By Latała's inequality ([38, Corollary 3]), it holds for  $r \geq 2$  that

$$\left( \mathbb{E} \left| \sum_{i \neq I} \varepsilon_i \mathbf{Y}_i^p \right|^r \right)^{\frac{1}{r}} \leq K_1 \frac{r}{\ln r} \max \left\{ \left( \mathbb{E} \sum_{i \neq I} \mathbf{Y}_i^{2p} \right)^{\frac{1}{2}}, \left( \mathbb{E} \sum_{i \neq I} \mathbf{Y}_i^{rp} \right)^{\frac{1}{r}} \right\}, \quad (9)$$

where  $K_1 > 0$  is an absolute constant.

It was shown in [51, Lemma 6.3] that

$$\mathbb{E} \mathbf{Y}_i^p \leq K_2^p p^p 10^{Tp}, \quad p \geq 1,$$

for some absolute constant  $K_2 > 0$ . It then follows that

$$\left( \mathbb{E} \sum_{i \neq I} \mathbf{Y}_i^{rp} \right)^{\frac{1}{r}} \leq K_2^p (rp)^p n^{1/r} 10^{pT} \quad (10)$$

The result follows from combining (8), (9) and (10).

*Remark.* We note an omission in [51]. In that paper, the proof of the second result, i.e., Equation (5), assumes that the larger term in (9) is  $(\mathbb{E} \sum_{i \neq I} \mathbf{Y}_i^{2r})^{1/2}$ , which is not necessarily the case. Lemma 2.5 in that paper is also an incorrect citation from [38], since the conclusion should be  $\max\{\Delta_1(X), \Delta_\ell(X)\}$  for nonnegative variables  $X$ , but this would be too large for the proof. Hence we first symmetrize the variables, which allows for a better bound on  $\max\{\Delta_2(X), \Delta_\ell(X)\}$ .

## B $\ell_0$ Estimation

### B.1 One-pass Algorithm

We describe a randomized algorithm which gives an  $n^{1/t}$ -approximation with constant probability using  $O(t \log \log M)$  bits of space, excluding the storage for random bits. We assume that  $n^{1/t} \geq c_2$  for some constant  $c_2$ , otherwise an optimal algorithm is known [35].

The algorithm is presented in Algorithm 1. The idea behind the algorithm is to subsample the coordinates at  $t$  levels, with a geometrically decreasing sampling probability. In each level, the surviving coordinates are hashed into a constant number of buckets. If the  $\ell_0$  of the subvector (which is the vector of surviving coordinates) at a level is at most a constant, then only a small number of these buckets will be occupied. Otherwise, all the buckets will be occupied with high probability. Based on this, we design a criterion to determine the occupancy of these buckets to infer the  $\ell_0$  of the subvector at a level. Finally, we find the specific level  $J$  such that the  $\ell_0$  in level  $J$  is between 0 and at most  $n^{1/t}$ , and then it can be shown that  $n^{J/t}$  is a good estimator to the overall  $\ell_0$ .

■ **Algorithm 1**  $n^{1/t}$ -approximator for  $\ell_0$ .

---

```

1 Initialize cKt counters $C_{1,1,1}, \dots, C_{t,K,c}$ to 0;
2 $c_1 \leftarrow \beta\sqrt{c}$;
3 Initialize pairwise independent hash functions $h : [n] \rightarrow [n]$, $g : [n] \rightarrow [c]$;
4 Initialize K 4-wise independent hash functions $s_i : [n] \rightarrow \{-1, 1\}$;
5 Pick a prime $p \in \Theta(c^3 \log^2 M)$;
6 foreach (x, v) in the data stream do
7 $b \leftarrow$ the largest j such that $h(x) \bmod \lfloor n^{1/t} \rfloor^j = 0$;
8 for $i \leftarrow 1$ to b do
9 for $j \leftarrow 1$ to K do
10 $C_{i,j,g(x)} \leftarrow (C_{i,j,g(x)} + v \cdot s_j(x)) \bmod p$;
11 end
12 end
13 end
14 if there exists j such that $|\{k \mid \exists l, C_{j,l,k} \neq 0\}| > c_1$ then
15 $J \leftarrow$ the largest j such that $|\{k \mid \exists l, C_{j,l,k} \neq 0\}| > c_1$;
16 else
17 $J \leftarrow 0$;
18 end
19 return $c_2 n^{J/t}$;

```

---

► **Theorem 25.** *Algorithm 1 outputs  $Z$ , which with probability at least 0.9 satisfies that  $\ell_0/n^{1/t} \leq Z \leq L_0 n^{1/t}$ . Furthermore, Algorithm 1 uses  $O(t \log \log M)$  bits of space, excluding its random tape.*

► **Remark 26.** Algorithm 1 uses  $O(\log n)$  random bits since the hash functions  $h, g$  are pairwise independent and the  $s_i$  are 4-wise independent.

## B.2 Lower Bound

We now prove a space lower bound of  $\Omega(t)$  bits for estimating  $\ell_0$  up to an  $n^{1/t}$ -approximation factor. Our lower bound holds even if the algorithm has access to an arbitrarily long random tape, which we do not charge for in its space. We reduce the  $\ell_0$  estimation problem to the Augmented Indexing communication problem, in the one-way public coin model, which we now define. We assume that  $t = O(\log n)$ .

► **Definition 27 (Augmented Indexing).** *Alice has a string  $u \in \{0, 1\}^l$ , Bob has an index  $i^* \in [l]$  and  $u_{i^*+1}, \dots, u_l$ . Alice is allowed to send a single message to Bob, and Bob wants to learn  $u_{i^*}$  from Alice with probability at least  $2/3$ .*

► **Lemma 28 ([10]).** *The one-way communication complexity of Augmented Indexing is  $\Omega(l)$  in the public coin model.*

Assume we have a streaming algorithm  $\mathcal{A}$ . Alice runs  $\mathcal{A}$  on her stream  $s(a)$ , then sends the state of  $\mathcal{A}$  to Bob. Bob feeds his stream  $s(b)$  into  $\mathcal{A}$  and obtains an estimate of  $\ell_0$ . We show how to design  $s(a)$  and  $s(b)$  so that Bob can solve the Augmented Indexing problem.

Without loss of generality, we assume that  $t$  is divisible by 8. Let  $u$  be the vector in an instance of the Augmented Indexing problem with  $l = t/8$ . We shall create an input vector  $x$  for the  $\ell_0$  estimation problem.



► **Theorem 29.** *Estimating  $\ell_0$  with approximation factor  $n^{1/t}$  requires  $\Omega(t)$  bits, even if the algorithm has an arbitrarily long random tape.*

### B.3 Multi-pass $\ell_0$ Estimation

We state our results for two-pass and three-pass algorithms below.

► **Theorem 30.** *There exists an absolute constant  $\varepsilon_0$  and a two-pass algorithm such that the following holds. For all  $\varepsilon \in (0, \varepsilon_0)$ , the algorithm outputs  $Z$  satisfying  $(1-\varepsilon)\ell_0 \leq Z \leq (1+\varepsilon)\ell_0$  with probability at least 0.8. The algorithm uses  $O(\log n + \varepsilon^{-2} \log \log M(\log(1/\varepsilon) + \log \log M))$  bits of space.*

► **Theorem 31.** *There exists an absolute constant  $\varepsilon_0$  and a three-pass algorithm such that the following holds. For all  $\varepsilon \in (0, \varepsilon_0)$ , the algorithm outputs  $Z$  satisfying  $(1-\varepsilon)\ell_0 \leq Z \leq (1+\varepsilon)\ell_0$  with probability at least 0.75. Furthermore, the algorithm uses  $O(\log n + \varepsilon^{-2}(\log(1/\varepsilon) + \log \log M))$  bits of space.*

### C Two-Pass Algorithm for $F_p$ ( $0 < p \leq 2$ )

As we have shown in the previous section, for the  $\|x\|_p^p$  estimation problem, even a large approximation also requires  $\Omega(\log n)$  bits of space. In this section, we will show that after obtaining a constant approximation to  $\|x\|_p^p$  in the first pass using  $O(\log n)$  bits, we can obtain a  $(1 \pm \varepsilon)$ -approximation to  $\|x\|_p^p$  using  $O(\log n + \varepsilon^{-2}(\log M + \log \frac{1}{\varepsilon}))$  bits. This is better than the previous  $O(\varepsilon^{-2} \log n M)$  space bound in one-pass if  $M$  is small.

► **Theorem 32.** *Suppose that  $0 < p \leq 2$ . There is a two-pass streaming algorithm which can be implemented in  $O(\log n + \varepsilon^{-2}(\log M + \log \frac{1}{\varepsilon}))$  bits of space and which outputs a  $(1 \pm \varepsilon)$ -approximation to  $\|x\|_p^p$  with probability at least  $9/10$ .*

### D Schatten- $p$ Norm Estimation

In this section, we consider approximating the Schatten- $p$  norm  $\|A\|_p$  of a given matrix  $A$  with large approximation factor  $\alpha$ , where  $\sigma_i(A)$  is the  $i$ -th singular value of  $A$ . We assume  $A \in \mathbb{R}^{n \times n}$  here because for a general matrix  $A \in \mathbb{R}^{n \times d}$ , we can first apply a subspace embedding to the left or to the right of  $A$  to preserve each of its singular values up to a constant factor and then pad with zero rows or columns (see, e.g., Appendix C of [40] for the details of this argument). As in the majority of previous work on Schatten norm estimation, we focus on the sketching dimension complexity.

**Upper Bound.** We will show that for an even integer  $p$  and an arbitrary  $\alpha = \Omega(1)$ , there is an  $O(n^{2-4/p}/\alpha^4)$  dimension sketching algorithm, while for  $p$  not an even integer, the  $O(n^{2-4/p}/\alpha^4)$  dimension bound still holds if  $\alpha$  is not too small. Our algorithm is based on a constant approximation algorithm for  $\|A\|_p$  when  $p$  is an even integer.

► **Lemma 33** (Theorem 8.2, [41]). *Suppose that  $p$  is an even integer. There is a sketching algorithm whose output  $Z$  satisfies  $\|A\|_p \leq Z \leq 2\|A\|_p$  with probability at least  $2/3$ . Furthermore, the sketching dimension of this algorithm is  $O(n^{2-4/p})$ .*

Our algorithm is given in Algorithm 2. For an even integer  $p$ , we maintain the matrix  $GAH^T$  where  $G$  and  $H$  are defined in algorithm 2 and use the constant approximation algorithm  $\mathcal{A}_q$  to estimate the Schatten- $q$  norm of  $GAH^T$ . The following lemma shows that  $\|GA\|_q$  can be an  $\alpha$ -approximation to  $\|A\|_p$ .

■ **Algorithm 2**  $\alpha$ -approximation for  $\|A\|_p$ .

---

```

1 Set $q = p$ if $p \in 2\mathbb{Z}$, or to be the largest even integer less than p otherwise;
2 Let \mathcal{A}_q be a streaming algorithm that can output a constant-factor approximation to
 $\|A\|_q$;
3 Set $t = (n^{1/2-1/p}/\alpha)^{1/(1/2-1/q)}$;
4 Let G be an $r = t \text{ poly}(\log(n/t)) \times n$ matrix with i.i.d $\mathcal{N}(0, 1/r)$ entries (these are i.i.d.
 normal random variables with mean 0 and variance $1/r$) and let H be an
 independent $r = O(t) \times n$ matrix with i.i.d $\mathcal{N}(0, 1/r)$ entries.
5 foreach $\Delta_{i,j}$ in the data stream do
6 | Compute the matrix $G\Delta_{i,j}H^T$;
7 | Add $G\Delta_{i,j}H^T$ to the input stream for \mathcal{A}_q ;
8 end
9 Let Z be the output from \mathcal{A}_q ;
10 return Z ;

```

---

► **Lemma 34** (rewording of Theorem 22, [43]). *Suppose that  $p \geq q \geq 2$ ,  $q$  is an even integer, and  $t = O(n)$ . Let  $G$  be an  $r \times n$  matrix with i.i.d.  $\mathcal{N}(0, 1/r)$  entries, where  $r = O(t)$  when  $q = 2$  and  $r = O(t \log^{1/(1/2-1/q)}(n/t))$  when  $q \geq 4$ . Then, with probability at least  $1 - \exp(-c't)$ , we have  $\|A\|_p \leq \|\gamma GA\|_q \leq (n^{1/2-1/p})/(t^{1/2-1/q}) \|A\|_p$ , where  $\gamma$  is an appropriate scaling factor.*

If  $H$  is a  $(1/2)$ -subspace embedding of  $GA$ , then the singular values of  $GAH^T$  are different from those of  $GA$  by at most a constant factor (see Section 2), and thus  $\|GAH^T\|_q$  is a constant approximation to  $\|GA\|_q$ . Recall that our sketch is a matrix of dimensions  $r \times O(t)$ , where  $r = t \text{ poly}(\log t)$ , so the sketching dimension of our algorithm is  $\tilde{O}(t^{2-4/p}) = \tilde{O}(n^{2-4/p}/\alpha^4)$ .

► **Theorem 35.** *Suppose that  $p \geq 2$  is an even integer. Then there is a sketching algorithm whose output  $Z$  satisfies  $\|A\|_p \leq Z \leq \alpha \|A\|_p$  with probability at least  $2/3$ . Furthermore, the sketching dimension of this algorithm is  $\tilde{O}(n^{2-4/p}/\alpha^4)$ .*

When  $p$  is not an even integer (and could even be a non-integer), let  $q$  be the largest even integer that is smaller than  $p$ . Then our choice of  $t$  still satisfies that  $t = O(n)$  if  $\alpha = \Omega(n^{1/p-1/q})$ . Our arguments above continue to hold and we obtain the following theorem.

► **Theorem 36.** *Suppose that  $p \geq 2$  is not an even integer. Let  $q$  be the largest even integer less than  $p$  and  $\alpha = \Omega(n^{1/q-1/p})$ . Then there is a sketching algorithm whose output  $Z$  satisfies  $\|A\|_p \leq Z \leq \alpha \|A\|_p$  with probability at least  $2/3$ . Furthermore, the sketching dimension of this algorithm is  $\tilde{O}(n^{2-4/p}/\alpha^4)$ .*

**Lower Bound.** Below we show that our upper bound is optimal up to  $\text{polylog}(n)$  factors. In [41], the authors give the following  $n^2/\alpha^4$  lower bound for  $\alpha$ -approximating  $\|A\|_{\text{op}}$ .

► **Lemma 37** (Corollary 3.3, [41]). *Suppose that  $\alpha \geq 1 + c$  where  $c$  is an arbitrarily small constant. Then, any sketching algorithm estimating  $\|A\|_{\text{op}}$  within a factor  $\alpha$  with failure probability smaller than  $1/6$  requires sketching dimension  $n^2/\alpha^4$ .*

Since  $\|x\|_\infty \leq \|x\|_p \leq n^{1/p} \|x\|_\infty$ , an  $\alpha$ -approximation of  $\|A\|_p$  implies an  $\alpha n^{1/p}$  approximation to  $\|A\|_\infty = \sigma_1(A)$ . The following lower bound follows.

► **Theorem 38.** Suppose that  $\alpha \geq 1 + c$ , where  $c > 0$  is an arbitrarily small constant. Then, any sketching algorithm estimating  $\|A\|_p$  within a factor  $\alpha$  with failure probability smaller than  $1/6$  requires sketching dimension  $O(n^{2-4/p}/\alpha^4)$ .

## E Cascaded Norms

In this section, we consider approximating the cascaded  $(p, q)$ -norm of a matrix  $X$ , defined as  $\|X\|_{p,q} = (\sum_i (\sum_j |x_{ij}|^q)^{p/q})^{1/p}$ , for a large approximation factor  $\alpha$ , when  $p \geq 1$  and  $q > 2$ . We have the following upper bound and show it is tight up to  $\text{poly}(\log n)$  factors.

► **Theorem 39.** Suppose that  $\alpha \geq 8$ . Then there is an algorithm whose output is  $Z$ , which satisfies that  $\|X\|_{p,q} \leq Z \leq \alpha \|X\|_{p,q}$  with probability at least  $2/3$ . Furthermore, the algorithm uses  $O(n^{1-2/p} d^{1-2/q} \cdot (pq \log n)^{O(1)}/\alpha^2)$  bits of space when  $p, q > 2$  and uses  $O(d^{1-2/q} \cdot (q \log n)^{O(1)}/\alpha^2)$  bits of space when  $1 \leq p < 2$  and  $q > 2$ .

► **Theorem 40.** For the case that  $p, q > 2$ , any one-pass streaming algorithm which outputs a number  $Z$  such that  $\|X\|_{p,q} \leq Z \leq \alpha \|X\|_{p,q}$  with probability at least  $2/3$  requires  $\Omega(n^{1-2/p} d^{1-2/q}/\alpha^2)$  bits of space. For the case that  $1 \leq p < 2$  and  $q > 2$ , any one-pass streaming algorithm which outputs a  $Z$  such that  $\|X\|_{p,q} \leq Z \leq \alpha \|X\|_{p,q}$  with probability at least  $1 - \delta$  requires  $\Omega(d^{1-2/q} \log(M) \log(1/\delta)/\alpha^2)$  bits of space.

## F Rectangle $F_p$ ( $p > 2$ )

In this section, we consider the rectangle  $F_p$  problem. A rectangle-efficient algorithm was proposed in [49]. Instead of updating the counter in each coordinate inside a rectangle, they develop a rectangle-efficient data structure called RECTANGLECOUNTSKETCH. We follow their notation that  $O^*(f)$  denotes a function of the form  $f \cdot \text{poly}(\log(mn/\delta))$  for constant rectangle dimension  $d$ .

► **Theorem 41.** Suppose that  $p > 2$ . There is a rectangle-efficient one-pass streaming algorithm which outputs a number  $Z$  that is an  $\alpha$ -approximation to  $\|x\|_p^p$ , i.e.,  $\|x\|_p^p \leq Z \leq \alpha \|x\|_p^p$ , with probability at least  $1 - \delta$ . It uses  $O^*(n^{d(1-2/p)}/\alpha^{2/p})$  bits of space and  $O^*(n^{d(1-2/p)}/\alpha^{2/p})$  time to process each rectangle in the stream.



# Local Stochastic Algorithms for Alignment in Self-Organizing Particle Systems

Hridesh Kedia ✉

Georgia Institute of Technology, Atlanta, GA, USA

Shunhao Oh ✉

Georgia Institute of Technology, Atlanta, GA, USA

Dana Randall ✉ 

Georgia Institute of Technology, Atlanta, GA, USA

---

## Abstract

We present local distributed, stochastic algorithms for *alignment* in self-organizing particle systems (SOPS) on two-dimensional lattices, where particles occupy unique sites on the lattice, and particles can make spatial moves to neighboring sites if they are unoccupied. Such models are abstractions of programmable matter, composed of individual computational particles with limited memory, strictly local communication abilities, and modest computational capabilities. We consider oriented particle systems, where particles are assigned a vector pointing in one of  $q$  directions, and each particle can compute the angle between its direction and the direction of any neighboring particle, although without knowledge of global orientation with respect to a fixed underlying coordinate system. Particles move stochastically, with each particle able to either modify its direction or make a local spatial move along a lattice edge during a move. We consider two settings: (a) where particle configurations must remain simply connected at all times and (b) where spatial moves are unconstrained and configurations can disconnect.

Our algorithms are inspired by the *Potts model* and its planar oriented variant known as the *planar Potts model* or *clock model* from statistical physics. We prove that for any  $q \geq 2$ , by adjusting a single parameter, these self-organizing particle systems can be made to collectively align along a single dominant direction (analogous to a solid or ordered state) or remain non-aligned, in which case the fraction of particles oriented along any direction is nearly equal (analogous to a gaseous or disordered state). In the connected SOPS setting, we allow for two distinct parameters, one controlling the ferromagnetic attraction between neighboring particles (regardless of orientation) and the other controlling the preference of neighboring particles to align. We show that with appropriate settings of the input parameters, we can achieve *compression* and *expansion*, controlling how tightly gathered the particles are, as well as *alignment* or *nonalignment*, producing a single dominant orientation or not. While alignment is known for the Potts and clock models at sufficiently low temperatures, our proof in the SOPS setting are significantly more challenging because the particles make spatial moves, not all sites are occupied, and the total number of particles is fixed.

**2012 ACM Subject Classification** Mathematics of computing → Stochastic processes; Theory of computation → Random walks and Markov chains; Theory of computation → Self-organization

**Keywords and phrases** Self-organizing particle systems, alignment, Markov chains, active matter

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.14

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2207.07956>

**Funding** *Hridesh Kedia*: Supported by ARO MURI award W911NF-19-1-0233.

*Shunhao Oh*: Supported by NSF award CCF-1733812 and ARO MURI award W911NF-19-1-0233.

*Dana Randall*: Supported by NSF awards CCF-1733812 and CCF-2106687 and ARO MURI award W911NF-19-1-0233.



© Hridesh Kedia, Shunhao Oh, and Dana Randall;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 14; pp. 14:1–14:20



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Autonomous, locally interacting agents can collectively organize to accomplish a variety of complex tasks such as foraging for food, building large-scale structures, and transporting objects many times heavier than their weight, as is routinely observed in the living world, in swarms of ants, flocks of birds, and schools of fish [34, 39, 38, 33]. A key component of these diverse self-organized behaviors is achieving consensus in large collectives of autonomous agents with only local interactions. The problem of achieving alignment in collectives of directed agents is an important example of such a consensus problem, and is a fundamental aspect of *flocking*: large scale collective motion in swarms of motile agents [34, 37, 28, 41, 38, 1]. While flocking has been studied extensively [18, 28, 36, 1] with few rigorous results, the more basic problem of alignment has received considerably less attention.

Here, we study alignment in self-organizing particle systems (SOPS) – a collection of simple, active computational particles that individually execute local distributed algorithms. We consider *oriented particle systems* on a two-dimensional lattice, where particles are oriented in one of  $q$  directions (with no global compass), for  $q \geq 2$ , and at most one particle occupies each lattice site. Particles perform moves independently and concurrently by making spatial moves to neighboring empty sites or reorient themselves in new directions with the goal of reaching nearly global alignment.

We consider a stochastic approach, used previously in [7, 8] to achieve *compression*, where connected sets of homogeneous particles self-organize to gather together tightly, *separation* in heterogeneous particle systems, where all of the particles compress, but also gather most tightly with other particles of the same type [5, 6], and *aggregation* of homogeneous particles that are not required to be connected, where most particles accumulate in a small, compact neighborhood [21]. In all of these, phase changes were used to characterize desirable behaviors at stationarity, with high probability. Following a similar approach, we begin by defining an energy function that assigns the highest weight (or lowest energy) to preferable configurations, and design a Markov chain whose long term behavior favors these low energy configurations using transition probabilities given by the Metropolis-Hastings algorithm [25, 15]. We ensure that the transition probabilities of the Markov chain can be computed locally and asynchronously, allowing them to be easily translated to a fully local, distributed algorithm that each particle can run independently. The collective behavior of this distributed algorithm is thus described by the long term behavior of the Markov chain.

### 1.1 Related work

The alignment problems we study can be viewed as finite, unsaturated variants of the ferromagnetic *Potts model* from statistical physics [40], and a related model known as the *clock* or *planar Potts model* [40, 29]. In the Potts model, vertices of a graph  $G$  are assigned one of  $q$  possible “spins,” represented here as orientations, and neighboring sites prefer to agree. Let  $J > 0$  be a parameter related to inverse temperature and let  $\delta(X, Y) = 1$  if  $X = Y$  and 0 otherwise. Then the probability of a standard Potts configuration  $\sigma$  is given as

$$\pi(\sigma) = \exp\left(-J \sum_{x \sim y} \delta(\sigma(x), \sigma(y))\right) / Z,$$

where the sum is taken over all nearest neighbors in  $G$  and  $Z$  is the normalizing constant or partition function. In the unsaturated setting studied here, spins are identified with particles, not sites, and particles can make spatial moves to unoccupied sites in addition to updating

their spins. We present alignment algorithms for two natural variants: (a) the connected setting, where particles are constrained to be simply connected in the lattice, and (b) the general setting, where particles occupy any distinct lattice sites regardless of connectivity.

Recent work on a closely related *site-diluted* Potts model [40, 9] also allows a non-zero fraction of lattice sites to be unoccupied, but the number of particles is not fixed, so particles can appear and disappear, in addition to making spatial moves. Chayes et al. [9] beautifully demonstrate the presence of ordered (aligned and occupied) and disordered (non-aligned and vacant) phases, along with novel “staggered” phases in this model. However, our constraint fixing the number of particles, which is necessary in SOPS models in programmable matter, makes our system fundamentally different from the site-diluted Potts model akin to the difference between the fixed magnetization Ising model, which has a fixed number of  $+$  spins, and the Ising model in the presence of a magnetic field, where the number of  $+$  spins can vary. Notably, the coexistence of phases that characterize the aligned and compressed behaviors we are seeking will not occur unless we fix the magnetization (or numbers of particles) as these configurations are exponentially unlikely in the site-diluted model and thus do not inherit any of its properties.

Since particles can make spatial moves, the boundary between the particle occupied sites and the unoccupied sites can assume arbitrary shapes, which makes achieving alignment more challenging than achieving compression. Consider the configurations shown in Figure 1(a),(b), where the particles can be oriented along one of two possible directions ( $q = 2$ ) shown by black and grey circles, with a total of  $n$  particles. While the number of unaligned pairs of adjacent particles is  $O(\sqrt{n})$  for the configuration in Figure 1(a), it can be as low as  $O(1)$  for the configuration shown in Figure 1(b), owing to the bottleneck shaped part of the configuration boundary, making it likely that the regions on either side of it will be aligned along different directions. Hence, achieving alignment requires suppressing the likelihood of such bottlenecks in the boundary of the particle configuration.

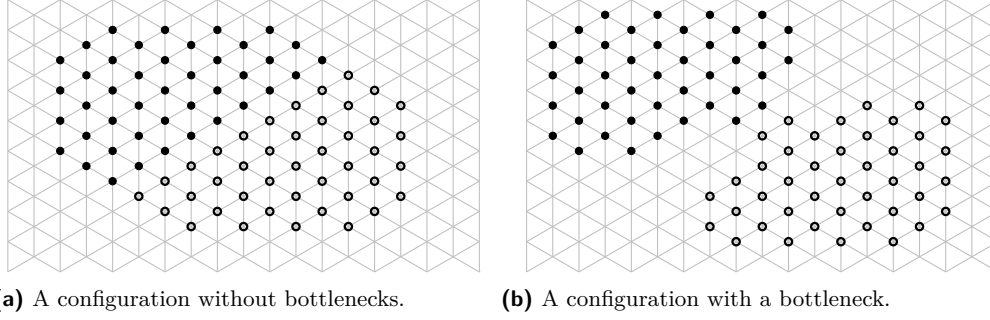
While the concept of an interfacial free energy can be used to constrain the shape of the boundary of a dilute system of homogeneous particles i.e., when  $q = 1$ , as in [26, 17, 2, 30], because particle occupied sites and vacant sites are akin to distinct coexisting phases of the system. However, the same ideas do not readily generalize to the case when  $q \geq 2$ . Instead, we show build on the notion of compression introduced in [7, 8], and use isoperimetric inequalities to show that for sufficiently compressed configurations, bottlenecks such as the one shown in Figure 1(b) are precluded with high probability.

## 1.2 Results

We present the first rigorous local distributed algorithms for achieving both low perimeter boundaries and alignment, for any number of orientations  $q \geq 2$ , in both connected and general settings. Informally, we say a particle system is *aligned* if a significant percentage of the particles have the same orientation.

In the connected SOPS setting, we define an energy function that encourages compression of the entire configuration and also defines a ferromagnetic interaction between particles’ orientations, inspired by the clock and Potts models. These two contributions are controlled by two independent parameters  $\lambda$  and  $\gamma$ . In this setting, we show that given any  $\alpha > 1$ , for any  $\lambda > 1$  and  $\gamma > 29.3(q - 1)$  such that  $\lambda\gamma > 7^{\alpha/(\alpha-1)}$ , the algorithms achieve  $\alpha$ -compression with high probability. Furthermore, when  $\gamma$  satisfies additional constraints given in Theorem 11, we show that the compressed configurations are very likely to be aligned. Next, we show that setting  $\lambda$  large and  $\gamma$  small will generate compressed configurations





■ **Figure 1** Configurations with two dominant orientations (black vs. gray circles); large interfaces as in (a) are unlikely for large  $\gamma$ , whereas small interfaces as in (b) are likely for any finite  $\gamma$ .

with an equitable balance of orientations (Theorem 18), while setting  $\lambda$  small will generate configurations that are *expanded*, nearly maximizing their perimeters, allowing the SOPS to explore space, potentially to forage for resources, for example (see Theorem 19).

For both the Potts and clock models in the connected setting, the proofs rely on the *cluster expansion* [23, 14, 20] from statistical physics, introducing a new so-called *polymer model* inspired by the relationship between flows and the Potts model [13]. Informally, the cluster expansion allows us to obtain upper and lower bounds on the so-called “polymer partition function” in terms of the volume and surface contributions, as in [5, 6, 14], to prove that our algorithms achieve compression (or aggregation), with high probability. Moreover, using isoperimetric inequalities, we prove the absence of bottlenecks in sufficiently highly compressed configurations, which is necessary to get the system to globally align. Finally, we use the *bridging techniques* first proposed in [27] and later adapted in [5, 6], to expand the information theoretic arguments in [5, 6] to prove that for sufficiently compressed configurations, our algorithms achieve alignment with high probability. Conversely, we show that our algorithms can achieve *expansion* and/or *non-alignment* (with all directions nearly equitably balanced), with the same algorithm by adjusting only two global parameters.

In the general SOPS setting, with no connectivity constraints, we present an algorithm based on a single parameter coupling both compression and ferromagnetism simultaneously. When this parameter is sufficiently large, we achieve aggregation and alignment, while when it is small we achieve expansion and a balance among the orientations (Theorem 21). We believe these parameters can be independently controlled in the general (disconnected) setting, but the proofs seemingly become significantly more challenging and coupling them into one parameter seems sufficient for most applications in programmable matter and swarm robotics. Because configurations tend to be highly disconnected, proofs in the general setting require additional technology to account for many small clusters that can be distributed throughout the lattice. Here we generalize the bridging techniques to account for more complex contours that form an interconnected network to show that the contour lengths of the bridging system can be made arbitrarily close to their minimum possible length and, as a result, alignment occurs with high probability. We note that our algorithms for alignment in both settings work for all  $q \geq 2$ ; separation (where the sizes of the color classes are fixed) has only been shown for  $q = 2$ , although the methods should also generalize to more colors [6].



## 2 Preliminaries

Our model of programmable matter is based on the *amoebot model*, introduced in [11] and described in detail in [10], which has served as the basis for previous stochastic algorithms for SOPS [8, 7, 6, 5]. In the amoebot model, particles occupy the nodes of a graph with each node occupied by at most one particle. When executing a spatial move, a particle expands into an adjacent unoccupied node, temporarily occupying both nodes and then contracts to the new node. Each particle stores whether it is expanded or contracted and can read whether its neighbors are expanded or contracted. No particle has access to global information such as system size or a shared co-ordinate system or compass.

We extend the amoebot model to model heterogeneous particles, where each particle has one of  $q$  orientations, akin to the variant introduced in [6, 5]. Each particle, when activated, chooses either a spatial move as in the original amoebot model, or an “orientation move” that updates its direction, each equal probability. The system performs these *atomic actions*, following the *ASync* model of computation from distributed computing [22]. It has been shown in this model that for any concurrent asynchronous execution of atomic actions, there exists a sequential ordering of actions with the same end state provided that all conflicts arising in the concurrent asynchronous execution are resolved. We assume that conflicts due to multiple particles expanding into an unoccupied node are resolved arbitrarily so that only one particle expands into the unoccupied node, allowing us to consider only one particle to be active at any given time.

### 2.1 The Potts and clock models

In our models, each configuration is an assignment of  $n$  particles to distinct vertices of a finite triangular lattice  $G_\Delta$  of  $N > n$  vertices with the toroidal topology. In addition, each particle is also assigned an orientation from  $\{0, 1, \dots, q-1\}$ . We assume  $G_\Delta$  to inhabit a  $\sqrt{N} \times \sqrt{N}$  square region with periodic boundary conditions. Each vertex  $(x, y)$  of  $G_\Delta$  has six outgoing edges, to the vertices  $(x+1, y)$ ,  $(x, y+1)$ ,  $(x+1, y+1)$ ,  $(x-1, y)$ ,  $(x, y-1)$ ,  $(x-1, y-1)$ , where addition and subtraction is taken modulo  $\sqrt{N}-1$ . Moreover, in this setup, the set of particles in our configurations must always be connected and hole-free. Given such a configuration, we define its *boundary*  $\mathcal{P}$  to be the minimal closed walk over occupied sites of  $G_\Delta$  that encloses all of the occupied sites in the configuration. The *perimeter*  $p(\sigma)$  of a configuration  $\sigma$  is then defined to be the length of this closed walk.

We consider the following Potts Hamiltonian, on  $G_\Delta$ , a variant of the site-diluted Potts model [9]:

$$H_{\text{Potts}}(\sigma) = -J \sum_{\langle i, j \rangle} n_i n_j \delta(\theta_i, \theta_j) - \kappa \sum_{\langle i, j \rangle} n_i n_j,$$

where the sum is over all pairs of adjacent sites:  $\langle i, j \rangle$  i.e., sites connected by a single lattice edge in  $G_\Delta$ ,  $n_i \in \{0, 1\}$  indicates whether site  $i$  is occupied or not,  $\theta_i$  indicates the orientation of the particle on site  $i$ , and  $J, \kappa$  are positive constants. We only consider configurations  $\sigma$  in  $\Omega$ , i.e., where the total number of particles is equal to  $n$ , and the particle-occupied sites form a connected, hole-free region.

The probability of a configuration  $\pi_{\text{Potts}}(\sigma)$  is given by the Boltzmann distribution:

$$\pi_{\text{Potts}}(\sigma) = e^{-\beta H_{\text{Potts}}(\sigma)} / Z_{\text{Potts}}, \quad \text{where} \quad Z_{\text{Potts}} = \sum_{\sigma' \in \Omega} e^{-\beta H_{\text{Potts}}(\sigma')},$$

where  $\beta$  denotes the inverse temperature. Setting parameters  $\lambda = \exp(\beta\kappa)$ , and  $\gamma = \exp(\beta J)$ , the above probability distribution can be expressed as:

$$\pi_{\text{Potts}}(\sigma) = \frac{w_{\text{Potts}}(\sigma)}{Z_{\text{Potts}}}, \quad w_{\text{Potts}}(\sigma) = (\lambda\gamma)^{-p(\sigma)}\gamma^{-h(\sigma)}, \quad Z_{\text{clock}} = \sum_{\sigma' \in \Omega} w_{\text{Potts}}(\sigma'), \quad (1)$$

where  $h(\sigma)$  is the number of heterogeneous edges in  $\sigma$ , i.e., edges connecting particles with different orientations, and  $p(\sigma)$  is its perimeter, as defined earlier. Here  $\pi_{\text{Potts}}$  is the stationary distribution for our Markov chain algorithm based on the ferromagnetic Potts model interactions.

Similarly, we consider the following clock model Hamiltonian on  $G_\Delta$ :

$$H_{\text{clock}}(\sigma) = -J \sum_{\langle i,j \rangle} n_i n_j \cos(2\pi(\theta_i - \theta_j)/q) - \kappa \sum_{\langle i,j \rangle} n_i n_j.$$

The probability of a configuration  $\pi_{\text{clock}}(\sigma)$  is given by the Boltzmann distribution as before, and can be expressed in terms of the parameters  $\lambda, \gamma$  as:

$$\pi_{\text{clock}}(\sigma) = \frac{w_{\text{clock}}(\sigma)}{Z_{\text{clock}}}, \quad w_{\text{clock}}(\sigma) = (\lambda\gamma)^{-p(\sigma)} \prod_{\langle i,j \rangle} \gamma^{-d_{ij}}, \quad Z_{\text{clock}} = \sum_{\sigma' \in \Omega} w_{\text{clock}}(\sigma'), \quad (2)$$

where  $\lambda > 0, \gamma > 0$  (as before),  $d_{ij} := 1 - \cos(2\pi(\theta_i - \theta_j)/q)$ , and the product is over all pairs of adjacent occupied sites. Here  $\pi_{\text{clock}}$  will be the stationary distribution for our Markov chain algorithm based on the clock model.

For each of the above models, we will refer to  $w(\sigma)$  ( $w_{\text{Potts}}$  or  $w_{\text{clock}}$ ) as the *weight* of a configuration. The stationary probability distribution  $\pi$  ( $\pi_{\text{Potts}}$  or  $\pi_{\text{clock}}$ ) is thus simply the weight function  $w$  normalized by the *partition function*  $Z$  ( $Z_{\text{Potts}}$  or  $Z_{\text{clock}}$ ).

## 2.2 Cluster expansions and bridging

Our proofs build on several tools from statistical physics and combinatorics, so we begin by introducing two key methods. The cluster expansion is one of the oldest tools in statistical physics [23, 24, 14], and has led to the development of the Pirogov-Sinai theory [31, 32], playing an important role in recent advances in efficient sampling and counting algorithms [16, 19, 3]. The cluster expansion expresses the logarithm of a polymer partition function as a sum over polymer clusters.

Let  $\mathcal{L}$  be a finite set of polymers  $\{\xi_i\}$ , where each polymer  $\xi_i$  has weight  $w(\xi_i)$ . We also define “compatibility” between polymers - each pair of polymers  $\xi, \xi'$  is either compatible ( $\xi \sim \xi'$ ) or incompatible ( $\xi \approx \xi'$ ). The polymer partition function is then given by:

$$\Xi = \sum_{\tau \in \Omega^{\mathcal{L}}} \prod_{\xi \in \tau} w(\xi),$$

where  $\Omega^{\mathcal{L}}$  is the set of all collections of pairwise compatible polymers in  $\mathcal{L}$ . The cluster expansion expresses the logarithm of the polymer partition function in terms of clusters, where a cluster  $X$  is an ordered multiset of polymers  $\{\xi_1, \dots, \xi_k\}$  such that their incompatibility graph  $H(X)$  is connected, where the incompatibility graph is constructed by representing each polymer by a vertex and connecting two vertices if the corresponding polymers are incompatible. The cluster expansion gives:

$$\log \Xi = \sum_{X \in \mathcal{C}} \Psi(X), \quad \text{where } \Psi(X) := \frac{1}{|X|!} \left( \sum_{G \subseteq H_X} (-1)^{|E(G)|} \right) \left( \prod_{\xi \in X} w(\xi) \right),$$

where the sum is taken over connected, spanning subgraphs  $G$  and  $\mathcal{C}$  is the set of all clusters. A sufficient condition for the convergence of the cluster expansion was given by Kotecký and Preiss [20]. We will prove this condition in Lemma 7 and use the cluster expansion to separate the volume and surface contributions to the partition function, as done in [14, 6].

Bridging is a combinatorial technique used to show that large contours are uncommon, while allowing for the possibility of many small contours corresponding to “defects”. It was first introduced in [27] and later adapted in [6]. We note that a constant fraction of defects will be unavoidable - an example of this is in the Ising model and Potts models, where a constant fraction of the vertices will not follow the majority color even at stationarity. Each configuration corresponds to a set of contours - informally, a bridge system comprises of a set of bridges, which are edges on the dual graph on the lattice that connect contours to the boundary of the lattice. Contours that are connected this way are called bridged contours, while the remaining contours are unbridged.

Bridge systems are defined so that the total length of the bridges is at most a constant fraction of the total length of the bridged contours, which allows us to bound the number of bridge systems with total bridged contour length  $\ell$  by  $C^\ell$  for some constant  $C$ . Consequently, a *Peierls argument* can be used to show that the gain in energy (probability weight) by the removal of the bridged contours is greater than the loss in entropy by the removal of these contours. Explicit constructions of bridge systems are shown in [6] and in our proof of alignment for disconnected SOPS (see Appendix A).

### 3 Compression and Alignment in Connected SOPS

Starting with any simply connected set of particles, we define a local Markov chain aiming to simultaneously compresses the configuration and align all but a small fraction of their orientations. On each iteration, a particle is activated uniformly at random using a Poisson clock. When activated, a particle chooses to attempt a spatial move or a reorientation move with a equal probability. Informally, spatial moves consist of the particle moving to a randomly chosen neighboring site, provided that site is unoccupied and the particle configuration remains simply connected, while a reorientation move allows the particle to change its orientation to point in a new direction. While it is surprising that a property such as connectivity can be determined locally, a set of local moves were defined in Cannon et al. [8] that prevent the configuration from disconnecting or forming holes and yet the chain remains ergodic on the infinite lattice, so all valid configurations can still be reached. This ergodicity result carries over to our setting as the we use a lattice that while finite, is sufficiently large that self-intersections via wraparound are not possible. Using the Metropolis-Hastings algorithm [25], once a move is determined to be valid, it is implemented with probability  $\min\{1, \pi(\sigma')/\pi(\sigma)\}$ , where  $\pi$  is the desired stationary distribution.

More precisely, consider a spatial move from a location  $\ell$  to an empty adjacent location  $\ell'$ . Let the sets of lattice sites adjacent to the locations  $\ell$  and  $\ell'$  be  $N(\ell)$  and  $N(\ell')$  respectively. Furthermore, let  $N(\ell \cup \ell')$  denote  $N(\ell) \cup N(\ell') \setminus \{\ell, \ell'\}$ , and  $\mathbb{S} := N(\ell) \cap N(\ell')$  denote the set of sites adjacent to both  $\ell$  and  $\ell'$  so that  $|\mathbb{S}| \in \{0, 1, 2\}$ .

► **Definition 1.** A move from  $\ell$  to  $\ell'$  is valid if  $\ell'$  is unoccupied, the number of particle-occupied sites in  $N(\ell)$  is less than 5, and either of the following two properties are satisfied:

Property 1:  $|\mathbb{S}| \geq 1$  and every particle-occupied site in  $N(\ell \cup \ell')$  is connected to a particle-occupied site in  $\mathbb{S}$  through  $N(\ell \cup \ell')$ .

Property 2:  $|\mathbb{S}| = 0$ ,  $\ell$  and  $\ell'$  each have at least one neighbor, and all particle-occupied sites in  $N(\ell) \setminus \{\ell'\}$  are connected by paths within this set, and all occupied sites in  $N(\ell') \setminus \{\ell\}$  are connected by paths within this set.

Note that in Section 4, we will consider almost the same algorithm in the general SOPS setting where there are no connectivity restrictions, so there all spatial moves from an occupied site to an adjacent unoccupied site are valid.

It is important to note that the ratio between the probabilities  $\pi(\sigma')/\pi(\sigma)$  that arises from the Metropolis-Hastings algorithm can be calculated by an activated particle using only local information - the positions and orientations of particles in its immediate neighborhood, as well as those in the neighborhood of the destination site if the particle is moving. Specifically, changes in perimeter in connected SOPS can be computed locally as shown in [8, 7].

We now proceed to show that when  $\lambda$  and  $\gamma$  are sufficiently large, the alignment algorithm will cause the system to compress to form a low-perimeter configurations with high probability. Moreover, in both the Potts and clock model settings, in any configuration with sufficiently low-perimeter, one of the  $q$  orientations will dominate with high probability.

We note that we did not attempt to give rigorous bounds on the rates of convergence for our Markov chains. We expect that convergence will be fast when the parameters  $\lambda$  and  $\gamma$  are small and the system evolves to a disordered (gaseous) state, but the connectivity constraint makes proving this challenging. In contrast, we expect convergence to equilibrium will be slow in the ordered (solid) state when  $\lambda$  is large, but we conjecture that desirable compressed and aligned states will be reached quickly, long before the system is very close to stationarity.

### 3.1 Compression in Connected SOPS

We denote the set of possible configurations in this paradigm by  $\Omega$ . Recall that  $N$  represents the number of sites of the lattice  $G_\Delta$ . To ensure that the proof of ergodicity from [7] carries over to our setting, we use a sufficiently large value of  $N$ , namely  $N \geq (n+1)^2$ , although we expect the results to hold for smaller  $N$ .

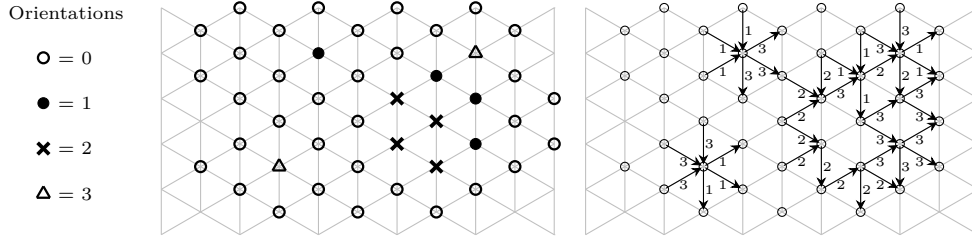
► **Definition 2 (Compression).** *A simply connected configuration  $\sigma$  of  $n$  particles on a lattice is said to be  $\alpha$ -compressed if its perimeter is at most  $\alpha \cdot p_{\min}(n)$ , where  $p_{\min}(n)$  is the minimum possible perimeter of a configuration of  $n$  particles.*

The main result of this section is the following theorem.

► **Theorem 3.** *Given any  $\alpha > 1$ , if constants  $\lambda > 1$  and  $\gamma > 29.3(q-1)$  satisfy  $\lambda\gamma > 7^{\alpha/(\alpha-1)}$  and  $n$  is sufficiently large, then the probability a configuration drawn from the stationary distribution  $\pi_{\text{Potts}}$  is not  $\alpha$ -compressed is exponentially small.*

Let  $\mathcal{P}$  denote the boundary of some configuration  $\sigma$  in our configuration space  $\Omega$ . As  $\sigma$  is connected, hole-free, and contains a finite ( $n$ ) number of particles,  $\mathcal{P}$  is a single closed walk on  $G_\Delta$  and the perimeter of the configuration,  $p(\sigma)$ , is equal to  $|\mathcal{P}|$ , the total length of walk  $\mathcal{P}$ . If we restrict our particle configurations to be connected and hole-free, there is a one-to-one correspondence between the possible sets of occupied sites and the possible boundaries  $\mathcal{P}$ . Let  $\Omega_{\mathcal{P}}$  denote the set of configurations in  $\Omega$  with boundary  $\mathcal{P}$ , and let  $\Lambda_{\mathcal{P}} \subseteq G_\Delta$  be the induced subgraph of the triangular lattice  $G_\Delta$  by the particle-occupied vertices for any configuration in  $\Omega_{\mathcal{P}}$ . A configuration in  $\Omega_{\mathcal{P}}$  thus corresponds to a mapping of the vertices of  $\Lambda_{\mathcal{P}}$  to the orientations  $\{0, \dots, q-1\}$ .

We consider the subset of configurations  $\Omega_{\mathcal{P}}^0 \subseteq \Omega_{\mathcal{P}}$  where all particles on the boundary  $\mathcal{P}$  have the same color 0. We will later analyze the weight of configurations in  $\Omega_{\mathcal{P}}^0$  using a polymer model and the cluster expansion. We would first like to obtain an upper bound on  $w(\Omega_{\mathcal{P}})$ , the total weight of configurations in  $\Omega_{\mathcal{P}}$ , in terms of  $w(\Omega_{\mathcal{P}}^0)$ , the total weight of configurations in  $\Omega_{\mathcal{P}}^0$ .



■ **Figure 2** Particle configuration in  $\Omega_{\mathcal{P}}^0$ , and its corresponding polymer configuration in  $\Omega_{\mathcal{P}}^{\mathcal{L}}$  (with two polymers).

► **Lemma 4.** *For  $\gamma > 3q$ , we have*

$$w(\Omega_{\mathcal{P}}) < w(\Omega_{\mathcal{P}}^0) \cdot q 2^{|\mathcal{P}|} \frac{\gamma}{\gamma - 3q}.$$

The proof is a generalized version of that in [6], by defining maps from  $\Omega_{\mathcal{P}} \rightarrow \Omega_{\mathcal{P}}^0$  such that all vertices on boundary  $\mathcal{P}$  are of orientation 0. We will use the cluster expansion to analyze the total weight  $w(\Omega_{\mathcal{P}}) := \sum_{\sigma \in \Omega_{\mathcal{P}}} w(\sigma)$  of the configurations in  $\Omega_{\mathcal{P}}$ . Since the cluster expansion can only be applied to polymer partition functions, we begin by representing the configurations of  $\Omega_{\mathcal{P}}$  with a polymer model.

**The Polymer Model.** We say two edges of  $G_{\Delta}$  are adjacent if they share a common vertex. A polymer  $\xi$  in  $\mathcal{L}$  is defined to be a labeling  $\xi : E(G_{\Delta}) \rightarrow \{0, 1, \dots, q-1\}$  of the edges of  $G_{\Delta}$  such that the set  $E(\xi)$ , defined to be the edges of  $G_{\Delta}$  with a non-zero label in  $\xi$ , is non-empty and connected under the above notion of adjacency. The labeling must also be *consistent*, as defined below.

► **Definition 5** (Consistent Labeling). *We fix a canonical direction for each edge in  $G_{\Delta}$ . This direction can be arbitrarily defined, so for simplicity we say that the edge is oriented toward the vertex with the larger  $x$ , followed by  $y$  coordinate, where the coordinate axes are oriented such that the  $x$  coordinate increases from left to right and the  $y$  coordinate increases from top to bottom.*

*We define labels  $\xi : E(G_{\Delta}) \rightarrow \{0, 1, \dots, q-1\}$ . These edge labels represent “flows” in our defined canonical direction, modulo  $q$ . In other words, when summing up the total flow along a walk on  $G_{\Delta}$ , for each edge  $e$  on the walk, we add the label  $\xi(e)$  to the sum if the walk is in the canonical direction of the edge, and  $q - \xi(e)$  if the walk is in the opposite direction. We call an assignment of labels consistent if every closed walk on  $G_{\Delta}$  has a total flow summing to 0 modulo  $q$ .*

Consider a fixed boundary  $\mathcal{P}$  as defined above, corresponding to some configuration in  $\Omega$ . For a polymer  $\xi$ , denote by  $V(\xi)$  the set of vertices incident to an edge with a non-zero label in  $\xi$ . We say a polymer  $\xi$  is within  $\mathcal{P}$  if  $V(\xi) \subseteq \Lambda_{\mathcal{P}}$ . As described earlier, the set  $\Omega_{\mathcal{P}}^{\mathcal{L}}$  of polymer configurations corresponding to  $\mathcal{P}$  is the set of all subsets of  $\mathcal{L}$  of pairwise compatible polymers within  $\mathcal{P}$ . The weight  $w(\tau)$  of a configuration  $\tau \in \Omega_{\mathcal{P}}^{\mathcal{L}}$  is the product of the weights of its constituent polymers.

Two polymers  $\xi_1, \xi_2$  are incompatible if there are edges  $e_1 \in E(\xi_1)$  and  $e_2 \in E(\xi_2)$  such that  $e_1$  and  $e_2$  are adjacent. The weight of a polymer  $\xi$  is defined as  $w(\xi) := \gamma^{-|E(\xi)|}$ , in the Potts model, and  $w(\xi) := \prod_{e \in E(\xi)} \gamma^{\cos(\frac{2\pi}{q}\xi(e)) - 1}$  in the clock model.

► **Lemma 6.** *There is a bijection  $\phi$  between  $\Omega_{\mathcal{P}}^0$  and  $\Omega_{\mathcal{P}}^{\mathcal{L}}$  with the property that for any  $\sigma \in \Omega_{\mathcal{P}}^0$ , we have  $w(\sigma) = (\lambda\gamma)^{-p(\sigma)} w(\phi(\sigma))$ .*

The map  $\phi$  simply encodes the orientations of particles in a configuration  $\sigma \in \Omega_{\mathcal{P}}^0$  as differences between orientations on the edges of  $G_{\Delta}$ . This is illustrated in Figure 2. The full version of the paper gives a full description of this mapping and a proof that it is indeed a bijection. From Lemma 6, we have

$$w(\Omega_{\mathcal{P}}^0) = \sum_{\sigma \in \Omega_{\mathcal{P}}^0} (\lambda\gamma)^{-|\mathcal{P}|} w(\phi(\sigma)) = \sum_{\tau \in \Omega_{\mathcal{P}}^{\mathcal{L}}} (\lambda\gamma)^{-|\mathcal{P}|} w(\tau) = (\lambda\gamma)^{-|\mathcal{P}|} \Xi_{\mathcal{P}},$$

where  $\Xi_{\mathcal{P}}$  is the partition function for the set of polymer configurations  $\Omega_{\mathcal{P}}^{\mathcal{L}}$ :

$$\Xi_{\mathcal{P}} := \sum_{\tau \in \Omega_{\mathcal{P}}^{\mathcal{L}}} w(\tau) = \sum_{\tau \in \Omega_{\mathcal{P}}^{\mathcal{L}}} \prod_{\xi \in \tau} w(\xi).$$

**The Potts Model.** From now, our analysis will be specific to the Potts model. The clock model will be discussed in Section 3.1. The following Lemmas and proofs are slight variations of those used in [6].

► **Lemma 7.** *For any polymer  $\xi \in \mathcal{L}$ , whenever  $\gamma > 29.3(q-1)$ , we have for  $c = 0.0001$ ,*

$$\sum_{\substack{\xi' \in \mathcal{L} \\ \xi' \sim \xi}} w(\xi') \exp(c|V(\xi')|) \leq c|V(\xi)|,$$

where  $V(\xi')$  denotes the set of vertices in the polymer  $\xi'$ , and  $|V(\xi')|$  denotes the number of vertices in  $\xi'$ .

The proof is on the lines of that in [6]. The key part of this proof is the use of an upper bound  $\nu(m, q) \leq (6e(q-1))^m/2$  from [4], where  $\nu(m, q)$  represents the number of polymers with  $m$  edges containing some fixed vertex  $v \in V(G_{\Delta})$ .

Lemma 7 has an important consequence in addition to guaranteeing the convergence of the cluster expansion, as stated in the original paper of Kotecký and Preiss [20], and rephrased in [19]. Consider the function  $\Psi(X)$  defined earlier for any cluster  $X$ . An additional consequence [20, 19] of Lemma 7 is that  $\Psi(X)$  will satisfy the following inequality

$$\sum_{\substack{X \in \mathcal{X} \\ X \sim \xi}} |\Psi(X)| \leq c|V(\xi)|. \quad (3)$$

for any polymer  $\xi$ , where  $\mathcal{X}$  is the set of all clusters of polymers, and a cluster  $X \sim \xi$  if there exists a polymer  $\xi' \in X$  such that  $\xi' \sim \xi$ . The support of a cluster  $X$  is denoted by  $\bar{X}$  and is given by  $\bar{X} = \bigcup_{\xi \in X} V(\xi)$ .

Consider an arbitrary vertex  $v \in G_{\Delta}$ , and let  $\xi_v$  be the smallest polymer consisting of six edges of equal weight attached to  $v$ . From Equation (3), we have:

$$\sum_{\substack{X \in \mathcal{X} \\ X \sim \xi_v}} |\Psi(X)| \leq c|V(\xi_v)| = 7c \Rightarrow \sum_{\substack{X \in \mathcal{X} \\ v \in \bar{X}}} |\Psi(X)| \leq \sum_{\substack{X \in \mathcal{X} \\ X \sim \xi_v}} |\Psi(X)| \leq 7c. \quad (4)$$

► **Lemma 8.** *If for any polymer  $\xi \in \mathcal{L}$ , there exists a constant  $c$  such that*

$$\sum_{\substack{\xi' \in \mathcal{L} \\ \xi' \sim \xi}} w(\xi') \exp(c|V(\xi')|) \leq c|V(\xi)|,$$

then for any connected region  $\Lambda_{\mathcal{P}}$  with boundary  $\mathcal{P}$ , the partition function  $\Xi_{\mathcal{P}}$  satisfies

$$\psi|\Lambda_{\mathcal{P}}| - 7c|\partial\Lambda| \leq \ln \Xi_{\mathcal{P}} \leq \psi|\Lambda_{\mathcal{P}}| + 7c|\partial\Lambda|.$$

The proof follows on the lines of the proof of a similar Lemma in [6], and section 5.7.1 of [14]. Using Lemma 8, and noting that  $|\partial\Lambda_{\mathcal{P}}| \leq p(\sigma) \forall \sigma \in \Omega_{\mathcal{P}}$  and  $|\Lambda_{\mathcal{P}}| = n$ , we get:

$$n\psi - 7cp(\sigma) \leq \ln \Xi_{\mathcal{P}} \leq n\psi + 7cp(\sigma) \quad (5)$$

Note that the partition function  $Z_{\text{Potts}}$  is greater than the contribution from particle configurations in  $\Omega_{\mathcal{P}}^0$  where the length of the boundary is the smallest attainable perimeter  $|\mathcal{P}| = p_{\min}$ :

$$Z_{\text{Potts}} \geq w(\Omega_{\mathcal{P}}^0) = (\lambda\gamma)^{-p_{\min}} \Xi_{\mathcal{P}} \geq (\lambda\gamma)^{-p_{\min}} e^{n\psi - 7cp_{\min}}. \quad (6)$$

Given  $\alpha > 1$ , let  $S_{\alpha}$  be all configurations that are not  $\alpha$ -compressed. We will prove that the probability of the set  $S_{\alpha}$  in the stationary distribution is exponentially small for sufficiently large  $\lambda, \gamma$ :

► **Lemma 9.** *Given any  $\alpha > 1$ , when constants  $\lambda > 1, c = 0.0001$ , and  $\gamma > 29.3(q-1)$  satisfy*

$$\lambda\gamma > (4 + 2\sqrt{2})^{\frac{\alpha}{\alpha-1}} (e^{7c})^{\frac{\alpha+1}{\alpha-1}} \quad (7)$$

*and  $n$  is sufficiently large, then the probability that a configuration drawn from the stationary distribution  $\pi_{\text{Potts}}$  is not  $\alpha$ -compressed is exponentially small,  $\pi_{\text{Potts}}(S_{\alpha}) < \zeta^{\sqrt{n}}$ .*

Note that Equation (7) is satisfied if  $\lambda\gamma > 7^{\alpha/(\alpha-1)}$ , proving Theorem 3. The proof of the Lemma requires using Lemma 4, Lemma 8 and Equation (6), and an upper bound on the number of self-avoiding walks of a given length on the triangular lattice from [12, 8].

**The Clock Model.** The proof of compression for the clock-model-inspired algorithm follows along the same lines as the proof for the Potts-model-inspired algorithm. The set of allowed particle configurations is the same as before, so the set of configurations in  $\Omega_{\mathcal{P}}^0$  is in a one-to-one correspondence with compatible collections of polymers with the same polymer model as above, albeit with the weight of a polymers redefined as  $w_{\text{clock}}(\xi) = \prod_{e \in \xi} \gamma^{-d_e}$ , where  $d_e = 1 - \cos(2\pi\ell(e)/q)$ , and  $\ell(e) \in \{1, 2, \dots, q-1\}$  is the label associated with an edge  $e \in \xi$ . This changes the prefactor in Lemma 4, replacing  $\gamma$  with  $\gamma^{-\cos(2\pi/q)}$ , and requiring  $\gamma^{-\cos(2\pi/q)} > 3q$ . The polymer partition function becomes

$$\Xi_{\mathcal{P}} = \sum_{\substack{\mathcal{L}' \subseteq \mathcal{L}_{\mathcal{P}} \\ \text{compatible}}} \prod_{\xi \in \mathcal{L}'} w_{\text{clock}}(\xi).$$

Since the maximum weight of an edge in a polymer is now  $\gamma^{-(1-\cos(2\pi/q))}$ , instead of  $\gamma^{-1}$ , the condition for Lemma 7 to hold becomes  $\gamma^{1-\cos(2\pi/q)} > 29.3(q-1)$ . Lemmas 8 and Theorem 3 follow without modification except for the modified condition:  $\gamma^{1-\cos(2\pi/q)} > 29.3(q-1)$  in Theorem 3.

### 3.2 Alignment in Compressed Configurations

► **Definition 10** (Alignment). *We say a configuration of  $n$  particles with  $q$  orientations is  $\delta$ -aligned if there exists an orientation  $\theta \in \{0, 1, \dots, q-1\}$ , such that the number of particles of orientation  $\theta$  is at least  $(1-\delta)n$ .*

Our main result is the following theorem:



► **Theorem 11.** Denote by  $\pi_{\text{Potts}, \mathcal{P}}$  the stationary distribution  $\pi_{\text{Potts}}$  conditioned on the boundary of the configuration being  $\mathcal{P}$ . For any  $\eta$  where  $1/2 < \eta < 1$ , there exists a constant  $\alpha^* = \alpha^*(\eta, q) > 1$ , such that for all  $\alpha$  where  $1 < \alpha < \alpha^*$ , there exists a sufficiently large  $\gamma^* = \gamma^*(\eta, q, \alpha, \alpha^*)$  where as long as  $\gamma > \gamma^*$  and  $\mathcal{P}$  is  $\alpha$ -compressed, the probability that a configuration drawn from  $\pi_{\text{Potts}, \mathcal{P}}$  is not  $(1 - \eta)$ -aligned is exponentially small.

In particular, possible values of  $\alpha^*$  and  $\gamma^*$  are:

$$\alpha^*(\eta, q) = \min \left\{ \sqrt{\eta} + \sqrt{1 - \eta}, \sqrt{q^{-1}} + \sqrt{1 - q^{-1}} \right\}$$

$$\gamma^*(\eta, q, \alpha, \alpha^*) = \left( 3^{\frac{2\alpha}{\alpha^* - \alpha}} \cdot 4^{\frac{3}{4} + \frac{\alpha^* - 1}{2\delta^*(\eta, q)(\alpha^* - \alpha)}} \right)^{q^{-1}} \text{ where } \delta^*(\eta, q) := \min\{1 - \eta, q^{-1}\}.$$

For any particle configuration, let  $2\pi\theta_p/q$ , be the *most popular orientation*, or the orientation possessed by the greatest number of particles, where  $\theta_p \in \{0, 1, \dots, (q - 1)\}$ , and let  $\rho_p$  be the fraction of particles with orientation  $\theta_p$ . Note that  $1/q \leq \rho_p \leq 1$ , and  $\rho_p \geq \eta$  for a  $(1 - \eta)$ -aligned configuration.

The dual lattice,  $G_\square$ , to the triangular lattice  $G_\Delta$  is obtained by creating a dual vertex in the center of each triangle in  $G_\Delta$ , and joining these dual vertices with edges if their corresponding triangular faces share an edge. Each edge  $e_\Delta$  of  $G_\Delta$  corresponds with the edge  $e_\square$  of  $G_\square$  that crosses it. This corresponding edge  $e_\square$  separates the two endpoints of  $e_\Delta$  in  $G_\Delta$ . A *contour* refers to a self-avoiding walk on the edges of the dual lattice  $G_\square$ . The length of a contour refers to the number of edges in the contour.

In this setting, we distinguish between the *boundary contour* and the *internal boundary contour* of a region  $R \subseteq V(\Lambda_{\mathcal{P}})$ . The boundary contour refers to the set of edges on the dual lattice  $G_\square$  corresponding to edges between sites in  $R$  and sites not in  $R$ , while the internal boundary contour includes edges only from  $E(\Lambda_{\mathcal{P}})$  rather than all of  $E(G_\square)$ . We make use of the following geometric result, which we show in the full version of the paper:

► **Lemma 12.** For a connected hole-free  $\alpha$ -compressed configuration with  $n$  particles, a particle-occupied region  $R$  containing  $\kappa n$  particles has an internal boundary contour  $\text{bd}_{\text{int}}(R)$  of length at least  $\nu\sqrt{n}(\sqrt{\kappa} + \sqrt{1 - \kappa} - \alpha)$  for any  $\nu < 2\sqrt{3}$  for sufficiently large  $n$ .

For the rest of this section, we assign particles the color  $c_1$  if they are of orientation  $\theta_p$ , and the color  $c_2$  otherwise. This lets us directly apply the bridging construction from [6].

► **Lemma 13** ([6], Lemma 7.3). Fix  $\delta \in (0, 1/2)$ . For each particle configuration  $\sigma \in \Omega_{\mathcal{P}}$ , there exists a function  $\mathcal{R}_\delta : \Omega_{\mathcal{P}} \rightarrow 2^{\Omega_{\mathcal{P}}}$  giving a region  $\mathcal{R}_\delta(\sigma)$  such that all particles on the boundary of  $\mathcal{R}_\delta(\sigma)$  have the color  $c_1$ , all particles on the boundary of its complement  $\bar{\mathcal{R}}_\delta(\sigma)$  have the color  $c_2$ ,  $\mathcal{R}_\delta(\sigma)$  contains at most  $\delta$  fraction of particles with the color  $c_2$ , and  $\bar{\mathcal{R}}_\delta(\sigma)$  contains at most  $\delta$  fraction of particles with the color  $c_1$ .

We use the bridging construction from [6] to define the region  $\mathcal{R}_\delta(\sigma)$  in Lemma 13.

► **Lemma 14.** For any particle configuration  $\sigma \in \Omega_{\mathcal{P}}$  with total number of particles  $n$  and  $\rho_p$  fraction of particles of color  $c_1$ , given any  $\delta > 0$ , the region  $\mathcal{R}_\delta(\sigma)$  defined in Lemma 13 is such that the number of particles in  $\mathcal{R}_\delta(\sigma)$ ,  $n_{\mathcal{R}_\delta}$  satisfies:  $(\rho_p - \delta)n/(1 - \delta) \leq n_{\mathcal{R}_\delta} \leq (\rho_p n)/(1 - \delta)$ .

The proof of Lemma 14 follows from noting that the particles in  $\mathcal{R}_\delta(\sigma)$  and  $\bar{\mathcal{R}}_\delta(\sigma)$  are predominantly of the colors  $c_1$  and  $c_2$  respectively, with an error fraction bounded by  $\delta$ , and enforcing that the total number of particles with the color  $c_1$  is  $\rho_p n$ .



► **Lemma 15.** *For a connected hole-free  $\alpha$ -compressed configuration  $\sigma \in \Omega_{\mathcal{P}}$  that is not  $(1 - \eta)$ -aligned for some  $\eta < 1$ , given any  $\delta$  where  $0 < \delta < \min\{q^{-1}, 1 - \eta\}$ , the internal boundary contour length  $|bd_{\text{int}}(\mathcal{R}_{\delta})|$  of the region  $\mathcal{R}_{\delta}(\sigma)$  defined in Lemma 13 obeys the lower bound  $|bd_{\text{int}}(\mathcal{R}_{\delta})| \geq \nu\sqrt{n}(\alpha_c(\delta, \eta, q) - \alpha)$  for any  $\nu < 2\sqrt{3}$  and  $n$  sufficiently large, where*

$$\alpha_c(\delta, \eta, q) := \min \left\{ \sqrt{\frac{q^{-1} - \delta}{1 - \delta}} + \sqrt{\frac{1 - q^{-1}}{1 - \delta}}, \sqrt{\frac{\eta}{1 - \delta}} + \sqrt{\frac{1 - (\eta + \delta)}{1 - \delta}} \right\}.$$

Lemma 15 is a direct consequence of Lemmas 14 and 12. Given an  $\alpha$ -compressed boundary  $\mathcal{P}$ , let  $S_{\mathcal{P}}^{\eta} \subseteq \Omega_{\mathcal{P}}$  be the set of  $\alpha$ -compressed configurations with boundary  $\mathcal{P}$  that are not  $(1 - \eta)$ -aligned for some  $\eta < 1$ . For each configuration  $\sigma \in S_{\mathcal{P}}^{\eta}$ , let  $\bar{\mathcal{R}}_{\delta}(\sigma)$  be the complement of the region  $\mathcal{R}_{\delta}(\sigma)$  defined in Lemma 13.

Let  $\mathcal{P}_{\bar{\mathcal{R}}_{\delta}}^{\text{int}}$  denote the walk on the edges of  $G_{\Delta}$ , each of whose endpoints is a particle in  $\bar{\mathcal{R}}_{\delta}(\sigma)$  that is connected by an edge in  $G_{\Delta}$  to a particle in  $\mathcal{R}_{\delta}(\sigma)$ . Let  $\Theta_{\bar{\mathcal{R}}_{\delta}}^{\text{int}}$  denote the set of orientations of particles that are incident to an edge in  $\mathcal{P}_{\bar{\mathcal{R}}_{\delta}}^{\text{int}}$ , where the orientation of a particle appears as many times as the number of edges connecting that particle to a particle in  $\mathcal{R}_{\delta}(\sigma)$ . Note that  $|\Theta_{\bar{\mathcal{R}}_{\delta}}^{\text{int}}| = |bd_{\text{int}}(\bar{\mathcal{R}}_{\delta})|$ . Let the orientation which appears the most number of times in the set  $\Theta_{\bar{\mathcal{R}}_{\delta}}^{\text{int}}$  be  $2\pi\bar{\theta}_p/q$ , where  $\bar{\theta}_p \in \{0, 1, \dots, q - 1\}$ . We consider a map  $f_{\eta} : S_{\mathcal{P}}^{\eta} \rightarrow \Omega_{\mathcal{P}}$  which applies a cyclic shift to the orientations of all particles in  $\bar{\mathcal{R}}_{\delta}(\sigma)$ , so that under  $f_{\eta}$ , a particle orientation  $\theta$  is mapped to  $(\theta + (\theta_p - \bar{\theta}_p)) \pmod{q}$ . Note that this transformation maps the orientation  $\bar{\theta}_p$  to  $\theta_p$ .

► **Lemma 16** ([6]). *For a configuration  $\tau \in \text{Im}(f_{\eta}(S_{\mathcal{P}}^{\eta}))$ , the number of preimages  $\sigma \in S_{\mathcal{P}}^{\eta}$  for which  $|bd_{\text{int}}(\mathcal{R}_{\delta}(\sigma))| = \ell$ , where  $\mathcal{R}_{\delta}(\sigma)$  is defined in Lemma 14, is at most  $q3^{|\mathcal{P}|}4^{\frac{1+3\delta}{4\delta}\ell}$ .*

The proof follows from Lemma 7.6 in [6] and by noting that once the internal boundary contour of  $\mathcal{R}_{\delta}(\sigma)$  is known, one of  $q$  cyclic shifts in  $\bar{\mathcal{R}}_{\delta}(\sigma)$  recovers  $\sigma$ , given  $\tau$ .

In this section so far, our results were valid for both the Potts and the clock models. We now consider specifically the case of the Potts model with stationary distribution  $\pi_{\text{Potts}}$ . Using the definition of  $f_{\eta}$ , we find the following.

► **Lemma 17.** *For a configuration  $\sigma \in S_{\mathcal{P}}^{\eta}$ , let region  $\mathcal{R}_{\delta}(\sigma)$  be defined as in Lemma 13 with  $|bd_{\text{int}}| = \ell$ . For the new configuration  $f_{\eta}(\sigma)$  under the map  $f_{\eta}$ , the ratio  $w(\sigma)/w(f(\sigma))$  is at most  $(1/\gamma)^{\ell/(q-1)}$ .*

The proof of Theorem 11 follows from an information theoretic argument similar to that in [6], by showing that the minimum gain in the weight of a configuration under the map  $f_{\eta}$  outweighs the maximum number of preimages of the map, and using Lemma 15 to get a lower bound on the gain under  $f_{\eta}$ . A key component is ensuring that it is possible to choose the parameter  $0 < \delta < q^{-1}$ , so that the conditions on  $\alpha$  and  $\gamma$  described in the theorem statement can be simultaneously satisfied.

**The Clock Model.** Lemma 17 and Theorem 11 hold for the clock model with stationary distribution  $\pi_{\text{clock}}$ , with  $\gamma$  replaced by  $\gamma^{1 - \cos(2\pi/q)}$  in both. The proofs follow on similar lines as for the Potts model.

### 3.3 Non-Alignment and Expansion in Connected SOPS

An interesting artifact of the alignment algorithm is that when  $\lambda, \gamma$  are small, the opposite properties are achieved, namely nonalignment and expansion. We outline the main results.

**Non-alignment in compressed configurations.** For  $\epsilon > 0$ , we say a configuration is  $\epsilon$ -non-aligned if the fraction of particles of each orientation is within an  $\epsilon$ -neighborhood of  $q^{-1}$ . Let  $S_{\mathcal{P}}^{\epsilon}$  denote the set of configurations which have perimeter  $\mathcal{P}$  and are not  $\epsilon$ -non-aligned, and let  $S^{\epsilon}$  be the set of configurations that are not  $\epsilon$ -non-aligned. Our main result is as follows:

► **Theorem 18.** *When  $\gamma > 0$  satisfies:*

$$\gamma^3 < \left(1 - \epsilon \frac{q}{q-1}\right)^{\frac{q-1}{q} - \epsilon} (1 + \epsilon q)^{\frac{1}{q} + \epsilon} = 1 + \frac{\epsilon^2 q^2}{q-1} + O(\epsilon^3),$$

*the probability that a configuration sampled from the stationary distribution of the Markov chain algorithm  $\pi_{\text{Potts}}$  is not  $\epsilon$ -non-aligned is exponentially small, for sufficiently large  $n$ .*

The proof follows from Stirling's approximation [35] for the number of configurations that are not  $\epsilon$ -non-aligned, and using rough lower and upper bounds on the weight of configurations in  $\Omega_{\mathcal{P}}$ . The result also holds for the clock model with  $\gamma$  replaced with  $\gamma^2$ .

**Expansion in Connected SOPS.** We define the notion of expansion, on the lines of [8], as follows. We say a configuration  $\sigma$  is  $\beta$ -expanded when its perimeter  $p(\sigma)$  is greater than  $\beta p_{\max}$ , where  $0 < \beta < 1$ . Consider the set of configurations  $S_{\beta}$  that are not  $\beta$ -expanded. Our main result is:

► **Theorem 19.** *For constants  $\lambda, \gamma > 0, c_1 = 2.17, c_2 = 2 + \sqrt{2}$  such that  $\lambda \gamma^{5/2} < c_1$ , and for any  $\beta$  such that:*

$$0 < \beta < \frac{\log c_1 - \log \lambda - \frac{5}{2} \log \gamma}{\log c_2 - \log \lambda - \log \gamma},$$

*the probability that a configuration drawn from the stationary distribution  $\pi$  is not  $\beta$ -expanded is exponentially small.*

We can get rough upper and lower bounds for the weight of configurations in  $\Omega_{\mathcal{P}}$  by estimating the number of ways of getting a fixed perimeter using the bounds in [12, 8].

The same theorem holds for the clock model, with  $\gamma^{5/2}$  replaced by  $\gamma^4$  in the theorem statement, and the proof follows on similar lines.

## 4 Aggregation and Alignment in General SOPS

In general SOPS, occupying any selection of  $n$  out of the  $N$  possible sites of  $G_{\Delta}$  is a valid configuration. Hence, we apply the same Metropolis-Hastings Markov chain as the connected SOPS model, with the exception that any move into an unoccupied location is considered valid regardless of connectivity effects. In this disconnected setting, particles exist on a lattice region with toroidal boundary conditions. We assume the particles occupy a constant fraction  $\rho$  of the lattice. Specifically, we define a  $\rho \in (0, \frac{1}{3})$  so that  $n = \rho N$ . The set of possible configurations is denoted  $\tilde{\Omega}^{\rho N}$ .

Similar to before, *boundary contour*  $bd(R)$  of a region  $R \subseteq V(G_{\Delta})$  refers to the set of dual edges on  $G_{\square}$  corresponding to edges between sites in  $R$  and  $V(G_{\Delta}) \setminus R$ . The *boundary length* of  $R$  is  $|bd(R)|$ . Let  $bd_{\min}(k)$  denote the minimum boundary length of a region of  $k$  sites in  $V(G_{\Delta})$ . We restrict  $\rho$  to be less than  $\frac{1}{3}$  as cases with so many particles (filled sites) that minimum boundary length configurations wrap around the torus  $G_{\Delta}$  is not instructive for our purposes (a precise explanation for this restriction is in the full version of the paper).

We show that in this general SOPS model, both alignment and aggregation can be achieved with high probability using only local movements. Alignment is defined in Section 3.2, and aggregation is defined as follows:

► **Definition 20 (Aggregation).** For  $\alpha > 1$ ,  $\delta > 0$  we say a configuration of  $n$  particles is  $\alpha, \delta$ -aggregated if there exists a region  $\mathcal{R}$  such that

1. The number of empty sites within  $\mathcal{R}$  is at most  $\delta|\mathcal{R}|$ .
2. The number of particles outside of  $\mathcal{R}$  is at most  $\delta(N - |\mathcal{R}|)$
3. The boundary length of  $\mathcal{R}$  is at most  $\alpha \cdot bd_{\min}(n)$ .

Note that changes in the perimeter of the configuration cannot be locally computed if the set of particles is disconnected. So instead, we make use of the boundary contour length to define our Hamiltonian. More precisely, we consider the following Potts Hamiltonian, another variant of the site-diluted Potts Hamiltonian [9], on  $G_\Delta$ :

$$\tilde{H}_{\text{Potts}}(\sigma) = -J \sum_{\langle i, j \rangle} [n_i n_j (\delta_{\theta_i, \theta_j} - 1) + (n_i(n_j - 1) + n_j(n_i - 1))] ,$$

where the sum is over all pairs of adjacent sites:  $\langle i, j \rangle$  i.e., sites connected by a single lattice edge in  $G_\Delta$ ,  $n_i \in \{0, 1\}$  indicates whether site  $i$  is occupied or not,  $\theta_i$  indicates the orientation of the particle on site  $i$ , and  $J$  is a positive constant. We only consider configurations  $\sigma$  in  $\tilde{\Omega}^{\rho N}$  i.e., where the total number of particles is equal to  $n$ .

The probability of a configuration  $\tilde{\pi}_{\text{Potts}}(\sigma)$  is given by the Boltzmann distribution which can be expressed in terms of the parameter  $\lambda = \exp(\beta J)$  as:

$$\tilde{\pi}_{\text{Potts}}(\sigma) = \frac{\tilde{w}_{\text{Potts}}(\sigma)}{\tilde{Z}_{\text{Potts}}}, \quad \tilde{w}_{\text{Potts}}(\sigma) = \lambda^{-a(\sigma) - h(\sigma)}, \quad \tilde{Z}_{\text{Potts}} = \sum_{\sigma' \in \tilde{\Omega}^{\rho N}} \tilde{w}_{\text{Potts}}(\sigma') \quad (8)$$

where  $\lambda > 0$ ,  $h(\sigma)$  is the number of heterogeneous edges in the configuration  $\sigma$ , and  $a(\sigma)$  is the number of edges between occupied and unoccupied sites in  $G_\Delta$ .

We prove the following theorem that establishes aggregation and alignment for appropriate settings of the parameters.

► **Theorem 21.** Fix  $\rho < \frac{1}{3}$  and assume that there will always be exactly  $\rho N$  filled sites on the lattice. For any  $\delta > 0$  and  $\alpha > 1$ , there exists a  $\lambda_0 = \lambda_0(q, \rho, \alpha, \delta)$  such that for all  $\lambda > \lambda_0$ , with probability  $1 - \zeta^{\sqrt{N}}$  for some constant  $\zeta = \zeta(q, \rho, \alpha, \delta, \lambda) < 1$ , there exists a region  $\mathcal{R} \subseteq V(G_\Delta)$ , where

1. There is an orientation  $\theta \in \{0, 1, \dots, q-1\}$  where the number of filled sites with orientation  $\theta$  in  $\mathcal{R}$  is at least  $(1 - \delta)|\mathcal{R}|$ .
2. The number of filled sites not in  $\mathcal{R}$  is at most  $\delta(N - |\mathcal{R}|)$
3. The boundary length of  $\mathcal{R}$  is at most  $\alpha \cdot bd_{\min}(\rho N)$ .

Due to space limitations, we relegate the main details of the proofs to Appendix A.

---

## References

- 1 Logan E. Beaver and Andreas A. Malikopoulos. An overview on optimal flocking. *Annual Reviews in Control*, 51:88–99, January 2021. doi:10.1016/j.arcontrol.2021.03.004.
- 2 T. Bodineau, D. Ioffe, and Y. Velenik. Rigorous Probabilistic Analysis of Equilibrium Crystal Shapes. *Journal of Mathematical Physics*, 41(3):1033–1098, March 2000. arXiv:math/9911106. doi:10.1063/1.533180.

- 3 Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on  $\mathbb{Z}^d$  at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 738–751, New York, NY, USA, June 2020. ACM. doi:10.1145/3357713.3384271.
- 4 Christian Borgs, Jennifer Chayes, Jeff Kahn, and László Lovász. Left and right convergence of graphs with bounded degree. *arXiv*, January 2010. arXiv:1002.0115.
- 5 Sarah Cannon, Joshua J. Daymude, Cem Gökmen, Dana Randall, and Andréa W. Richa. A Local Stochastic Algorithm for Separation in Heterogeneous Self-Organizing Particle Systems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:22, Dagstuhl, Germany, 2019. ISSN: 1868-8969. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.54.
- 6 Sarah Cannon, Joshua J. Daymude, Cem Gokmen, Dana Randall, and Andréa W. Richa. A Local Stochastic Algorithm for Separation in Heterogeneous Self-Organizing Particle Systems. *arXiv*, June 2019. URL: <http://arxiv.org/abs/1805.04599>, arXiv:1805.04599.
- 7 Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A Markov Chain Algorithm for Compression in Self-Organizing Particle Systems. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 279–288, New York, NY, USA, July 2016. Association for Computing Machinery. doi:10.1145/2933057.2933107.
- 8 Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A Markov Chain Algorithm for Compression in Self-Organizing Particle Systems. *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing - PODC '16*, pages 279–288, 2016. arXiv:1603.07991.
- 9 L. Chayes, R. Kotecký, and S. B. Shlosman. Aggregation and intermediate phases in dilute spin systems. *Commun.Math. Phys.*, 171(1):203–232, July 1995. doi:10.1007/BF02103776.
- 10 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by Programmable Particles. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 615–681. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-11072-7\_22.
- 11 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Amoebot - a new model for programmable matter. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, SPAA '14, pages 220–222, New York, NY, USA, June 2014. Association for Computing Machinery. doi:10.1145/2612669.2612712.
- 12 Hugo Duminil-Copin and Stanislav Smirnov. The connective constant of the honeycomb lattice equals  $\sqrt{2 + \sqrt{2}}$ . *Annals of Mathematics*, 175(3):1653–1665, 2012. Publisher: Annals of Mathematics. URL: <https://www.jstor.org/stable/23234646>.
- 13 J. W. Essam and C. Tsallis. The Potts model and flows. I. The pair correlation function. *J. Phys. A: Math. Gen.*, 19(3):409–422, February 1986. Publisher: IOP Publishing. doi:10.1088/0305-4470/19/3/022.
- 14 Sacha Friedli and Yvan Velenik. Cluster Expansion. In *Statistical Mechanics of Lattice Systems: A Concrete Mathematical Introduction*, pages 232–261. Cambridge University Press, Cambridge, 2017. doi:10.1017/9781316882603.006.
- 15 W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. Publisher: [Oxford University Press, Biometrika Trust]. doi:10.2307/2334940.
- 16 Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov-Sinai theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1009–1020, New York, NY, USA, June 2019. Association for Computing Machinery. doi:10.1145/3313276.3316305.

- 17 Dmitry Ioffe and Roberto H. Schonmann. Dobrushin-kotecký-Shlosman Theorem up to the Critical Temperature. *Comm Math Phys*, 199(1):117–167, December 1998. doi:10.1007/s002200050497.
- 18 A. Jadbabaie, Jie Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003. Conference Name: IEEE Transactions on Automatic Control. doi:10.1109/TAC.2003.812781.
- 19 Matthew Jenssen and Will Perkins. Independent sets in the hypercube revisited. *Journal of the London Mathematical Society*, 102(2):645–669, 2020. doi:10.1112/jlms.12331.
- 20 R. Kotecký and D. Preiss. Cluster expansion for abstract polymer models. *Comm. Math. Phys.*, 103(3):491–498, 1986. Publisher: Springer-Verlag. URL: <http://projecteuclid.org/euclid.cmp/1104114796>.
- 21 Shengkai Li, Bahnisikha Dutta, Sarah Cannon, Joshua J. Daymude, Ram Avinery, Enes Aydin, Andréa W. Richa, Daniel I. Goldman, and Dana Randall. Programming active cohesive granular matter with mechanically induced phase changes. *Science Advances*, 7(17):eabe8494, April 2021. Publisher: American Association for the Advancement of Science Section: Research Article. doi:10.1126/sciadv.abe8494.
- 22 Nancy A. Lynch. *Distributed Algorithms*. Elsevier, April 1996.
- 23 Joseph E. Mayer. The Statistical Mechanics of Condensing Systems. I. *J. Chem. Phys.*, 5(1):67–73, January 1937. Publisher: American Institute of Physics. doi:10.1063/1.1749933.
- 24 Joseph E. Mayer. The Theory of Ionic Solutions. *J. Chem. Phys.*, 18(11):1426–1436, November 1950. Publisher: American Institute of Physics. doi:10.1063/1.1747506.
- 25 Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21(6):1087–1092, June 1953. Publisher: American Institute of Physics. doi:10.1063/1.1699114.
- 26 R. A. Minlos and Ja G. Sinaĭ. The phenomenon of phase separation at low temperatures in some lattice models of a gas. I. *Math. USSR Sb.*, 2(3):335, April 1967. Publisher: IOP Publishing. doi:10.1070/SM1967v002n03ABEH002345.
- 27 Sarah Miracle, Dana Randall, and Amanda Pascoe Streib. Clustering in Interfering Binary Mixtures. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science, pages 652–663, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-22935-0\_55.
- 28 R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, March 2006. Conference Name: IEEE Transactions on Automatic Control. doi:10.1109/TAC.2005.864190.
- 29 G. Ortiz, E. Cobanera, and Z. Nussinov. Dualities and the phase diagram of the p-clock model. *Nuclear Physics B*, 854(3):780–814, January 2012. doi:10.1016/j.nuclphysb.2011.09.012.
- 30 C.-E. Pfister. Interface free energy or surface tension: definition and basic properties. *arXiv*, 2009. arXiv:0911.5232.
- 31 S. A. Pirogov and Ya. G. Sinai. Phase diagrams of classical lattice systems. *Theor Math Phys*, 25(3):1185–1192, December 1975. doi:10.1007/BF01040127.
- 32 S. A. Pirogov and Ya. G. Sinai. Phase diagrams of classical lattice systems continuation. *Theor Math Phys*, 26(1):39–49, January 1976. doi:10.1007/BF01038255.
- 33 Wenquan Qin, Shucong Lin, Xuan Chen, Jian Chen, Lei Wang, Hongpeng Xiong, Qinxie Xie, Zhaohui Sun, Xiujun Wen, and Cai Wang. Food Transport of Red Imported Fire Ants (Hymenoptera: Formicidae) on Vertical Surfaces. *Scientific Reports*, 9(1):3283, March 2019. Number: 1 Publisher: Nature Publishing Group. doi:10.1038/s41598-019-39756-4.
- 34 Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, August 1987. Association for Computing Machinery. doi:10.1145/37401.37406.

- 35 Herbert Robbins. A Remark on Stirling’s Formula. *The American Mathematical Monthly*, 62(1):26–29, 1955. Publisher: Mathematical Association of America. doi:10.2307/2308012.
- 36 Herbert G. Tanner, Ali Jadbabaie, and George J. Pappas. Flocking in Fixed and Switching Networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, May 2007. Conference Name: IEEE Transactions on Automatic Control. doi:10.1109/TAC.2007.895948.
- 37 Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel Type of Phase Transition in a System of Self-Driven Particles. *Phys. Rev. Lett.*, 75(6):1226–1229, August 1995. Publisher: American Physical Society. doi:10.1103/PhysRevLett.75.1226.
- 38 Tamás Vicsek and Anna Zafeiris. Collective motion. *Physics Reports*, 517(3):71–140, August 2012. doi:10.1016/j.physrep.2012.03.004.
- 39 José D. Villa. Swarming Behavior of Honey Bees (Hymenoptera: Apidae) in Southeastern Louisiana. *Ann Entomol Soc Am*, 97(1):111–116, January 2004. Publisher: Oxford Academic. doi:10.1603/0013-8746(2004)097[0111:SBOHBH]2.0.CO;2.
- 40 F. Y. Wu. The Potts model. *Rev. Mod. Phys.*, 54(1):235–268, January 1982. Publisher: American Physical Society. doi:10.1103/RevModPhys.54.235.
- 41 Hai-Tao Zhang, Chao Zhai, and Zhiyong Chen. A General Alignment Repulsion Algorithm for Flocking of Multi-Agent Systems. *IEEE Transactions on Automatic Control*, 56(2):430–435, February 2011. Conference Name: IEEE Transactions on Automatic Control. doi:10.1109/TAC.2010.2089652.

## **A** Details for Aggregation and Alignment in General SOPS

In the general SOPS setting, we can treat the problem as a  $q + 1$ -state Potts model on  $G_\Delta$  with  $q + 1$  orientations  $\{-1, 0, 1, \dots, q - 1\}$  in which the number of sites assigned  $-1$  is fixed to be exactly  $(1 - \rho)N$ , where  $N = |V(G_\Delta)|$ . In other words, sites of the lattice are no longer filled or unfilled, but are instead assigned one of  $q + 1$  orientations with the special spin  $-1$  assigned to unoccupied lattice sites. We refer to any edge between particles of differing orientations as “heterogeneous edges,” including those assigned the special orientation  $-1$ .

We again use a Peierls argument to show that for sufficiently large  $\lambda$ , the configuration will compress and one of the  $q$  orientations will dominate, with high probability. This proof is an adaptation of the bridging argument used for separation in [5, 6] and thus follows their arguments very closely. The following sections build up to a proof of Theorem 21.

We observe that the result of Theorem 21 will imply both alignment and aggregation (for some values of  $\alpha$  and  $\delta$ ) as given in Definitions 10 and 20. The key component of our proof is the construction of a  $\delta$ -bridge system ( $\delta \in (0, 1)$  is a positive constant) for each configuration in  $\tilde{\Omega}^{\rho N}$ . Recall that a bridge system is a connected network of the long contours of a configuration  $\sigma$ , that is used to “remove” long contours in the Peierls argument to show that they are unlikely. It will also be used to define the region  $\mathcal{R}$  required for Theorem 21.

Let  $E_{\text{wrap}}$  be the set of edges on  $G_\square$  corresponding to the edges on  $G_\Delta$  that wrap around the torus. Thus  $|E_{\text{wrap}}| = 2\sqrt{N} - 1$ . In a setting with more than three possible orientations, regions of differing orientations are divided up by networks of contours rather than closed walks separating two different orientations. We call these contour networks *complex contours*. Formally, a complex contour refers to a connected subgraph of  $G_\square$  of minimum degree at least 2. For a given configuration  $\sigma \in \tilde{\Omega}^{\rho N}$ , the set of edges  $\mathcal{C}$  on  $G_\square$  corresponding to its heterogeneous edges will be a union of complex contours. The complex contours of  $\sigma$  thus refers to the edge sets of connected components of the subgraph induced by  $\mathcal{C}$  in  $G_\square$ .

We now define a bridge system  $(B, I, \Theta)$  where the set  $I$  represents the complex contours in the bridge system,  $B$  represents the bridges used to connect these complex contours, and  $\Theta$  is a mapping that assigns an orientation to each of the components formed after removing the edges of  $G_\Delta$  corresponding to the edges in  $I$ .



► **Definition 22** (Bridge Systems). Fix  $\delta > 0$ . Consider a tuple  $(B, I, \Theta)$ , where  $B$  and  $I$  are subsets of  $E(G_\square)$  and  $\Theta : V(G_\Delta) \rightarrow \{-1, 0, 1, \dots, q-1\}$  is a function assigning each vertex an orientation or the value  $-1$  (which we will use to represent vacant sites). We say  $(B, I, \Theta)$  is a  $\delta$ -bridge system if:

1. The subgraph induced in  $G_\square$  by  $I$  has no vertex of degree less than 2. Practically,  $I$  represents a union of complex contours that subdivides  $G_\Delta$  into regions.
2. The subgraph induced in  $G_\square$  by  $B \cup I \cup E_{\text{wrap}}$  is connected and has no vertex of degree less than 2.
3.  $B \cap I = \emptyset$  and  $|B| \leq \frac{1-\delta}{2\delta}|I|$
4. For any two neighboring sites  $u, v \in G_\Delta$ ,  $\Theta(u) = \Theta(v)$  if and only if the dual edge corresponding to  $\{u, v\}$  is not in  $I$ .

Consider a set of edges  $I$ , that is a union of the edge sets of complex contours. Let  $\sigma$  be a configuration in  $\tilde{\Omega}^{\rho N}$ . We say a complex contour  $C$  of  $\sigma$  is *bridged* (by  $I$ ) if  $C \subseteq I$ . We say a site  $v$  is *bridged* (by  $I$ ) if there is a path over  $G_\Delta$  using only sites of the same orientation (including  $-1$ ) in  $\sigma$  as  $v$  to a site incident to an edge in  $I$ . Consider a region  $R \subseteq V(G_\Delta)$  that is connected as an induced subgraph of  $G_\Delta$ . We call  $R$  a *bridged region* if  $\text{bd}(R) \subseteq I$  and a *minimal bridged region* if there is no bridged region  $R'$  where  $R' \subsetneq R$ . Notably, the edge set  $I$  partitions  $V(G_\Delta)$  into minimal bridged regions.

► **Definition 23** (Bridge System for a Configuration). Fix  $\delta > 0$  and a configuration  $\sigma \in \tilde{\Omega}^{\rho N}$ . We say a tuple  $(B, I, \Theta)$  is a  $\delta$ -bridge system for a configuration  $\sigma$  if

1. Each minimal bridged region  $R$  by  $(B, I, \Theta)$  contains at most  $\delta|R|$  unbridged particles.
2. No complex contour  $C$  of  $\sigma$  meets any edge in  $B \cup I \cup E_{\text{wrap}}$ . Formally, the edge-induced subgraphs  $G_\Delta[C]$  and  $G_\Delta[B \cup I \cup E_{\text{wrap}}]$  do not share any vertices.
3. For each minimal bridged region  $R$ ,  $\Theta(v)$  must have the same value for every site  $v \in R$  and this value  $\Theta(v)$  must correspond to the orientation in  $\sigma$  of some bridged particle in  $R$ .

► **Definition 24** (Orientation of a Minimal Bridged Region). Given a  $\delta$ -bridge system  $(B, I, \Theta)$  for a configuration  $\sigma \in \tilde{\Omega}^{\rho N}$ . We can associate with each minimal bridged region  $R$  of  $I$  an orientation  $y_R \in \{-1, 0, 1, \dots, q-1\}$ .

To determine  $y_R$ , we denote by  $R^*$  the set of sites  $v \in R$  with a path over  $G_\Delta$  using only sites of the same orientation in  $\sigma$  as  $v$  to a site incident to an edge in  $\text{bd}(R)$ . We note that  $\text{bd}(R) \subseteq I$  and the edges  $B \cup I \cup E_{\text{wrap}}$  connect the components of  $\text{bd}(R)$  in  $G_\square$ . This implies that every vertex in  $R^*$  must have the same orientation in  $\sigma$ , as any contour  $C$  between regions of differing orientations in  $R^*$  must intersect  $B \cup I \cup E_{\text{wrap}}$ , implying that  $C$  also must be included in the set  $I$ , allowing us to subdivide  $R$ , contradicting its minimality. The orientation  $y_R$  of  $R$  is thus defined to be the common orientation of the sites of  $R^*$ .

Thus, for each minimal bridged region  $R$  with orientation  $y_R$ , we must have  $\Theta(v) = y_R$  for all  $v \in R$ . The proofs of the Lemmas will be given in the long version of the paper.

Our next step is to associate with each  $\sigma \in \tilde{\Omega}^{\rho N}$  a  $\delta$ -bridge system.

► **Lemma 25.** For each  $\sigma \in \tilde{\Omega}^{\rho N}$  and  $\delta \in (0, 1)$ , we can construct a  $\delta$ -bridge system  $\mathcal{B}_\delta(\sigma) = (B_\delta(\sigma), I_\delta(\sigma), \Theta_\delta(\sigma))$ .

Without reference to any specific configuration in  $\tilde{\Omega}^{\rho N}$ , we use the connectedness requirement of bridge systems to compute an upper bound on the number of bridge systems that is exponential on  $|I|$ . This is important as the Peierls argument “removes” the heterogeneous edges in  $I$ , which gives an improvement in weight of a similar order of growth.

► **Lemma 26.** *The number of  $\delta$ -bridge systems  $(B, I, \Theta)$  where  $|I| = \ell$  is at most  $7 \cdot 6^{2\sqrt{N}-1} \cdot (3(q+1))^{\frac{1+\delta}{2\delta}\ell}$ .*

Assuming  $\delta \in (0, \rho)$ , we define  $\tilde{\Omega}_\ell^{\rho N} := \{\sigma \in \tilde{\Omega}^{\rho N} : |I_\delta(\sigma)| = \ell\}$ , where  $I_\delta(\sigma)$  is comes from the  $\delta$ -bridge system constructed for  $\sigma$ . Also, let  $\tilde{\Omega}^{\leq \delta N}$  be the the set of configurations over  $G_\Delta$  where at least  $(1 - \delta)N$  sites have orientation  $-1$  (this corresponds to empty sites in our model). Note that  $\tilde{\Omega}^{\leq \delta N} \not\subseteq \tilde{\Omega}^{\rho N}$ . For the Peierls argument, we define two functions,  $f_\ell^1 : \tilde{\Omega}_\ell^{\rho N} \rightarrow \tilde{\Omega}^{\leq \delta N}$  and  $f^2 : \tilde{\Omega}^{\leq \delta N} \rightarrow \tilde{\Omega}^{\rho N}$ . The function  $f_\ell^1$  is used to erase the heterogeneous edges in  $I$ , creating a configuration of significantly higher weight, though not one with  $\rho N$  particles. To fix this, a second function,  $f^2$  is used to restore the number of particles back to  $\rho N$ . This way,  $f^2 \circ f_\ell^1$  maps each  $\sigma$  in  $\tilde{\Omega}_\ell^{\rho N}$  to a valid configuration with exactly  $\rho N$  filled sites. The definitions of  $f_\ell^1$  and  $f^2$  are given in the full version of the paper.

As the bridge system with just a polynomial amount of additional information can be used to reconstruct  $\sigma$  from  $f^2 \circ f_\ell^1$ , our upper bound on the number of bridge systems can be used to upper bound  $|(f^2 \circ f_\ell^1)^{-1}(\tau)|$  for any  $\tau$  in the image of  $f^2 \circ f_\ell^1$ . This allows us to prove the following Lemma:

► **Lemma 27.** *Fix  $\rho < \frac{1}{3}$ , any  $\alpha > 1$ ,  $\delta \in (0, \min\{\rho, 1 - \frac{1}{\alpha^2}\})$  and  $\lambda > \lambda_0(q, \rho, \alpha, \delta)$  sufficiently large, where:*

$$\lambda_0(q, \rho, \alpha, \delta) := \left( (3(q+1))^{\alpha \frac{1+\delta}{2\delta}} 36^{\frac{1}{4\sqrt{3\rho}}} \right)^{\frac{1}{\alpha - \frac{1}{\sqrt{1-\delta}}}}.$$

Denote by  $\tilde{\Omega}_{\geq \alpha \cdot bd_{\min}(\rho N)}^{\rho N}$  the set of configurations  $\sigma$  where  $|I_\delta(\sigma)| \geq \alpha \cdot bd_{\min}(\rho N)$ , where  $bd_{\min}(k)$  is the minimum possible boundary length of a region of  $k \in \mathbb{N}$  particles. Then there exists a constant  $\zeta = \zeta(q, \rho, \alpha, \delta, \lambda) < 1$  such that  $\tilde{\pi}_{\text{Potts}}(\tilde{\Omega}_{\geq \alpha \cdot bd_{\min}(\rho N)}^{\rho N}) < \zeta^{\sqrt{N}}$  for all sufficiently large values of  $N$ .

As the bridge system with just a polynomial amount of additional information can be used to reconstruct  $\sigma$  from  $f^2 \circ f_\ell^1$ , our upper bound on the number of bridge systems can be used to upper bound  $|(f^2 \circ f_\ell^1)^{-1}(\tau)|$  for any  $\tau$  in the image of  $f^2 \circ f_\ell^1$ . This allows us to prove the following Lemma:

The use of Lemma 27 along with some results on the minimum possible boundary lengths of regions of  $k$  particles allows us to show that there will exist a low perimeter region dominated by a single color, allowing us to prove Theorem 21.



# Tight Chernoff-Like Bounds Under Limited Independence

Maciej Skorski 

University of Luxembourg, Luxembourg

---

## Abstract

This paper develops sharp bounds on moments of sums of  $k$ -wise independent bounded random variables, under constrained average variance. The result closes the problem addressed in part in the previous works of Schmidt et al. and Bellare, Rompel. The work also discusses other applications of independent interests, such as asymptotically sharp bounds on binomial moments.

**2012 ACM Subject Classification** Mathematics of computing → Statistical paradigms; Mathematics of computing → Probabilistic inference problems; Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** concentration inequalities, tail bounds, limited independence,  $k$ -wise independence

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.15

**Category** RANDOM

**Related Version** *Full Version:* [https://www.researchgate.net/publication/362034749\\_Tight\\_Chernoff-Like\\_Bounds\\_under\\_Limited\\_Independence](https://www.researchgate.net/publication/362034749_Tight_Chernoff-Like_Bounds_under_Limited_Independence)

**Funding** The research supported by the FNR grant C17/IS/11613923.

**Acknowledgements** The author thanks the reviewers of RANDOM'22 for insightful comments.

## 1 Introduction

### 1.1 Motivation

Consider sums of random variables, possibly differently distributed. What can be said about the probability distribution, particularly the tails, if we only assume  *$k$ -wise independence*, that is any  $k$  of the  $n$  summands are independent? Such a dependency condition is an appealing concept studied in a broad class of problems related to pseudorandomness, including hashing, random graphs, random projections and circuits [34, 3, 28, 6]; specifically, concentration results under  $k$ -wise independence find important applications including (but not limited to) constructions of pseudorandom generators [25], load balancing [29, 50], derandomization [49, 23, 14], streaming algorithms [36] and cryptography [7, 5, 17, 4].

At first glance, the problem seems well addressed by concentration inequalities, such as the classical bounds due to Bernstein, Chernoff, Hoeffding, Bennet [10, 15, 24, 8] or their modern sub-gaussian or sub-gamma generalizations [13] (obtained from moment generating functions); at the very least one may hope to utilize more exotic moment bounds such as Rosenthal-type inequalities [47, 12] or more general frameworks [32]. However, the exponential moment methods are inadequate for limited dependence, whereas moment methods are hard to apply for sums of heterogenic terms. The state-of-the-art is held by the two influential works [50, 7] which resort to direct moment calculations, offering bounds for certain parameter regimes.

The goal of the current paper is to establish *sharp moment bounds* for sums of bounded  $k$ -wise independent variables, strengthening the state-of-art results [50, 7]. As in prior work, we assume that the summands are bounded and the sum variance is known. Formally:



© Maciej Skorski;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 15; pp. 15:1–15:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Let  $S = \sum_{i=1}^n X_i$  be a sum of  $k$ -wise independent random variables, possibly differently distributed. Suppose that a)  $X_i \in [-1, 1]$  and b) the average variance is  $\frac{1}{n} \sum_{i=1}^n \mathbb{V}[X_i] = \sigma^2$ . What is the *best* bound on moments of  $S - \mathbb{E}S$ ?

Answering this question obviously gives desired Chernoff-like tail bounds, via an application of Markov's inequality. This approach, called the moment method, is the state-of-art technique of establishing tail bounds [13, 32], even superior to the exponential moment method [44], so it should give us as much as we can get.

## 1.2 Our Contribution

The novelty of this work has the following aspects

- sharp bounds are found for *all* parameter regimes, which improves upon prior works
- some elegant techniques, novel in this context, are demonstrated; particularly the powerful method of *symmetrization* from high-dimensional probability [51] and *extreme inequalities* for symmetric polynomials [48]<sup>1</sup>
- other applications, in particular sharp bounds for moments of binomial distribution

We now move to present our results, adopting the following notation: for two expressions  $A, B$  we write  $A \lesssim B$  when  $A \leq K \cdot B$  for some absolute constant  $K$ , and  $A \simeq B$  when the inequality holds in both direction. By  $\|Z\|_d = (\mathbb{E}|Z|^d)^{1/d}$  we denote the  $d$ -th norm of a random variable  $Z$ . By  $\mathbb{E}Z$  and  $\mathbb{V}[Z]$  we denote, respectively, the mean and variance of  $Z$ .

### 1.2.1 Sharp Bounds for $k$ -wise Independent Sums

The theorem below gives the complete answer to the posed problem.

► **Theorem 1** (Moments of  $k$ -wise Independent Sums). *Consider random variables  $(X_i)_{i=1}^n$  satisfying the following conditions*

(a)  $(X_i)_i$  are  $k$ -wise independent ( $k \geq 2$ ) and  $|X_i - \mathbb{E}X_i| \leq 1$

(b)  $\sum_{i=1}^n \mathbb{V}[X_i] \leq n\sigma^2$  (the sum variance bounded)

Then for  $S = \sum_{i=1}^n X_i$  and any positive even integer  $d \leq k$  we have:

$$\max_{(X_i)} \|S - \mathbb{E}S\|_d \simeq M(n, \sigma^2, d) = \begin{cases} \sqrt{dn\sigma^2} & \log(d/n\sigma^2) < \max(d/n, 2) \\ \frac{d}{\log(d/n\sigma^2)} & \max(d/n, 2) \leq \log(d/n\sigma^2) \leq d, \\ (n\sigma^2)^{1/d} & d < \log(d/n\sigma^2) \end{cases} \quad (1)$$

where the maximum is over all r.v.s.  $(X_i)_i$  satisfying the conditions (a) and (b).

Moreover, the maximal value is realized (up to a constant) when

$$X_i \sim B - B', \quad B, B' \sim^{iid} \text{Bern}(p), p = \frac{1}{2}(1 - \sqrt{1 - 2\sigma^2}). \quad (2)$$

► **Remark 2** (Value of  $k$ ). In applications value of  $k$  should be possibly big, so that we can use as high moments  $d$  as possible. For example, some cryptographic applications use  $k = 80$  [17].

<sup>1</sup> The prior work [50] actually recognized usefulness of symmetry, but was not able to exploit it in the case of general  $[-1, 1]$ -valued random variables.

► **Remark 3 (Formula Regimes).** The formulas may look a little exotic, especially to a reader familiar with previous works. The reason is that the novel bounds above are sharp and capture some non-standard behaviors. The branch with  $\sqrt{dn\sigma^2}$  should be familiar, as this is the  $d$ -th moment of gaussian distribution with variance  $n\sigma^2$ ; in this regime it would produce the tail  $\Pr[|S - \mathbb{E}S| > t] \leq e^{-\Omega(t^2/n\sigma^2)}$ . The branch with  $d/\log(d/n\sigma^2)$  gives the behavior slightly faster than this of the exponential distribution; it would produce the tail  $\Pr[|S - \mathbb{E}S| > t] \leq e^{-\omega(t)}$ . Finally, the branch with  $(n\sigma^2)^{1/d}$  resembles the behavior of a distribution bounded by  $\sigma^2$ .

► **Remark 4 (Odd values of  $d$ ).** It can be shown that the same upper bounds apply when  $d > 2$  is odd, due to interpolation inequalities [9].

► **Remark 5 (Explicit Tail Bounds).** For  $M$  as in Equation (1) we obtain the following tail bound (which depends on the parameters regime), for any  $t > 0$

$$\Pr[|S - \mathbb{E}S| > t] \leq O(M(n, \sigma^2, d)/t)^d. \quad (3)$$

For  $t = cM(n, \sigma^2, d)$  with an appropriate constant  $c$ , we obtain the tail of  $2^{-\Omega(d)}$ .

## 1.2.2 Techniques

We show that for even  $d \leq k$  we can assume in addition c) full independence and d) full symmetry of the summands, leveraging *symmetrization* [51]. Then we proceed in two steps.

### 1.2.2.1 Characterization of Extreme Distribution

First, we characterize “worst-case” distributions  $X_i$  that maximize  $\|\sum_i X_i\|_d$ . This result is the core of our approach and of broader interest, we thus present it as the standalone lemma.

► **Lemma 6 (IID Majorization of Symmetric Sums).** *Let  $(Z_i)_{i=1}^n$  be independent symmetric random variables with values in  $[-1, 1]$  with average variance  $\sigma^2 = \frac{1}{n} \sum_{i=1}^n \mathbb{V}[Z_i]$ . Then for any positive even integer  $d$  we have that*

$$\left\| \sum_i Z_i \right\|_d \leq \left\| \sum_i Z'_i \right\|_d \quad (4)$$

where  $Z'_i$  are independent and identically distributed as

$$Z'_i \sim \begin{cases} +1 & \text{w.p. } \sigma^2/2 \\ 0 & \text{w.p. } 1 - \sigma^2 \\ -1 & \text{w.p. } \sigma^2/2 \end{cases}. \quad (5)$$

► **Remark 7 (Interpretation).** Observe that  $\mathbb{V}[Z'_i] = 2 \cdot \sigma^2/2 = \sigma^2$ , thus the theorem essentially says that moments of the sum  $\sum_i Z_i$  are maximized for  $Z_i$  that are iid with the distribution (5). This three-point distribution is *extreme*, in the sense that it pushes as much mass as possible towards the edge of the interval constraint. This behavior may look intuitive, but we should beware of such intuitions as even for simple symmetric problems whether the maximizer’s behavior is “push to boundary” or “pull to the middle” may not be that intuitive, depending on Schur convexity properties of the optimized expression [21]; to be specific the problems of maximization of  $\sum_{i \neq j} p_i p_j$  and  $\sum_i p_i^2$  have quite different behavior. Our proof requires some non-trivial facts about multivariate symmetric polynomials.

► **Remark 8 (Proof Techniques).** The symmetry assumption is crucial, and makes it possible to greatly simplify the multinomial expansion of the moment formula. We manage to regroup expressions and see them as positive combinations of elementary symmetry polynomials; then specialized inequalities from the theory of symmetric functions [39] come then to the rescue, allowing for proving that our extreme distribution is indeed the maximizer.

### 1.2.2.2 Closed-Form Bounds for Extreme Distributions

In addition to characterizing the worst-case behavior, we give the closed-form formula for the bound in Lemma 6. As we will see later, this is also a fact of broader interest; for example, we use it to derive bounds for binomial moments which are sharp in all parameter regimes.

► **Corollary 9 (Best bounds for IID).** *For independent ( $d$ -wise independent) symmetric  $Z_i$  with values  $\{-1, 0, 1\}$  and variance  $\sigma^2$  and positive even integer  $d$  we have*

$$\left\| \sum_{i=1}^n Z_i \right\|_d \simeq \begin{cases} \sqrt{dn\sigma^2} & \log(d/n\sigma^2) < \max(d/n, 2) \\ \frac{d}{\log(d/n\sigma^2)} & \max(d/n, 2) \leq \log(d/n\sigma^2) \leq d \\ (n\sigma^2)^{1/d} & d < \log(d/n\sigma^2) \end{cases} \quad (6)$$

► **Remark 10 (Proof techniques).** The proof requires some effort to compute moments for Equation (5). Loosely speaking, we leverage the specific form of the distribution to obtain regular combinatorial patterns in multinomial expansions. We then obtain an explicit formula, being a weighted sum of binomial-like expressions which involve  $n$ ,  $d$  and  $\sigma$ . Establishing the order of growth, with the help of some calculus, completes the proof.

## 1.3 Related Work

There are many bounds which cover different models of dependencies among random variables, for example Janson’s correlation inequality [27] which has become very popular in analyses of random graphs [18], or the theory of negative dependence [11] best known from applications to various “balls and bins” problems [19]. However the focus of this paper is on *k-wise independence*, in which the state-of-art bounds are due to Schmidt et al. [50] and Bellare and Rompel [7], derived in the essentially same setup as ours (moment bounds under the variance constraint). These bounds, although useful for many applications, hold only in certain regimes and are not sharp in general; when discussing our applications we will show that in most cases they are inferior to Theorem 1.

It in our work we consider the most natural variance constraint, following prior works [50, 7]. However, one might consider more exotic structural assumptions; recently there has been an attempt, limited only to  $k = 2$ , to characterize worst bounds by exploring the whole sequence (rather than the sum variance) of Bernoulli parameters of  $X_i$  [45].

Regarding the established concentration bounds, we will see that known inequalities actually imply stronger bounds than those developed by Schmidt et al. and Bellare, Rompel [50, 7]. However, even with the use of state-of-art moment inequalities [13] we were not able to recover the sharp bounds from our main result and the characterization from Lemma 6. The key challenge is to precisely characterize the worst-case behavior, while allowing *differently distributed* random variables.

An interesting way of attacking the problem, possibly working for much more exotic constraints, may be to formally follow the presented idea of majorization and establish formally some Schur-convexity properties; this approach has been successfully applied in the past to many other problems (see [20] and follow-up works).

■ **Table 1** Bounds for moments of sum  $S = \sum_{i=1}^n X_i$  of  $d$ -wise independent random variables, where  $d \geq 2$ . As discussed, our bounds are strictly better than those of Schmidt at al., which in turn are strictly better than those of Bellare and Rompel.

| Bound on $\ S - \mathbb{E}S\ _d$                                  | Author                           | Assumptions                                                  |
|-------------------------------------------------------------------|----------------------------------|--------------------------------------------------------------|
| $\max(\sqrt{dn\sigma^2}, d/\log(d/n\sigma^2), (n\sigma^2)^{1/d})$ | <b>this paper</b>                | $n\sigma^2 = \mathbb{V}[S]$ , $ X_i - \mathbb{E}X_i  \leq 1$ |
| $\max(\sqrt{dn\sigma^2}, d)$                                      | Schmidt at al., <b>optimized</b> | $n\sigma^2 = \mathbb{V}[S]$ , $ X_i - \mathbb{E}X_i  \leq 1$ |
| $\min(\sqrt{dn}, \sqrt{dn\mu + d^2})$                             | Bellare and Rompel               | $n\mu = \mathbb{E}[S]$ , $0 \leq X_i \leq 1$                 |

## 1.4 Applications

### 1.4.1 Limited Independence: Clarifying the State-of-Art

We will demonstrate how our bounds improve on those of Schmidt at al. [50] and Bellare and Rompel [7], clarifying this way the state-of-the-art. In what follows we assume, as in our theorem, that  $\|X_i - \mathbb{E}X_i\| \leq 1$ ,  $X_i$  are  $k$ -wise independent, and  $d \leq k$  for positive even  $d$ . The best bound due to Schmidt at al. reads as (cf Eq. 10 in [50])

$$\|S - \mathbb{E}S\|_d^d \leq \sqrt{2} \cdot \cosh(\sqrt{d^3/36C}) \cdot (dC/e)^{d/2}, \quad C \geq n\sigma^2, \quad \sigma^2 = \mathbb{V}[S]/n.$$

The authors did not fully optimize the choice of  $C$ , offering a bunch of weaker corollaries instead. In order to clarify the state-of-the-art, we do this effort (see Section 3.4) obtaining

$$\|S - \mathbb{E}S\|_d \lesssim \max(\sqrt{dn\sigma^2}, d). \quad (7)$$

When  $dn\sigma^2 \geq 1$  the formula matches ours, but otherwise it is much worse: by a factor of  $\log(d/n\sigma^2)$  in the regime  $\max(2, d/n) \leq \log(d/n\sigma^2) \leq d$ , and by a factor of  $d$  in the regime  $d < \log(d/n\sigma^2)$  (which necessarily means  $n\sigma^2 < 1$ ). In applications, these factors can be a big constant or more, so derived tail bounds are worse by a big constant in the exponent.

In turn, the bound due to Bellare and Rompel [7] states that when  $X_i \in [0, 1]$

$$\|S - \mathbb{E}S\|_d \lesssim \min(\sqrt{dn}, \sqrt{dn\mu + d^2}), \quad \mu = \frac{1}{n}\mathbb{E}S. \quad (8)$$

We claim this is worse than our optimized version of Schmidt at al., in all regimes. Namely,

$$\max(\sqrt{dn\sigma^2}, d) \lesssim \min(\sqrt{dn}, \sqrt{dn\mu + d^2}).$$

Indeed, when  $d > n$  the left-hand side is at most  $n$ , while the right-hand side is at least  $n$ . When  $d \leq n$ , due to  $\mu \leq 1$  (a consequence of  $X_i \leq 1$ ) we see that  $\min(\sqrt{dn}, \sqrt{dn\mu + d^2}) \simeq \sqrt{dn\mu + d^2}$ . But we have  $\mathbb{V}[X_i] \leq \mathbb{E}X_i$ , as the consequence of  $0 \leq X_i \leq 1$ , and thus  $\sigma^2 \leq \mu$ ; this shows  $\min(\sqrt{dn}, \sqrt{dn\mu + d^2}) \gtrsim \sqrt{dn\sigma^2}$  and the claim follows.

This discussion should be of broader interest to the TCS community, as it seems that no rigorous comparison between [7] and [50] has been done before (the surveys such as [38] and application works credit both exchangeably). In Table 1 we give a readable summary.

### 1.4.2 Obtaining Previous Results form Classical Inequaliteis

Our literature search shows, perhaps surprisingly, that the optimized bounds of Schmidt at. al are actually a *simple consequence of classical inequalities*; we note that the prior works [7, 50] do not discuss the related literature on concentration bounds. The intent of this discussion is to bring those inequalities to the awareness of the wider TCS audience, particularly given the huge interest and the citation credit given to the bounds in [7, 50].

Assume that  $X_i$  are  $k$ -wide independent; recall that the event moment of order  $d \leq k$  can be calculated as if the summands were independent. More precisely, we have  $\sum_i \|X_i - \mathbb{E}X_i\|_d = \sum_i \|X'_i - \mathbb{E}X'_i\|_d$  where  $X'_i$  are distributed as  $X_i$  and independent. The tail bounds due to a century old (!) *Bernstein's inequality* [10] imply that the tail of  $S' = \sum_i X'_i$  satisfies  $\Pr[|S' - \mathbb{E}S'| > t] \leq \exp(-\Theta(\min(t^2/n\sigma^2, t)))$  for any positive  $t$ , if  $X_i$  are bounded, and  $n\sigma^2 = \sum_i \mathbb{V}[X'_i] = \sum_i \mathbb{V}[X_i]$ . By the standard tail integration formula, we find that the moments of the IID sum are  $\|S' - \mathbb{E}S'\|_d \lesssim \max(\sqrt{nd\sigma^2}, d)$ . As remarked, this matches  $\|S - \mathbb{E}S\|_d$  when  $d < k$ , so we recover the optimized (!) bounds of Schmidt at al., and implies the bounds of Bellare and Rompel. Another argument can be given by the use of *Rosenthal's inequality*, a version of which [22] implies  $\|S - \mathbb{E}S\|_d \lesssim d \cdot (\sum_{i=1}^n \mathbb{E}|X_i - \mathbb{E}X_i|^d)^{1/d} + d^{1/2} \cdot (\sum_{i=1}^n \mathbb{E}|X_i - \mathbb{E}X_i|^2)^{1/2}$ . This can be further bounded by  $\max(\sqrt{dn\sigma^2}, d)$ .

### 1.4.3 Sharp Explicit Bounds on Binomial Moments

Somewhat surprisingly, to the best of author's knowledge, there are no good closed-form estimates on moments of the binomial distribution, despite the clear demand from applications (such as the analysis of random projections [2, 26]). The sharp (up to an  $o(1)$  relative error term) tail bounds due to Littlewood [33, 37] in theory imply sharp moment estimates, but calculations lead to very difficult integrals with Kullback-Leibler divergence in the exponent. We obtain closed-form bounds for even binomial moments as a byproduct of our analysis, which are sharp in all parameter regimes. More precisely, we have

► **Corollary 11.** *Let  $S \sim \text{Binom}(n, p)$  where  $p \leq 1/2$  and  $d$  be a positive even integer. Then*

$$\|S - \mathbb{E}S\|_d \simeq \begin{cases} \sqrt{dnp} & \log(d/np) < \max(d/n, 2) \\ \frac{d}{\log(d/np)} & \max(d/n, 2) \leq \log(d/np) \leq d \\ (np)^{1/d} & d < \log(d/np) \end{cases} \quad (9)$$

This follows from the fact that the extreme variables  $Z'_i$  in our main result can be expressed as symmetrized Bernoulli distributions, namely  $Z'_i \sim B - B'$  where  $B, B' \stackrel{iid}{\sim} \text{Bern}(p)$  with  $p = \frac{1}{2}(1 - \sqrt{1 - 2\sigma^2})$ . Let  $S \sim \text{Binom}(n, p)$ . By symmetrization  $\|S - \mathbb{E}S\|_d \simeq \|S - S'\|_d$  and thus  $\|S - \mathbb{E}S\|_d \simeq \|\sum_i (B_i - B'_i)\|_d \simeq \|\sum_i Z'_i\|_d$ , thus by our result  $\|S - \mathbb{E}S\|_d$  obeys the bound as above with  $p$  replaced by  $\sigma^2$ . It remains to observe that  $\sigma^2 = 2p(1 - p)$  so  $p \leq \sigma^2 \leq 2p$ , and that the bounds above do not change by a more than a constant when  $p$  is replaced by  $p' \in [p, 2p]$ .

### 1.4.4 Exact Binomial Moments

Binomial moments can be evaluated by means of combinatorics, which yields somewhat complicated recursions [30]. Interestingly, a byproduct of Corollary 9 gives an exact formula for *symmetrized* binomials which has very simple form.

### 1.4.5 Estimating binomial-like moments

The line of research focused on estimating Renyi entropy of unknown probability distributions ([1, 43]) faces the problem of estimating moments of sum of random variables in “small variance” regime, that is when  $n\sigma^2 \ll 1$ . For example, the collision estimator requires bounds on the 4-th sum moment. This has been previously done by exploiting somewhat tedious combinatorial identities, but follows easier from Corollary 9 and Lemma 6.

## 2 Preliminaries

### 2.1 Multinomial Expansion

The *multinomial coefficient* is defined as

$$\binom{d}{\mathbf{j}} = d! / \prod_{j \in \mathbf{j}} j! \quad (10)$$

when all components of  $\mathbf{j}$  are non-negative and  $\sum_{j \in \mathbf{j}} j = d$ . We also extend this to  $\binom{d}{\mathbf{j}} = 0$  when  $\min\{j : j \in \mathbf{j}\} < 0$  or  $\sum_{j \in \mathbf{j}} j \neq d$ ; this allows for concise notation. The multinomial formula takes the form  $(\sum_i x_i)^d = \sum_{\mathbf{i}} \prod_{i \in \mathbf{i}} x_i^{i_i}$ .

► **Remark 12.** Factorials, and therefore binomial coefficients can be formally extended to negative numbers by means of Gamma function. Then indeed multinomial coefficients are zero when negative integers appear as downward arguments [31].

We will occasionally use the Stirling's formula in estimation of multinomial coefficients [40, 46]

$$(d/\ell)! \simeq \sqrt{d/\ell} \cdot (d/e\ell)^{d/\ell}. \quad (11)$$

### 2.2 Symmetrization

We will need the following facts about symmetrization (cf [51]).

► **Proposition 13** (Convex Symmetrization). *For zero-mean iid  $X, X'$  and convex  $f$*

$$\mathbb{E}f(X) \leq \mathbb{E}f(X - X'). \quad (12)$$

**Proof of Proposition 13.** By independence  $\mathbb{E}f(X - X') = \mathbb{E}_X \mathbb{E}_{X'}[f(X - X')|X]$  and by Jensen's inequality  $\mathbb{E}_{X'}[f(X - X')|X] \geq f(X - \mathbb{E}X')$ . By the zero-mean assumption  $f(X - \mathbb{E}X') = f(X)$ . The inequality follows by chaining these three bounds. ◀

► **Proposition 14** (Moments are robust under symmetrization). *For any iid random variables  $\|X\|_d \leq \|X - X'\|_d \leq 2\|X\|_d$*

**Proof of Proposition 14.** Since  $\|X\|_d = (\mathbb{E}|X|^d)^{1/d}$ , the left-hand side follows by applying Proposition 13 to  $f(u) = |u|^d$ . The right-hand side is due to the triangle inequality (Minkovski's inequality for  $L_p$  spaces). ◀

### 2.3 Symmetric Functions

The  $\ell$ -th elementary symmetric polynomial in variables  $u = (u_i)_i$  is defined as

$$\Pi_\ell(u) = \sum_{i_1 < \dots < i_\ell} u_{i_1} u_{i_2} \cdot \dots \cdot u_{i_\ell}. \quad (13)$$

The fundamental theorem on symmetric polynomials states that they generate all other symmetric polynomials (in a sense of the algebraic ring) [16]. We will need some facts about their extreme properties, which we recall below.

► **Proposition 15** (Newton Inequalities [41]). *For  $u = (u_i)_{i=1}^n$  let  $S_\ell(u) \triangleq \Pi_\ell(u) / \binom{n}{\ell}$  be the  $\ell$ -th elementary symmetric mean. Then  $S_{\ell-1}(u)S_{\ell+1}(u) \leq S_\ell(u)^2$ .*

This implies the useful inequality due to Maclaurin

► **Proposition 16** (Maclaurin's Inequality [35, 42]). *For  $u = (u_i)_{i=1}^n$  we have the inequality  $S_\ell(u)^{1/\ell} \geq S_{\ell'}(u)^{1/\ell'}$  when  $1 \leq \ell < \ell' \leq n$  (with the equality when  $u_i$  are equal).*



### 3 Proofs

#### 3.1 Proof of Lemma 6

We use the fact that  $d$  is an even integer and the multinomial formula to expand

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d = \mathbb{E} \left( \sum_{i=1}^n Z_i \right)^d = \sum_{\mathbf{j}} \binom{d}{\mathbf{j}} \mathbb{E}[Z_1^{j_1} \cdots Z_n^{j_n}] \quad (14)$$

The summation is over integer tuples  $\mathbf{j} \in \mathbb{Z}^n$  called also multiindices. Utilizing the independence assumption, we obtain

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d = \sum_{\mathbf{j}} \binom{d}{\mathbf{j}} \mathbb{E}[Z_1^{j_1}] \cdots \mathbb{E}[Z_n^{j_n}] \quad (15)$$

Since  $Z_i$  are symmetric, all odd moment vanish. Thus, we can write

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d = \sum_{\mathbf{j}} \binom{d}{2\mathbf{j}} \mathbb{E}[Z_1^{2j_1}] \cdots \mathbb{E}[Z_n^{2j_n}]. \quad (16)$$

Since  $Z_i$  are absolutely bounded by 1 and symmetric, we have  $\mathbb{E}|Z_i|^{2j} \leq \mathbb{E}|Z_i|^2 \leq \mathbb{V}[Z_i]$  for  $j \geq 1$ . Denoting  $\sigma_i^2 = \mathbb{V}[Z_i]$  we can write

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d \leq \sum_{\mathbf{j}} \binom{d}{2\mathbf{j}} \prod_{i: j_i \neq 0} \sigma_i^2. \quad (17)$$

► **Remark 17.** The equality is met when  $Z_i$  are symmetric with values in the set  $\{-1, 0, 1\}$ , as this implies  $\mathbb{E}|Z_i|^j = \mathbb{E}|Z_i|^2$ .

Let  $\|\mathbf{j}\|_0 = \#\{i : j_i \neq 0\}$  be the number of non-zero indices in the multiindex  $\mathbf{j}$ . Clearly  $\ell = \|\mathbf{j}\|_0$  can take values from 1 to  $\frac{d}{2}$  and thus

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d \leq \sum_{\ell=1}^{d/2} \underbrace{\sum_{\mathbf{j}: \|\mathbf{j}\|_0 = \ell} \binom{d}{2\mathbf{j}} \prod_{i: j_i \neq 0} \sigma_i^2}_{S_\ell}. \quad (18)$$

Note that  $S_\ell$  is multilinear of order  $\ell$  in  $u_i = \sigma_i^2$ . We claim that it equals the elementary symmetric polynomial, up to a constant multiplier (this is not clear a-priori as different weights could break the symmetry).

▷ **Claim 18.** The polynomial  $S_\ell$  is a (non-negative) multiplicity of the  $\ell$ -th elementary symmetric polynomial  $\Pi_\ell$  in variables  $\sigma_i^2$ .

*Proof of Claim.* Indeed, consider  $S_\ell$  as the weighted sum of monomials  $\prod_{i \in I} \sigma_i^2$ , where  $\|I\| = \ell$ . Every such a monomial appears with the coefficient  $c_I \triangleq \sum_{\mathbf{j}: \mathbf{j}_i \neq 0 \Leftrightarrow i \in I} \binom{d}{2\mathbf{j}}$ . Due to the symmetry of the multinomial coefficient  $\binom{d}{2\mathbf{j}}$ , namely the invariance under permuting  $\mathbf{j}$ , we claim that  $c_I$  is the same for every set  $I$ . Indeed, if  $\rho(I') = I$  for a bijection  $\rho$  then

$$c_{I'} = \sum_{\mathbf{j}: \mathbf{j}_{i'} \neq 0 \Leftrightarrow i' \in I'} \binom{d}{2\mathbf{j}} = \sum_{\mathbf{j}: \mathbf{j}_{\rho(i)} \neq 0 \Leftrightarrow \rho(i) \in I} \binom{d}{2\mathbf{j}} = \sum_{\mathbf{j}: \mathbf{j}_{\rho(i)} \neq 0 \Leftrightarrow \rho(i) \in I} \binom{d}{2\rho(\mathbf{j})} = c_I \quad (19)$$

It follows that  $S_\ell$  is a multiplicity of the  $\ell$ -th elementary symmetric polynomial (as it contains all monomials of order  $\ell$  with equal coefficients). This proves the claim. ◁



We now establish extreme properties of  $S_\ell$ . Namely

▷ **Claim 19.** The expression  $S_\ell$  is maximized, subject to the constraint that  $\sum \sigma_i^2$  is kept constant, when all  $\sigma_i$  are equal.

Proof of Claim. This follows by Maclaurin's Inequality in Proposition 16.  $\triangleleft$

Let  $\sigma^2 = \frac{1}{n} \sum_i \sigma_i^2$ , from the claim we obtain

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d \leq \sum_{\ell=1}^{d/2} \sum_{\mathbf{j}: \|\mathbf{j}\|_0 = \ell} \binom{d}{2\mathbf{j}} \cdot \sigma^{2\ell} \quad (20)$$

The right-hand side is like in Equation (18) with all  $\sigma_i$  equal, and by the remark we know that it equals  $\mathbb{E} \left| \sum_{i=1}^n Z'_i \right|^d$  if  $Z'_i$  is symmetric with values  $\{-1, 0, 1\}$  and has variance  $\sigma^2$ .

### 3.2 Proof of Corollary 9

For  $Z_i$  as in Lemma 6 the previous section derives the identity

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d = \sum_{\ell=1}^{d/2} \sum_{\mathbf{j}: \|\mathbf{j}\|_0 = \ell} \binom{d}{2\mathbf{j}} \cdot \sigma^{2\ell}. \quad (21)$$

We will further simplify this expression. Considering positive components of  $\mathbf{j}$  we obtain

$$\sum_{\mathbf{j}: \|\mathbf{j}\|_0 = \ell} \binom{d}{2\mathbf{j}} = \sum_{\ell=1}^{d/2} \sum_{i_1 < \dots < i_\ell} \sum_{\mathbf{j}: j_i \geq 1 \Leftrightarrow i \in \{i_1, \dots, i_\ell\}} \binom{d}{2\mathbf{j}} \cdot \sigma^{2\ell}. \quad (22)$$

Since the expression is invariant under permutations of  $\mathbf{j}$  we obtain

$$\sum_{\mathbf{j}: \|\mathbf{j}\|_0 = \ell} \binom{d}{2\mathbf{j}} = \sum_{\ell=1}^{d/2} \sum_{\mathbf{j}=(j_1, \dots, j_\ell) \geq 1} \binom{d}{2\mathbf{j}} \cdot \binom{n}{\ell} \quad (23)$$

where  $\binom{n}{\ell}$  counts the number of choices for  $i_1, \dots, i_\ell$ . Therefore

$$\mathbb{E} \left| \sum_{i=1}^n Z_i \right|^d = \sum_{\ell=1}^{d/2} \sum_{\mathbf{j}=(j_1, \dots, j_\ell) \geq 1} \binom{d}{2\mathbf{j}} \binom{n}{\ell} \sigma^{2\ell}. \quad (24)$$

We now estimate the moment up to constants. Since for any positive  $a_i$  we have  $(\sum_{i=1}^d a_i)^{1/d} \simeq (\max_i a_i)^{1/d}$  up to some absolute constants (in fact, constants are 1 and  $d^{1/d} \leq 2$ ), we obtain

$$\left\| \sum_{i=1}^n Z_i \right\|_d \simeq \max_{\ell=1, \dots, d/2} \left[ \underbrace{\sum_{\mathbf{j}=(j_1, \dots, j_\ell) \geq 1} \binom{d}{2\mathbf{j}} \cdot \binom{n}{\ell} \cdot \sigma^{2\ell}}_{F(\ell)} \right]^{1/d} \quad (25)$$

In the next step we estimate  $F(\ell)^{1/d}$ .

▷ **Claim 20.** We have  $F(\ell)^{1/d} \simeq \ell$ .

## 15:10 Tight Chernoff-Like Bounds Under Limited Independence

Proof. For the lower bound we can assume that  $\ell$  (and hence  $d$ ) are sufficiently big (otherwise the bound is trivial). We can also assume that  $\ell$  divides  $d$  and that  $r = \lfloor d/\ell \rfloor$  is even; otherwise we replace  $\ell$  with  $\ell'$  between  $\ell$  and  $\ell/2$  which satisfies this, use  $F(\ell) \geq F(\ell')$  and prove for  $\ell'$ . Consider the term  $2j_1 = \dots 2j_{\ell-1} = d/\ell$ , we have

$$F(\ell) \geq \binom{d}{r, \dots, r} = \frac{d!}{(r!)^\ell}.$$

Observe that  $(d!)^{1/d} \simeq d$ ,  $(r!)^{1/r} \simeq r$  and  $(q!)^{1/q} \simeq q$  (Stirling's formula). Since  $r \cdot \frac{\ell}{d} \leq 1$  we get  $(r!)^{\frac{\ell}{d}} \simeq r^{r \cdot \frac{\ell}{d}} = r$  (the relation  $\simeq$  can be raised to a bounded power). This gives

$$F(\ell)^{1/d} \gtrsim \frac{d}{r} = \ell.$$

As for the upper bound, we simply note that

$$F(\ell)^{1/d} < \left( \sum_{\mathbf{j}=(j_1, \dots, j_\ell)} \binom{d}{\mathbf{j}} \right)^{1/d} \leq (\ell^d)^{1/d} = \ell.$$

These two bounds completes the proof.  $\triangleleft$

By Claim 20 and Equation (25) we obtain the following, much simpler bound

$$\left\| \sum_{i=1}^n Z_i \right\|_d \simeq \max_{\ell=1, \dots, d/2} \left[ \ell^d \cdot \binom{n}{\ell} \cdot \sigma^{2\ell} \right]^{1/d}. \quad (26)$$

With some more effort we simplify even further. Namely, we can assume  $\ell \leq n$  as for  $\ell > n$  we have  $\binom{n}{\ell} = 0$ . By the elementary inequality  $(n/\ell)^\ell \leq \binom{n}{\ell} \leq (ne/\ell)^\ell$  we have  $\binom{n}{\ell}^{1/d} \simeq (n/\ell)^{\ell/d}$ , for  $\ell = 1 \dots d/2$ . Thus

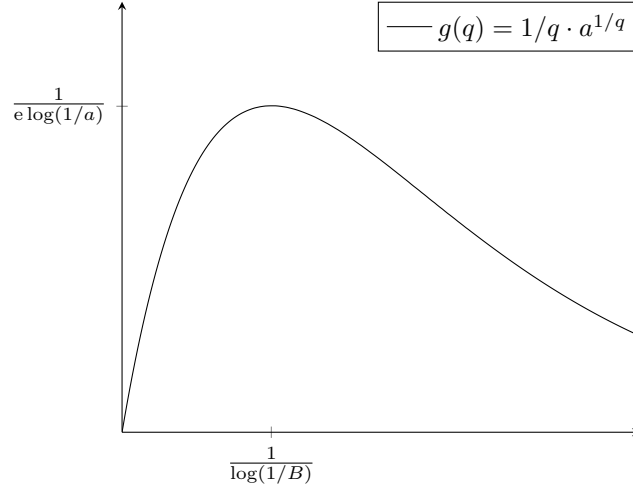
$$\begin{aligned} \left\| \sum_{i=1}^n Z_i \right\|_d &\simeq \max_{\ell=1, \dots, \min(d/2, n)} \left[ \ell^d \cdot \binom{n}{\ell} \cdot \sigma^{2\ell} \right]^{1/d} \\ &\simeq \max_{\ell=1, \dots, \min(d/2, n)} \left[ \ell \cdot (n/\ell)^{\ell/d} \cdot \sigma^{2\ell/d} \right]. \end{aligned} \quad (27)$$

Losing not more than a constant factor, we can extend the maximum to the continuous interval (the expression under maximum differs by at most a constant factor between two values of  $\ell$  that differ by one or less). Let  $q = d/\ell$ , we have the equivalent constraint  $\max(2, dn/n) \leq q \leq d$  and the maximum of  $d/q \cdot (n\sigma^2/d)^{1/q} \cdot q^{1/q}$ . Since  $q^{1/q} \simeq 1$  when  $q \geq 1$

$$\left\| \sum_{i=1}^n Z_i \right\|_d \simeq \max_{q: \max(2, d/n) \leq q \leq d} \left[ d/q \cdot (n\sigma^2/d)^{1/q} \right]. \quad (28)$$

It now suffices to analyze the auxiliary function  $g(q) \triangleq 1/q \cdot a^{1/q}$  for  $q > 0$ . The derivative test shows that it is decreasing when  $a > 1$  and has the global maximum at  $q = \log(1/a)$  with the value  $1/e \log(1/a)$  when  $a < 1$ . This behavior is illustrated on Figure 1.

Applying this fact to Equation (28), with  $a = n\sigma^2/d$ , and comparing  $\log(1/a)$  with the interval boundaries finishes the proof.



■ **Figure 1** Auxiliary function  $g$  which determines the moment behavior (for  $a < 1$ ).

### 3.3 Proof of Theorem 1

We recall the folklore fact that the moment of order  $d \leq k$  of the sum of  $k$ -wise independent r.v.s. can be computed as if they were independent. That is, let  $X'_i$  be distributed as  $X_i$  but independent. For even  $d$  we have

$$\mathbb{E} \left| \sum_i X_i \right|^d = \mathbb{E} \left| \sum_i X'_i \right|^d \quad (29)$$

which follows by applying the multinomial expansion on both sides and observing that the obtained formulas depend only on products of at most  $d$  of random variables  $X_i$  (respectively  $X'_i$ ). Without loss of generality, we can also assume that  $X_i$  are centered. Now the result follows if  $X_i$  are symmetric, by Lemma 6 and Corollary 9. If they are not symmetric, we can use the general reduction as in Proposition 14; namely, we apply the proof to  $X'_i - X''_i$  where  $X'_i, X''_i \sim^{iid} X_i$ . The moments differ by at most a factor of two. Particularly, the variance changes by a factor of 2, which has no impact on the asymptotic bounds in Equation (1). More precisely, we use the fact that  $M(n, \sigma^2, d) \simeq M(n, \sigma'^2, d)$  where  $\sigma^2/2 \leq \sigma'^2 \leq 2\sigma^2$ .

### 3.4 Optimized moment bound of Schmidt et al.

Up to a constant factor, their bound is equivalent to

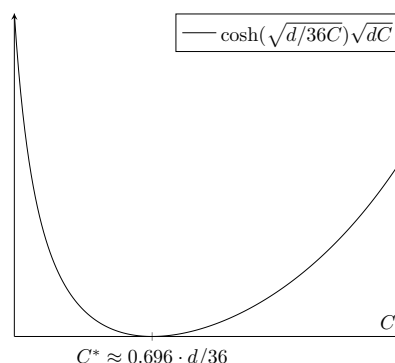
$$\|S - \mathbb{E}S\|_d \lesssim \cosh(\sqrt{d/36C})\sqrt{dC}, \quad \text{for any } C \geq \mathbb{V}[S]. \quad (30)$$

Let  $t = \sqrt{d/36C}$ , the upper bound is equivalent to  $c \cdot d \cdot \cosh(t)/t$  where  $c$  is an absolute constant. We use this to function understand the behavior of Equation (30) on  $C$ , which is as illustrated on Figure 2. Since  $C$  is constrained by  $C \geq \mathbb{V}[S]$ , the best bound is obtained for

$$C = \max(C^*, \mathbb{V}[S]), \quad (31)$$

which gives the claimed bound of

$$\|S - \mathbb{E}S\|_d \lesssim \max(\sqrt{dn\sigma^2}, d). \quad (32)$$



■ **Figure 2** The moment bound of Schmidt et al., dependency on  $C$ .

## 4 Conclusion

We have developed sharp estimates on the moments of (non necessarily identically distributed) sums of random variables, assuming the variance is constrained. This essentially closes the problem of establishing good concentration bounds, discussed in prior works. Our approach demonstrates the power of *symmetrization* technique, and is of independent interest. We also showed applications, not limited to  $k$ -wise independence.

---

## References

- 1 Jayadev Acharya, Alon Orlitsky, Ananda Theertha Suresh, and Himanshu Tyagi. Estimating rényi entropy of discrete distributions. *IEEE Transactions on Information Theory*, 63(1):38–56, 2016.
- 2 Zeyuan Allen-Zhu, Rati Gelashvili, Silvio Micali, and Nir Shavit. Sparse sign-consistent johnson–lindenstrauss matrices: Compression with neuroscience-based constraints. *Proceedings of the National Academy of Sciences*, 111(47):16872–16876, 2014.
- 3 Noga Alon and Asaf Nussboim.  $K$ -wise independent random graphs. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 813–822. IEEE, 2008.
- 4 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 826–837. IEEE, 2018.
- 5 Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 166–180. Springer, 2003.
- 6 Louay MJ Bazzi. Polylogarithmic independence can fool dnf formulas. *SIAM Journal on Computing*, 38(6):2220–2272, 2009.
- 7 Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 276–287. IEEE, 1994.
- 8 George Bennett. Probability inequalities for the sum of independent random variables. *Journal of the American Statistical Association*, 57(297):33–45, 1962.
- 9 Jöran Bergh and Jörgen Löfström. *Interpolation spaces: an introduction*, volume 223. Springer Science & Business Media, 2012.
- 10 SN Bernstein. *Probability theory, ogiz, moscow–leningrad* (1946).
- 11 Henry W Block, Thomas H Savits, Moshe Shaked, et al. Some concepts of negative dependence. *The Annals of Probability*, 10(3):765–772, 1982.

- 12 Stéphane Boucheron, Olivier Bousquet, Gábor Lugosi, Pascal Massart, et al. Moment inequalities for functions of independent random variables. *The Annals of Probability*, 33(2):514–560, 2005.
- 13 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.
- 14 Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *Distributed Computing*, 33(3):349–366, 2020.
- 15 Herman Chernoff et al. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.
- 16 Hamza ES Daoub. The fundamental theorem on symmetric polynomials. *The Teaching of Mathematics*, 28:55–59, 2012.
- 17 Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 93–110. Springer, 2014.
- 18 Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.
- 19 Devdatt P Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996.
- 20 Morris L Eaton. A note on symmetric bernoulli random variables. *The annals of mathematical statistics*, 41(4):1223–1226, 1970.
- 21 Morris L Eaton. A review of selected topics in multivariate probability inequalities. *The Annals of Statistics*, pages 11–43, 1982.
- 22 T Figiel, P Hitczenko, W Johnson, G Schechtman, and J Zinn. Extremal properties of rademacher functions with applications to the khintchine and rosenthal inequalities. *Transactions of the American Mathematical Society*, 349(3):997–1027, 1997.
- 23 Mohsen Ghaffari and Fabian Kuhn. Derandomizing distributed algorithms with small messages: Spanners and dominating set. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 24 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- 25 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 111–119. IEEE, 2012.
- 26 Meena Jagadeesan. Simple analysis of sparse, sign-consistent jl. *arXiv preprint*, 2017. [arXiv:1708.02966](https://arxiv.org/abs/1708.02966).
- 27 Svante Janson. New versions of suen’s correlation inequality. *Random Structures and Algorithms*, 13(3-4):467–483, 1998.
- 28 Daniel M Kane and Jelani Nelson. A derandomized sparse johnson-lindenstrauss transform. *arXiv preprint*, 2010. [arXiv:1006.3585](https://arxiv.org/abs/1006.3585).
- 29 David R Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 36–43, 2004.
- 30 Andreas Knoblauch. Closed-form expressions for the moments of the binomial probability distribution. *SIAM Journal on Applied Mathematics*, 69(1):197–204, 2008.
- 31 MJ Kronenburg. The binomial coefficient for negative arguments. *arXiv preprint*, 2011. [arXiv:1105.3689](https://arxiv.org/abs/1105.3689).
- 32 Rafał Łatała et al. Estimation of moments of sums of independent real random variables. *The Annals of Probability*, 25(3):1502–1513, 1997. URL: [https://projecteuclid.org/download/pdf\\_1/euclid.aop/1024404522](https://projecteuclid.org/download/pdf_1/euclid.aop/1024404522).
- 33 John E Littlewood. On the probability in the tail of a binomial distribution. *Advances in Applied Probability*, 1(1):43–72, 1969.

- 34 Michael Luby, Michael George Luby, and Avi Wigderson. *Pairwise independence and derandomization*, volume 4. Now Publishers Inc, 2006.
- 35 Colin Maclaurin. A second letter to martin folkes, esq.; concerning the roots of equations, with demonstration of other rules of algebra. *Philos. Trans. Roy. Soc. London Ser. A*, 36:59–96, 1729.
- 36 Andrew McGregor and Hoa T Vu. Better streaming algorithms for the maximum coverage problem. *Theory of Computing Systems*, 63(7):1595–1619, 2019.
- 37 Brendan D McKay. On littlewood’s estimate for the binomial distribution. *Advances in Applied Probability*, 21(2):475–478, 1989.
- 38 Abbas Mehrabian. Summary of concentration inequalities for the sum of k-wise independent random variables, 2011. URL: [https://www.cs.mcgill.ca/~amehra13/Articles/kwise\\_independent\\_concentration\\_summary.pdf](https://www.cs.mcgill.ca/~amehra13/Articles/kwise_independent_concentration_summary.pdf).
- 39 Dragoslav S. Mitrinović. Certain inequalities for elementary symmetric functions. *Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika*, (181/196):17–20, 1967. URL: <http://www.jstor.org/stable/43667273>.
- 40 Gergő Nemes. On the coefficients of the asymptotic expansion of  $n!$  *arXiv preprint*, 2010. [arXiv:1003.2907](https://arxiv.org/abs/1003.2907).
- 41 Isaac Newton. *Arithmetica universalis sive de compositione et resolutione arithmetica liber*, volume 1. apud Marcum Michaelis Rey, 1761.
- 42 Constantin Niculescu and Lars-Erik Persson. *Convex functions and their applications*. Springer, 2006.
- 43 Maciej Obremski and Maciej Skorski. Renyi entropy estimation revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
- 44 Thomas K Philips and Randolph Nelson. The moment bound is tighter than chernoff’s bound for positive tail probabilities. *The American Statistician*, 49(2):175–178, 1995.
- 45 Arjun Ramachandra and Karthik Natarajan. Tight probability bounds with pairwise independence. *arXiv preprint*, 2020. [arXiv:2006.00516](https://arxiv.org/abs/2006.00516).
- 46 Herbert Robbins. A remark on stirling’s formula. *The American mathematical monthly*, 62(1):26–29, 1955.
- 47 Haskell P Rosenthal. On the subspaces of  $\ell_p$  ( $p > 2$ ) spanned by sequences of independent random variables. *Israel Journal of Mathematics*, 8(3):273–303, 1970.
- 48 Shmuel Rosset. Normalized symmetric functions, newton’s inequalities, and a new set of stronger inequalities. *The American Mathematical Monthly*, 96(9):815–819, 1989.
- 49 Jörg-Rüdiger Sack and Jorge Urrutia. *Handbook of computational geometry*. Elsevier, 1999.
- 50 Jeanette P Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.
- 51 Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

# Eigenstripping, Spectral Decay, and Edge-Expansion on Posets

Jason Gaitonde ✉

Cornell University, Ithaca, NY, USA

Max Hopkins ✉

University of California, San Diego, La Jolla, CA, USA

Tali Kaufman ✉

Bar-Ilan University, Ramat-Gan, Israel

Shachar Lovett ✉

University of California, San Diego, La Jolla, CA, USA

Ruizhe Zhang ✉

The University of Texas at Austin, TX, USA

---

## Abstract

Fast mixing of random walks on hypergraphs (simplicial complexes) has recently led to myriad breakthroughs throughout theoretical computer science. Many important applications, however, (e.g. to LTCs, 2-2 games) rely on a more general class of underlying structures called *posets*, and crucially take advantage of non-simplicial structure. These works make it clear that the global expansion properties of posets depend strongly on their underlying architecture (e.g. simplicial, cubical, linear algebraic), but the overall phenomenon remains poorly understood. In this work, we quantify the advantage of different poset architectures in both a spectral and combinatorial sense, highlighting how *regularity* controls the spectral decay and edge-expansion of corresponding random walks.

We show that the spectra of walks on expanding posets (Dikstein, Dinur, Filmus, Harsha APPROX-RANDOM 2018) concentrate in strips around a small number of approximate eigenvalues controlled by the regularity of the underlying poset. This gives a simple condition to identify poset architectures (e.g. the Grassmann) that exhibit strong (even exponential) decay of eigenvalues, versus architectures like hypergraphs whose eigenvalues decay linearly – a crucial distinction in applications to hardness of approximation and agreement testing such as the recent proof of the 2-2 Games Conjecture (Khot, Minzer, Safra FOCS 2018). We show these results lead to a tight characterization of edge-expansion on expanding posets in the  $\ell_2$ -regime (generalizing recent work of Bafna, Hopkins, Kaufman, and Lovett (SODA 2022)), and pay special attention to the case of the Grassmann where we show our results are tight for a natural set of sparsifications of the Grassmann graphs. We note for clarity that our results do not recover the characterization of expansion used in the proof of the 2-2 Games Conjecture which relies on  $\ell_\infty$  rather than  $\ell_2$ -structure.

**2012 ACM Subject Classification** Theory of computation → Expander graphs and randomness extractors

**Keywords and phrases** High-dimensional expanders, posets, eposets

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.16

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2205.00644> [24]

**Funding** *Jason Gaitonde*: Supported by NSF Award CCF-1408673 and AFOSR Award FA9550-19-1-0183.

*Max Hopkins*: Supported by NSF Award DGE-1650112.

*Tali Kaufman*: Supported by ERC and BSF.

*Shachar Lovett*: Supported by NSF Award CCF-1909634.



© Jason Gaitonde, Max Hopkins, Tali Kaufman, Shachar Lovett, and Ruizhe Zhang; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 16; pp. 16:1–16:24



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Random walks on high dimensional expanders (HDX) have been the object of intense study in theoretical computer science in recent years. Starting with their original formulation by Kaufman and Mass [31], a series of works on the spectral structure of these walks [33, 16, 2] led to significant breakthroughs in approximate sampling [4, 2, 3, 12, 13, 11, 22, 28, 41, 9], CSP-approximation [1, 6], error-correcting codes [29, 30], agreement testing [19, 15, 32], and more. Most of these works focus on the structure of expansion in *hypergraphs* (also called *simplicial complexes*). However, hypergraphs are not always the appropriate object – recent breakthroughs in locally testable [17] and quantum LDPC codes [42, 40, 39] rely crucially on *cubical* structure not seen in hypergraphs, while many agreement testing results like the proof of the 2-2 Games Conjecture [44] relies on *linear algebraic* rather than simplicial structure.

In this work, we study a generalized notion of HDX on *partially ordered sets* (posets) introduced by Dikstein, Dinur, Filmus, and Harsha (DDFH) [16] called *expanding posets* (eposets). Random walks on eposets capture a broad range of structures beyond their hypergraph analogs, including natural sparsifications of the Grassmann graphs recently crucial to the resolution of the 2-2 Games Conjecture [44, 37, 21, 20, 8, 36]. While originally a *global* notion of expansion, Kaufman and Tessler (KT) [34] recently extended the study of eposets by introducing *local-to-global* analysis to the setting and by identifying *regularity* as a key parameter controlling expansion. The authors strengthened local-to-global theorems for strongly regular posets like the Grassmann, giving the first general formulation for characterizing expansion based on an eposet’s underlying architecture.

While analysis of the second eigenvalue is certainly important, a deeper understanding of the spectral structure of eposets is required for applications like the proof of the 2-2 Games Conjecture. Our main focus in this work lies in characterizing the spectral and combinatorial behavior of walks on eposets *beyond the second eigenvalue*. Strengthening DDFH and work of Bafna, Hopkins, Kaufman, and Lovett (BHKL) [6], we prove that at a coarse level (walks on) eposets exhibit the same spectral and combinatorial characteristics as expanding hypergraphs (e.g. spectral stripping, expansion of pseudorandom sets). On the other hand, as in KT, we show that the finer-grained properties of these objects are controlled by the underlying poset’s regularity, including the *rate of decay* of the spectrum and combinatorial expansion of associated random walks. This gives a strong separation between structures like hypergraphs with weak (linear) eigenvalue decay, and Grassmann-based eposets with strong (exponential) decay (a crucial property in the proof of the 2-2 Games Conjecture [44]).

### 1.1 Background

We briefly overview the theory of expanding posets and higher order random walks (see Section 2 for details). A  $d$ -dimensional graded poset is a set  $X$  equipped with a partial order “ $<$ ” and a ranking function  $r : X \rightarrow [d]$  that respects “ $<$ ” and partitions  $X$  into levels  $X(0) \cup \dots \cup X(d)$ . When  $x < y$  and  $r(y) = r(x) + 1$ , we write  $x \prec y$ . We assume throughout this work that our posets are *downward regular*: there exists a regularity function  $R(k, i)$  such that every  $k$ -dimensional element is greater than exactly  $R(k, i)$   $i$ -dimensional elements.

Graded posets come equipped with a natural set of operators called the *up* and *down* operators that lift or lower functions  $f : X(i) \rightarrow \mathbb{R}$  by averaging:

$$U_i f(x) = \mathbb{E}_{y \prec x} [f(y)] \quad D_i f(y) = \mathbb{E}_{x \succ y} [f(x)].$$

Composing the averaging operators leads to a natural notion of random walks on the underlying poset called *higher order random walks* (HD-walks). The simplest example is the *upper (lower) walk*  $D_{i+1} U_i (U_{i-1} D_i)$  which moves between elements  $x, x' \in X(i)$  via a common



element  $y \in X(i+1) \setminus X(i)$  with  $y > x, x'$  ( $y < x, x'$ ). We also consider longer variants of the upper and lower walks called *canonical walks*  $\hat{N}_k^i = D_{k+1} \circ \dots \circ D_{k+i} \circ U_{k+i-1} \circ \dots \circ U_k$  and  $\check{N}_k^i = U_k \circ \dots \circ U_{k-i} \circ D_{k-i+1} \circ \dots \circ D_k$  which similarly walk between  $k$ -dimensional elements in  $X(k)$  via a shared element in  $X(k+i)$  or  $X(k-i)$  respectively.

Following DDFH [16], we call a poset a  $(\delta, \gamma)$ -expander for  $\delta \in [0, 1]^{d-1}$  and  $\gamma \in \mathbb{R}_+$  if the upper and lower walks are spectrally similar up to a laziness factor:

$$\|D_{i+1}U_i - (1 - \delta_i)I - \delta_i U_{i-1}D_i\| \leq \gamma.$$

This generalizes standard spectral expansion which can be equivalently defined as looking at the spectral norm of  $A_G - U_0D_1$ , where  $A_G$  (the adjacency matrix) is exactly the non-lazy upper walk.<sup>1</sup> While most of our results hold in general, we assume a weak non-laziness condition on our underlying posets that holds in most cases of interest (see Definition 13).

## 1.2 Results

We now give an overview of our results, splitting this section into three parts for readability: spectral stripping, characterizing edge expansion, and applications to the Grassmann.

**Eigenstripping.** We start with our generalized spectral stripping theorem.

► **Theorem 1** (Spectrum of HD-Walks (informal Corollary 20)). *Let  $M$  be an HD-walk on the  $k$ th level of a  $(\delta, \gamma)$ -eposet. Then the spectrum of  $M$  is highly concentrated in  $k+1$  strips:*

$$\text{Spec}(M) \in \{1\} \cup \bigcup_{i=1}^k [\lambda_i(M) - e, \lambda_i(M) + e]$$

where  $e \leq O_{k,\delta}(\gamma)$ . Moreover, the span of eigenvectors in the  $i$ th strip approximately correspond to functions lifted from  $X(i)$  to  $X(k)$ .

This substantially simplifies and improves an analogous result of BHKL [6] on expanding hypergraphs, which had sub-optimal error dependence of  $O_k(\gamma^{1/2})$ . The main improvement stems from an optimal spectral stripping result for arbitrary inner product spaces of independent interest. Theorem 1 follows by showing that the HD-Level-Set Decomposition, a natural basis on eposets introduced by DDFH [16], gives such an approximate eigendecomposition.

In full generality, the approximate eigenvalues in Theorem 1 depend on the eposet parameters  $\delta$ , and can be fairly difficult to interpret. However, we show that under weak assumptions (see Section 2) the eigenvalues can be associated with the regularity of the underlying poset. We state the result just for lower walks for simplicity:

► **Theorem 2** (Regularity Controls Spectral Decay (informal Theorem 22)). *The approximate eigenvalues of the lower walk  $\check{N}_k^{k-i}$  on a  $(\delta, \gamma)$ -eposet are controlled by the poset's regularity function:  $\lambda_j(\check{N}_k^{k-i}) \in \frac{R(i,j)}{R(k,j)} \pm O_{k,\delta}(\gamma)$ .*

This generalizes work of Kaufman and Tessler [34] on the second eigenvalue, and reveals a major distinction among poset architectures: posets with higher regularity enjoy faster decay of eigenvalues.<sup>2</sup> Theorem 2 gives a new method of identifying poset architectures exhibiting

<sup>1</sup> For a broad range of posets, this is equivalent (up to constants) to *local-spectral expansion*, a notion of high dimensional expansion introduced by Dinur and Kaufman [19], as originally proved for simplicial complexes by DDFH [16], and later extended to a larger class of posets by Kaufman and Tessler [34].

<sup>2</sup> We note that Theorem 1 can also be obtained by combining our spectral stripping result with recent independent work of Dikstein, Dinur, Filmus, and Harsha [16, Section 8.4.1].

strong spectral decay in the sense that for any  $\delta > 0$ , the lower walk only contains  $O_\delta(1)$  approximate eigenvalues larger than  $\delta$ . This property is crucial for both the run-time of approximation algorithms on HDX [6] and the proof of the 2-2 Games Conjecture [44].

**Characterizing Edge Expansion.** Much of our motivation for studying the spectrum of HD-walks is to understand the *edge expansion* of subsets  $S$ , denoted  $\Phi(S)$  (see Section 5 for formal definition). Characterizing edge-expansion in HD-walks has recently proven crucial to understanding both algorithms for [5, 6] and hardness of unique games [44]. On expanding hypergraphs, it is known that *links* give the canonical example of small non-expanding sets.

► **Definition 3 (Link).** *Let  $X$  be a  $d$ -dimensional graded poset. The  $k$ -dimensional link, called a “ $k$ -link,” of an element  $\sigma \in X$  is the set of rank  $k$  elements greater than  $\sigma$ <sup>3</sup>, i.e.  $X_\sigma^k = \{y \in X(k) : y > \sigma\}$ . When  $k$  is clear from context, we write  $X_\sigma$  for  $X_\sigma^k$  for simplicity.*

BHKL [6] proved that on hypergraphs, the expansion of links is exactly controlled by their corresponding spectral strip. While their proof of this fact relied crucially on simplicial structure, we show via a more general analysis that the result can be recovered for eposets.

► **Theorem 4 (Expansion of Links (informal Theorem 29)).** *Let  $X$  be a  $(\delta, \gamma)$ -eposet and  $M$  an HD-walk on  $X(k)$ . Then for all  $0 \leq i \leq k$  and  $\tau \in X(i)$ ,  $\Phi(X_\tau) = 1 - \lambda_i(M) \pm O_{M,k,\delta}(\gamma)$ .*

Conversely one might ask: *are all non-expanding sets explained by links?* Following BHKL [6], given a set  $S$ , consider the function defined on a link  $\tau \in X(i)$  by  $L_{S,i}(\tau) := \mathbb{E}_{X_\tau}[\mathbb{1}_S] - \mathbb{E}[\mathbb{1}_S]$ .

Two standard formulations of “non-expansion is explained by links” correspond to  $L_{S,i}$  having noticeable  $\ell_2$  or  $\ell_\infty$ -norm for a non-expanding set  $S$ . Thus, we say  $S$  is *pseudorandom* if  $L_{S,i}$  is small with respect to one of these norms for all  $i \leq \ell$  (see Section 4 for precise definitions). We prove that pseudorandom sets expand near-optimally.

► **Theorem 5 (Pseudorandom Sets Expand (informal Theorem 33)).** *Let  $X$  be a  $(\delta, \gamma)$ -eposet and  $M$  a walk on  $X(k)$ . Then the expansion of any  $(\varepsilon, i)$ -pseudorandom set  $S$  is at least:*

$$\Phi(S) \geq 1 - \lambda_{i+1} - O_\delta(R(k, i)\varepsilon) - O_{k,\delta,M}(\gamma).$$

The main technical component behind Theorem 5 is a result called a “level- $i$ ” inequality (cf. Theorem 26) which asserts that pseudorandomness controls the projection of the indicator of a subset  $S$  onto eigenstrips. This strictly generalizes the result for simplicial complexes in [6] where  $R(k, i) = \binom{k}{i}$ , and is tight for other important settings such as the Grassmann (discussed below). Theorem 5 and Theorem 26 can also be viewed as another separation between eposet architectures, this time in terms of *combinatorial* properties.

**Applications:  $q$ -Eposets and the Grassmann Graphs.** We conclude with applications of our results to a particularly important class of eposets called “ $q$ -eposets.” Just like standard high dimensional expanders arise from expanding subsets of the complete complex (hypergraph),  $q$ -eposets arise from expanding subsets of the Grassmann Poset:

► **Definition 6 (Grassmann Poset).** *The Grassmann Poset is a graded poset  $(X, <)$  where  $X$  is the set of all subspaces of  $\mathbb{F}_q^n$  of dimension at most  $d$ , the partial ordering “ $<$ ” is given by inclusion, and the rank function is given by dimension.*

<sup>3</sup> In the literature, a link is often defined to be all such elements, not just those of rank  $k$ . We adopt this notation since we are mostly interested in working at a fixed level of the complex.

We call a (downward-closed) subset of the Grassmann poset a  $q$ -simplicial complex, and an expanding  $q$ -simplicial complex a  $q$ -eposet (see Section 2 for exact details). Using our machinery for general eposets, we prove a tight level- $i$  inequality for  $(\varepsilon, \ell)$ -pseudorandom sets  $S \subseteq X(k)$  (see Theorem 37): for all  $1 \leq i \leq \ell$ ,

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| \leq \left( \binom{k}{i}_q \varepsilon + O_{q,k}(\gamma) \right) \langle \mathbb{1}_S, \mathbb{1}_S \rangle,$$

where  $\mathbb{1}_{S,i}$  is the  $i$ th level of the HD-Level-Set Decomposition and  $\binom{k}{i}_q = \frac{(1-q^k) \cdots (1-q^{k-i+1})}{(1-q^i) \cdots (1-q)}$  is the Gaussian binomial coefficient. We also prove this bound cannot be improved by any constant factor, even in the  $\ell_\infty$ -regime. Furthermore, it is well known the dependence on  $k$  is necessary [37], even if one is willing to suffer a worse dependence on the pseudorandomness  $\varepsilon$ . This differs from simplicial complexes where the dependence can be removed in the  $\ell_\infty$ -regime [36, 7, 25]. Still, it is possible that the dependence on  $k$  can be removed by *changing the definition of pseudorandomness*, as was done on the Grassmann poset via finer-grained local structure called “zoom-in zoom-outs” [44]. The existence of a notion of locality based on the underlying poset structure that gives rise to  $k$ -independent bounds in the  $\ell_\infty$ -regime is an interesting open problem.

Finally, we give applications of these results to edge-expansion in an important class of walks that give rise to the well-studied *Grassmann graphs*.

► **Definition 7** (Grassmann Graphs). *The Grassmann Graph  $J_q(n, k, t)$  is the graph on  $k$ -dimensional subspaces of  $\mathbb{F}_q^n$  where  $(V, W) \in E$  exactly when  $\dim(V \cap W) = t$ .*

Note that non-lazy upper walk on the Grassmann poset is exactly the Grassmann graph  $J_q(n, k, k-1)$ . In Section 6, we show how to express any  $J_q(n, k, t)$  (in fact, for any  $q$ -simplicial complex) as a sum of standard higher order random walks. This leads to a set of natural sparsifications of the Grassmann graphs that may be of independent interest for agreement testing, PCPs, and hardness of approximation. For simplicity, on a given  $q$ -simplicial complex  $X$ , we refer to these “sparsified” Grassmann graphs as  $J_{X,q}(n, k, t)$  for the moment. The level- $i$  inequality then implies for a  $(\varepsilon, \ell)$ -pseudorandom set  $S \subseteq X(k)$  (Corollary 40):

$$\Phi(S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \varepsilon \sum_{i=1}^{\ell} \binom{t}{i}_q - q^{-(\ell+1)t} - O_{q,k}(\gamma).$$

In practice,  $t$  is generally thought of as being  $\Omega(k)$  (or even  $k - O(1)$ ), which results in a  $k$ -dependent bound. It remains an open problem whether a  $k$ -independent version can be proved for any  $q$ -eposet beyond the Grassmann poset itself.

### 1.3 Related Work

**Higher Order Random Walks.** Higher order random walks were introduced in 2016 by Kaufman and Mass [31]. Their spectral structure was later elucidated in a series of works by Kaufman and Oppenheim [33], DDFH [16], Alev, Jeronimo, and Tulsiani [1], Alev and Lau [2], and finally BHKL [6]. With the exception of DDFH, all of these works focused on hypergraphs rather than general posets. Our spectral stripping theorem for eposets essentially follows from combining eposet machinery developed by DDFH with our improved variant of BHKL’s general spectral stripping theorem.

Among the myriad applications of higher order random walks described above, our work is closest to that of Bafna, Barak, Kothari, Schramm, and Steurer [5], and BHKL [6], who used the spectral and combinatorial structure of HD-walks to build new algorithms for unique games. The analysis in this paper also lends itself to the algorithmic techniques developed in those works, but we are unaware of interesting examples beyond those in BHKL.

**High Dimensional Expansion Beyond Hypergraphs.** Most works listed above focus only on the setting of hypergraphs. However, recent years have also seen the nascent development and application of expansion beyond this setting [18, 42, 40, 39, 26], including the seminal work of DDFH [16] on expanding posets as well as more recent breakthroughs on locally testable and quantum codes [17, 42]. While DDFH largely viewed eposets as having similar structure, we strengthen the case that different underlying poset architectures exhibit different properties. This complements the results of Kaufman and Tessler [34], who showed that expanding posets with strong regularity conditions such as the Grassmann exhibit more favorable properties with respect to the second eigenvalue. Our results provide a statement of the same flavor looking at the entire spectrum, along with additional separations in more combinatorial settings. A related connection between poset regularity and the approximate spectrum of walks was independently developed by DDFH in a recent update of their seminal work [16].

**Expansion and Unique Games.** One motivation behind this work is towards building a more general framework for understanding the structure underlying the Unique Games Conjecture [35], a standard hardness assumption in complexity for many combinatorial optimization problems (see e.g. Khot’s survey [38]). In 2018, Khot, Minzer, and Safra [44] made a major breakthrough towards the UGC in proving the weaker 2-2 Games Conjecture, completing a long line of work in this direction [37, 21, 20, 8, 36, 44]. The key to the proof is the “Grassmann expansion hypothesis,” which states that any non-expanding set in the Grassmann graph  $J_q(d, k, k-1)$  is non-trivially concentrated inside a local-structure called “zoom-in zoom-outs.” As noted in the previous section, this result differs from our analysis in two key ways: it lies in the  $\ell_\infty$ -regime, and must be totally independent of dimension.

Unfortunately, little progress has been made towards the UGC since, as KMS’ proof of the Grassmann expansion hypothesis is quite complicated and highly tailored to the exact structure of the Grassmann, making it difficult to generalize to related conjectures [8]. However, just as the  $\ell_2$ -regime analysis of DDFH and BHKL recently lead to a dimension independent bound in the  $\ell_\infty$ -regime for standard HDX [7, 25], we expect the groundwork laid in this paper will be important for proving generalized dimension independent expansion hypotheses in the  $\ell_\infty$ -regime beyond the special case of the Grassmann graphs.

## 2 Preliminaries

**Graded Posets.** A partially ordered set (poset)  $P = (X, <)$  is a set of elements  $X$  endowed with a partial order “ $<$ ”. A graded poset has a rank function  $r : X \rightarrow \mathbb{N}$  satisfying:

1.  $r$  preserves “ $<$ ”: if  $y < x$ , then  $r(y) < r(x)$ .
  2.  $r$  preserves cover relations: if  $x$  is the smallest element greater than  $y$ , then  $r(x) = r(y) + 1$ .
- The function  $r$  partitions  $X$  into subsets by rank  $X(0) \cup \dots \cup X(d)$ , where  $\max_X(r) = d$ , and  $X(i) = r^{-1}(i)$ . We refer to a poset with maximum rank  $d$  as “ $d$ -dimensional”, and elements in  $X(i)$  as “ $i$ -faces”. Throughout this work, we consider  $d$ -dimensional graded posets that: (i) have a unique minimal element, and (ii) are “pure”: all maximal elements have rank  $d$ . Many graded posets of interest, like pure simplicial complexes and the Grassmann poset, satisfy certain regularity conditions which will be crucial to our analysis.

► **Definition 8** (Regularity). A  $d$ -dimensional graded poset is downward regular if for all  $i \leq d$  there exists some constant  $R(i)$  such that every element  $x \in X(i)$  covers exactly  $R(i)$  elements  $y \in X(i-1)$ .

A  $d$ -dimensional graded poset is middle-regular if for all  $0 \leq i \leq k \leq d$ , there exists a constant  $m(k, i)$  such that for any  $x_k \in X(k)$  and  $x_i \in X(i)$  satisfying  $x_k > x_i$ , there are exactly  $m(k, i)$  chains<sup>4</sup> of elements  $x_k > x_{k-1} > \dots > x_{i+1} > x_i$  where each  $x_j \in X(j)$ .

A poset is regular if it is both downward and middle regular.

We will assume all posets we discuss in this work are regular from this point forward. Regular posets also have the nice property that for any dimensions  $i < k$ , there exists a higher order regularity function  $R(k, i)$  such that any  $x \in X(k)$  is greater than exactly  $R(k, i)$  elements in  $X(i)$  (see Appendix A). We define  $R(i, i) = 1$  and  $R(j, i) = 0$  whenever  $j < i$  for convenience.

**Measured Posets and The Random Walk Operators.** A *measured poset* is a graded poset  $X$  endowed with a distribution  $\Pi = (\pi_0, \dots, \pi_d)$ , where each marginal  $\pi_i$  is a distribution over  $X(i)$ . We focus on the case where  $\Pi$  is induced entirely from  $\pi_d$ . That is,  $\forall 0 \leq i < d$ :

$$\pi_i(x) = \frac{1}{R(i+1, i)} \sum_{y \succ x} \pi_{i+1}(y).$$

In other words, each lower dimensional distribution  $\pi_i$  may be induced through the following process: an element  $y \in X(i+1)$  is selected with respect to  $\pi_{i+1}$ , and an element  $x \in X(i)$  such that  $x < y$  is then chosen uniformly at random.

The averaging operators  $U$  and  $D$  are defined analogously to their notions on simplicial complexes, with the main change being the use of the general regularity function  $R(i+1, i)$ :

$$U_i f(y) = \frac{1}{R(i+1, i)} \sum_{x \prec y} f(x) \quad D_{i+1} f(x) = \frac{1}{\pi_{i+1}(X_x)} \sum_{y \succ x} \pi_{i+1}(y) f(y),$$

where for  $i < k$  and  $x \in X(i)$ , the appropriate normalization factor is

$$\pi_k(X_x) = \sum_{y \in X(k): y > x} \pi_k(y) = R(k, i) \pi_i(x).$$

In Appendix A, we show that the up operators compose nicely, and in particular that:

$$U_i^k f(y) := U_{k-1} \circ \dots \circ U_i f(y) = \frac{1}{R(k, i)} \sum_{x \in X(i): x < y} f(x).$$

As with simplicial complexes, the down and up operators are adjoint with respect to the standard inner product on measured posets: for any  $f : X(k) \rightarrow \mathbb{R}$  and  $g : X(k-1) \rightarrow \mathbb{R}$ ,

$$\langle f, U_{k-1} g \rangle_{X(k)} = \langle D_k f, g \rangle_{X(k-1)}, \quad \text{where} \quad \langle f, g \rangle_{X(k)} = \sum_{\tau \in X(k)} \pi_k(\tau) f(\tau) g(\tau).$$

We omit  $X(k)$  from the notation when clear from context. This useful fact allows us to define basic self-adjoint notions of higher order random walks just like on simplicial complexes.

<sup>4</sup> Such objects are sometimes called flags, e.g. in the case of the Grassmann poset.

**Higher Order Random Walks.** Let  $C_k$  denote the set of functions  $f : X(k) \rightarrow \mathbb{R}$ . Following prior work, we define natural sets of random walk operators via the averaging operators.

► **Definition 9** (Walk Operators [31, 16, 1]). *Given a measured poset  $(X, \Pi)$ , a  $k$ -dimensional pure walk  $Y : C_k \rightarrow C_k$  on  $(X, \Pi)$  (of height  $h(Y)$ ) is a composition  $Y = Z_{2h(Y)} \circ \dots \circ Z_1$ , where each  $Z_i$  is a copy of  $D$  or  $U$ , and there are  $h(Y)$  of each type.*

*Let  $\mathcal{Y}$  be a family of pure walks  $Y : C_k \rightarrow C_k$  on  $(X, \Pi)$ . We call an affine combination  $M = \sum_{Y \in \mathcal{Y}} \alpha_Y Y$  a  $k$ -dimensional HD-walk on  $(X, \Pi)$  if it is stochastic and self-adjoint. The height of  $M$ , denoted  $h(M)$ , is the maximum height of any pure  $Y \in \mathcal{Y}$  with a non-zero coefficient. The weight of  $M$ , denoted  $w(M)$ , is  $|\alpha|_1$ .*

► **Definition 10** (Canonical Walk). *Given a  $d$ -dimensional measured poset  $(X, \Pi)$  and parameters  $k + j \leq d$ , the upper canonical walk is  $\hat{N}_k^j := D_k^{k+j} U_k^{k+j}$ , while for  $j \leq k$  the lower canonical walk is  $\check{N}_k^j := U_{k-j}^k D_{k-j}^k$ , where  $U_\ell^k = U_{k-1} \dots U_\ell$ , and  $D_\ell^k = D_{\ell+1} \dots D_k$ .*

Since the non-zero spectrum of  $\hat{N}_k^j$  and  $\check{N}_{k+j}^j$  are equivalent (c.f. [2]), we focus in this work mostly on the upper walks which we write simply as  $N_k^j$ .

For certain specially structured posets, we will also study an important class of HD-walks known as (partial) *swap walks*. We will introduce these well-studied walks momentarily.

**Expanding Posets and the HD-Level-Set Decomposition.** DDFH [16] observed that one can use the averaging operators to define an extension of spectral expansion to graded posets:

► **Definition 11** (Eposet [16]). *Let  $(X, \Pi)$  be a measured poset,  $\delta \in [0, 1]^{d-1}$ , and  $\gamma < 1$ .  $X$  is an  $(\delta, \gamma)$ -eposet if for all  $1 \leq i \leq d-1$ :*

$$\|D_{i+1}U_i - (1 - \delta_i)I - \delta_i U_{i-1}D_i\| \leq \gamma.$$

Much of our analysis in this work will be based off of an elegant approximate Fourier decomposition for eposets introduced by DDFH [16].

► **Theorem 12** (HD-Level-Set Decomposition, Theorem 8.2 [16]). *Let  $(X, \Pi)$  be a  $d$ -dimensional  $(\delta, \gamma)$ -eposet with  $\gamma$  sufficiently small. For all  $0 \leq k \leq d$ , let  $H^0 = C_0$ ,  $H^i = \text{Ker}(D_i)$ ,  $V_k^i = U_k^i H^i$ . Then  $C_k = V_k^0 \oplus \dots \oplus V_k^k$ . In other words, every  $f \in C_k$  has a unique decomposition  $f = f_0 + \dots + f_k$  such that  $f_i = U_k^i g_i$  for  $g_i \in \text{Ker}(D_i)$ .*

It is well known that the HD-Level-Set Decomposition is approximately an eigenbasis for HD-walks on simplicial complex [16, 1, 6]. We will show this statement extends to all eposets (extending DDFH’s similar analysis of the upper walk  $N_k^1$ ).

We will further assume for simplicity throughout this work an additional property of eposets we called (approximate) non-laziness.

► **Definition 13** ( $\beta$ -Non-Lazy Eposets). *Let  $(X, \Pi)$  be a  $d$ -dimensional measured poset. We call  $(X, \Pi)$   $\beta$ -non-lazy if for all  $1 \leq i \leq d$ ,  $\max_{\sigma \in X(i)} \{\mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\sigma\} \leq \beta$ .*

This condition asserts that no element in the poset carries too much weight, even upon conditioning. All of our results hold for general eposets,<sup>5</sup> but their form is significantly more interpretable when the poset is additionally non-lazy. In fact, most  $\gamma$ -eposets of interest are  $O(\gamma)$ -non-lazy. It is easy to see for instance that any “ $\gamma$ -local-spectral” expander satisfies this condition, an equivalent notion of expansion to  $\gamma$ -eposets under suitable regularity conditions [34]. We discuss this further in Appendix A.

<sup>5</sup> The one exception is the lower bound of Theorem 4.



**The Grassmann Poset and  $q$ -Eposets.** At the moment, there are only two known families of expanding posets of significant interest in the literature: those based on pure simplicial complexes (the downward closure of a  $k$ -uniform hypergraph), and pure  $q$ -simplicial complexes (the analogous notion over subspaces). The  $\ell_2$ -structure of the former is studied in detail in [6]; we focus on the latter which is less-studied, but responsible for a number of important results including the resolution of the 2-to-2 Games Conjecture [44].

► **Definition 14** ( $q$ -Simplicial Complex). *Let  $G_q(n, d)$  denote the  $d$ -dimensional subspaces of  $\mathbb{F}_q^n$ . A weighted, pure  $q$ -simplicial complex  $(X, \Pi)$  is given by a family of subspaces  $X \subseteq G_q(n, d)$  and a distribution  $\Pi$  over  $X$ . We will usually consider the downward closure  $X(0) \cup \dots \cup X(d)$ , where  $X(i) \subseteq G_q(n, i)$  consists of all  $i$ -dimensional subspaces contained in some element in  $X = X(d)$ . Further, on each level  $X(i)$ ,  $\Pi$  induces a natural distribution  $\pi_i$ :*

$$\forall V \in X(i) : \pi_i(V) = \frac{1}{\binom{d}{i}_q} \sum_{W \in X(d) : W \supset V} \pi_d(W),$$

where  $\pi_d = \Pi$  and  $\binom{d}{i}_q = \frac{(1-q^d) \dots (1-q^{d-i+1})}{(1-q^i) \dots (1-q)}$  is the Gaussian binomial coefficient.

Taking  $X = G_q(n, d)$  yields the Grassmann poset, the  $q$ -analog of the complete simplicial complex. The Grassmann poset is well known to be an expander in this sense (see e.g. [43]) – in fact it is a 0-eposet with parameters

$$\delta_i = \frac{(q^i - 1)(q^{n-i+1} - 1)}{(q^{i+1} - 1)(q^{n-i} - 1)}, \quad (1)$$

the  $q$ -analog of the epset parameters for the complete complex [16]. With this in mind, let's define a special class of epsets based on  $q$ -simplicial complexes.

► **Definition 15** ( $\gamma$ - $q$ -Eposet [16]). *A pure,  $d$ -dimensional weighted  $q$ -simplicial complex  $(X, \Pi)$  is a  $\gamma$ - $q$ -eposet if it is a  $(\delta, \gamma)$ -eposet satisfying  $\delta_i = q^{\frac{q^i - 1}{q^{i+1} - 1}}$  for all  $1 \leq i \leq d - 1$ .*

Constructing bounded-degree  $q$ -eposets (a problem proposed by DDFH [16]) remains an interesting open problem. Kaufman and Tessler [34] recently made some progress in this direction, but the expansion parameter of their construction is fairly poor (around  $1/2$ ).

Finally, in our applications to the Grassmann we consider a particularly important class of walks called *partial-swap walks*, which are non-lazy variants of the upper canonical walks.

► **Definition 16** (Partial-Swap Walk). *Let  $(X, \Pi)$  be a weighted,  $d$ -dimensional  $q$ -simplicial complex. The partial-swap walk  $S_k^j$  is the restriction of the canonical walk  $N_k^j$  to faces whose intersection has dimension  $k - j$ . In other words, if  $|V \cap W| > k - j$  then  $S_k^j(V, W) = 0$ , and otherwise  $S_k^j(V, W) \propto N_k^j(V, W)$ .*

When applied to the Grassmann poset itself, it is clear by symmetry that the partial-swap walk  $S_k^j$  returns exactly the Grassmann graph  $J_q(d, k, k - j)$ . On the other hand, it is not immediately obvious these objects are even HD-walks when applied to a generic  $q$ -simplicial complex. We prove this is the case in Section 6.

### 3 Eigenstripping and the Spectra of HD-Walks

We now discuss HD-walks' spectral structure. It turns out that on expanding posets, these walks exhibit almost exactly the same properties as on the special case of simplicial complexes studied in [33, 16, 1, 6]: a walk's spectrum lies concentrated in strips corresponding to levels of the HD-Level-Set Decomposition. The key to proving this lies in a more general theorem characterizing the spectral structure of any inner product space admitting an "approximate eigendecomposition" [6]. We prove a significantly simpler, tight variant of this result.

## 16:10 Eigenstripping, Spectral Decay, and Edge-Expansion on Posets

► **Theorem 17** (Eigenstripping). *Let  $M$  be a self-adjoint operator over an inner product space  $V$ , and suppose  $V = V^1 \oplus \dots \oplus V^k$  satisfies  $\|Mf - \lambda_i f\| \leq c_i \|f\|$  for all  $f \in V^i$  for parameters  $\lambda_1 \geq \dots \geq \lambda_n$  and  $c_i \geq 0$ . Then as long as  $c_i + c_{i+1} < \lambda_i - \lambda_{i+1}$ , the spectrum of  $M$  is concentrated around each  $\lambda_i$ :*

$$\text{Spec}(M) \subseteq \bigcup_{i=1}^k [\lambda_i - c_i, \lambda_i + c_i]$$

**Proof.** For each  $i$ , consider the (self-adjoint) operator  $M_i^2 = (M - \lambda_i I)^2$ . We claim it is enough to show that  $M_i^2$  has exactly  $\dim(V^i)$  eigenvalues less than  $c_i^2$  in absolute value. To see why, observe that the eigenvalues of  $M_i^2$  are exactly  $(\mu - \lambda_i)^2$  for each  $\mu$  in  $\text{Spec}(M)$  (with matching multiplicities), and therefore that any eigenvalue  $\mu_i \in \text{Spec}(M_i^2)$  less than  $c_i^2$  implies the existence of a corresponding eigenvalue of  $M$  in  $[\lambda_i \pm c_i]$ . If each  $M_i^2$  has  $\dim(V^i)$  eigenvalues less than  $c_i^2$ , then  $M$  has at least  $\dim(V^i)$  eigenvalues in each interval  $[\lambda_i \pm c_i]$ . Moreover, since these intervals are disjoint by assumption and  $\sum \dim(V^i) = \dim(V)$ , this must account for all eigenvalues of  $M$ .

To prove the claim, we apply the Courant-Fischer theorem [23], which asserts that the  $k$ th smallest eigenvalue of self-adjoint operator  $A$  is

$$\lambda_{n-k+1} = \min_U \left\{ \max_{f \in U} \left\{ \frac{\langle f, Af \rangle}{\langle f, f \rangle} \right\} \mid \dim(U) = k \right\}.$$

Taking  $U = V^i$ ,  $A = M_i^2$  and  $k = \dim(V^i)$  (noting that  $\langle f, M_i^2 f \rangle = \|(M - \lambda_i I)f\|^2$  by self-adjointness) with the approximate eigendecomposition assumption yields the claim. ◀

Note that this result is also trivially tight for any true eigendecomposition. We remark that similar strategies have been used in the numerical analysis literature (see e.g. [27]).

Thus it is enough to prove that the HD-Level-Set Decomposition is an approximate eigenbasis. This follows similarly as for local-spectral expanders [6], though somewhat more care is required to deal with general eposet parameters. First, an inductive application of [16, Claim 8.8] (itself a repeated application of Definition 11) shows that functions in the HD-Level-Set Decomposition are close to being eigenvectors (see full version for details).

► **Proposition 18.** *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet, and  $Y$  the pure balanced walk of height  $j$ , with down operators at positions  $(i_1, \dots, i_j)$ . For  $1 \leq \ell \leq k$ , let  $f_\ell = U_\ell^k g_\ell$  for some  $g_\ell \in H^\ell$ , and let*

$$\delta_j^k = \prod_{i=k-j}^k \delta_i, \quad \gamma_j^k = \gamma \sum_{i=-1}^{j-1} \delta_i^k,$$

where  $\delta_i^k = 1$  for any  $i < 0$  for notational convenience. Then  $f_\ell$  is an approximate eigenvector of  $Y$ :

$$\left\| Y f_\ell - \prod_{s=1}^j \left( 1 - \delta_{k-2s+i_s-\ell}^{k-2s+i_s} \right) f_\ell \right\| \leq \|g_\ell\| \sum_{s=1}^j \gamma_{k-2s+i_s-\ell}^{k-2s+i_s} \prod_{t=1}^{s-1} \left( 1 - \delta_{k-2t+i_t-\ell}^{k-2t+i_t} \right) \leq (j+k)j\gamma \|g_\ell\|.$$

When  $\gamma = 0$ , this implies that the HD-Level-Set decomposition is a true eigendecomposition. Since balanced walks are simply affine combinations of pure walks, this immediately implies a similar result for the more general case.

Before proceeding, for a  $d$ -dimensional  $(\delta, \gamma)$ -eposet, and  $0 \leq \ell \leq k < d$ , define:

$$\rho_\ell^k = \prod_{i=1}^{k-\ell} (1 - \delta_{k-\ell-i}^{k-i}), \quad \rho_{\min} = \min_{0 \leq \ell \leq k} \{\rho_\ell^k\}. \quad (8)$$



The parameter  $\rho_\ell^k$  arises throughout much of our work, and while it is difficult to interpret on general eposets, we prove it has a very natural form as long as non-laziness holds.

▷ **Claim 19** ( $\rho_\ell^k$  for Regular Eposets). Let  $(X, \Pi)$  be a regular,  $\gamma$ -non-lazy<sup>6</sup>  $d$ -dimensional  $(\delta, \gamma)$ -eposet. Then for any  $i \leq k < d$ , we have:

$$\rho_i^k \in \frac{1}{R(k, i)} \pm \text{err},$$

where  $\text{err} \leq O\left(\frac{i^3 k^2 R_{\max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$ . Likewise as long as  $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{i^3 k^2 R_{\max}^2}\right)$  we have  $\rho_{\min}^{-1} \leq O(R_{\max})$ , where  $R_{\max} := \max_{0 \leq i \leq k} \{R(k, i)\}$ .

This gives a nice generalization of the interpretation of  $\rho_i^k$  on hypergraphs, where  $\rho_i^k = \binom{k}{i}^{-1}$  [16]. We prove this claim in Appendix A. For simplicity, we will assume throughout the rest of this work that our eposets are  $\gamma$ -non-lazy, which is true for most cases of interest (see Appendix A). All results hold in the more general case using  $\rho_i^k$  unless otherwise noted.

Combining Proposition 18 and [16, Lemma 8.11] immediately implies that the HD-Level-Set Decomposition is an approximate eigendecomposition:

► **Corollary 20.** Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet and let  $M = \sum_{Y \in \mathcal{Y}} \alpha_Y Y$  be an HD-walk. For  $1 \leq \ell \leq k$ , if  $f_\ell = U_\ell^k g_\ell$  for some  $g_\ell \in H^\ell$ , then for  $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R_{\max}^2}\right)$ :

$$\left\| M f_\ell - \left( \sum_{Y \in \mathcal{Y}} \alpha_Y \lambda_{Y, \delta, \ell} \right) f_\ell \right\| \leq c \gamma \|f_\ell\|,$$

where  $\lambda_{Y, \delta, \ell}$  is the corresponding eigenvalues of the pure balanced walk  $Y$  on a  $(\delta, 0)$ -eposet (the form of which are given in Proposition 18), and  $c \leq O((h(M) + k)h(M)R(k, \ell)w(M))$ .

Theorem 17 then immediately implies that for any self-adjoint walk (e.g. canonical or swap walk), the true spectrum is concentrated around these approximate eigenvalues.

A straightforward, but useful example application of Corollary 20 immediately yields the approximate spectrum of a basic higher order random walk.

► **Corollary 21** (Spectrum of Lower Canonical Walks). Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet. The approximate eigenvalues of the canonical lower walk  $\tilde{N}_k^{k-\ell}$  are:

$$\lambda_j(\tilde{N}_k^{k-\ell}) = \prod_{s=1}^{k-\ell} (1 - \delta_{k-s-j}^{k-s}).$$

Similar to the case of  $\rho_i^k$ , while this is difficult to interpret in the general setting, the eigenvalues have a very natural form on non-lazy eposets given by the regularity parameters.

► **Theorem 22.** Let  $(X, \Pi)$  be a  $\gamma$ -non-lazy  $(\delta, \gamma)$ -eposet. The approximate eigenvalues of the canonical lower walk  $\tilde{N}_k^{k-i}$  are  $\lambda_j(\tilde{N}_k^{k-i}) \in \frac{R(i, j)}{R(k, j)} \pm c \gamma$ , where  $c \leq O\left(\frac{i^4 k^2 R_{\max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$ .

The proof requires machinery developed in Section 5 and is given in Appendix A.

<sup>6</sup> One can prove this claim more generally for any  $\beta$ -non-laziness, but most  $\gamma$ -eposets of interest are additionally  $\gamma$ -non-lazy, so this simplified version is generally sufficient.

#### 4 Pseudorandomness and the HD-Level-Set Decomposition

Now that we know the spectral structure of HD-walks, we shift to studying their combinatorial structure. In particular, we will focus on how natural notions of pseudorandomness control the projection of functions onto the HD-Level-Set Decomposition. As much of this theory generalizes arguments of BHKL, we defer the proofs in this section to the full version.

We start with the definition of pseudorandomness in the  $\ell_2$ -regime, which measures the variance of a set across links.

► **Definition 23** ( $\ell_2$ -Pseudorandom Functions [6]). *A function  $f \in C_k$  is  $(\varepsilon_1, \dots, \varepsilon_\ell)$ - $\ell_2$ -pseudorandom if its variance across  $i$ -links is small for all  $1 \leq i \leq \ell$ :*

$$\text{Var}(D_i^k f) \leq \varepsilon_i |\mathbb{E}[f]|.$$

In their work on simplicial complexes, BHKL [6] observed a close connection between  $\ell_2$ -pseudorandomness, the HD-Level-Set Decomposition, and the spectra of the lower canonical walks. Using the approximate eigendecomposition developed in the previous section in Corollary 21, it turns out that the same connection holds in general for eposets.

► **Theorem 24.** *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet with  $\gamma \leq O\left(\frac{\max_i \{\delta_i(1-\delta_{i-1})\}}{k^5 R_{\max}^2}\right)$ . If  $f \in C_k$  has HD-Level-Set Decomposition  $f = f_0 + \dots + f_k$ , then for any  $\ell \leq k$ ,  $\text{Var}(D_\ell^k f)$  is controlled by its projection onto  $V_k^0 \oplus \dots \oplus V_k^\ell$  in the following sense:*

$$\text{Var}(D_\ell^k f) \in \sum_{j=1}^{\ell} \lambda_j(\tilde{N}_k^{k-\ell}) \langle f, f_j \rangle \pm c_k \gamma \|f\|^2,$$

where  $c_k \leq O(k^{5/2} R_{\max})$  and  $\lambda_j(\tilde{N}_k^{k-\ell}) = \prod_{s=1}^{k-\ell} (1 - \delta_{k-s-j}^{k-s})$ .

While  $\ell_2$ -pseudorandomness is useful in its own right (e.g. for local-to-global algorithms for unique games [5, 6]), there is also significant interest in a stronger  $\ell_\infty$ -variant in the hardness of approximation literature [36, 44].

► **Definition 25** ( $\ell_\infty$ -Pseudorandom Functions [6]). *A function  $f \in C_k$  is  $(\varepsilon_1, \dots, \varepsilon_\ell)$ - $\ell_\infty$ -pseudorandom if for all  $1 \leq i \leq \ell$  its local expectation is close to its global expectation:*

$$\|D_i^k f - \mathbb{E}[f]\|_\infty \leq \varepsilon_i.$$

In their recent work on  $\ell_2$ -structure of expanding simplicial complexes, BHKL prove a basic reduction from  $\ell_\infty$  to  $\ell_2$ -pseudorandomness that allows for an analogous level- $i$  inequality for this notion as well. We show the same result holds for general eposets by applying Theorem 24 with Claim 19 to obtain a level- $i$  inequality for pseudorandom sets (see the full version for the more general version). The key idea is to lower bound the variance of  $D_i^k$  by the  $i$ th component in the expansion of variance given by Theorem 24.

► **Theorem 26.** *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet with  $\gamma \leq O\left(\frac{\max_i \{\delta_i(1-\delta_{i-1})\}}{k^5 R_{\max}^2}\right)$  and suppose that for  $S \subseteq X(k)$ ,  $\mathbb{1}_S$  is  $(\varepsilon_1, \dots, \varepsilon_\ell)$ - $\ell_\infty$  pseudorandom. Then  $\mathbb{1}_S$  is also  $(\varepsilon_1, \dots, \varepsilon_\ell)$ - $\ell_2$  pseudorandom, and for all  $1 \leq i \leq \ell$ :*

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| \leq (R(k, i) \varepsilon_i + c \gamma) \mathbb{E}[\mathbb{1}_S],$$

where  $c \leq O\left(\frac{k^5 R_{\max}^2}{\max_i \{\delta_i(1-\delta_{i-1})\}}\right)$ .

This recovers the tight inequality for simplicial complexes given in [6] where  $R(k, i) = \binom{k}{i}$ , as well as providing the natural  $q$ -analog for  $q$ -simplicial complexes where  $R(k, i) = \binom{k}{i}_q$ .

## 5 Expansion of HD-walks

It is well known that higher order random walks on simplicial complexes are not small-set expanders. BHKL gave an exact characterization of this phenomenon for local-spectral expanders: they showed that the expansion of any  $i$ -link with respect to an HD-walk  $M$  is almost exactly  $1 - \lambda_i(M)$ . Moreover, using the level- $i$  inequality from the previous section, BHKL proved a tight converse to this result in an  $\ell_2$ -sense: *any* non-expanding set must have high variance across links. This gave a complete  $\ell_2$ -characterization of non-expanding sets on local-spectral expanders, and lay the structural groundwork for new algorithms for unique games over HD-walks. In this section, we extend these results to general expanding posets.

► **Definition 27** (Weighted Edge Expansion). *Let  $M$  be a  $k$ -dimensional HD-Walk on a graded poset  $(X, \Pi)$ . Let  $M(v, X(k) \setminus S) = \sum_{y \in X(k) \setminus S} M(v, y)$  where  $M(v, y)$  is the transition probability from  $v$  to  $y$ . The weighted edge expansion of a subset  $S \subset X(k)$  with respect to  $M$  is*

$$\Phi(S) = \mathbb{E}_{v \sim \pi_k|_S} [M(v, X(k) \setminus S)].$$

Before we prove the strong connections between links and expansion, we need to introduce an important property of HD-walks, monotonic eigenvalue decay.

► **Definition 28** (Monotonic HD-walk). *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet. We call an HD-walk  $M$  monotonic if its approximate eigenvalues  $\lambda_i(M)$  (given in Corollary 20) are non-increasing.*

Most HD-walks of interest (e.g. pure walks, partial-swap walks on simplicial or  $q$ -simplicial complexes, etc.) are monotonic. This property will be crucial to understanding expansion. To start, let's see how it allows us to upper bound the expansion of links.

► **Theorem 29** (Local Expansion vs Global Spectra). *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet and  $M$  be a  $k$ -dimensional monotonic HD-walk. Then for all  $0 \leq i \leq k$  and  $\tau \in X(i)$ , it holds that  $\Phi(X_\tau) \in 1 - \lambda_i(M) \pm c\gamma$ , where  $c \leq O\left(\frac{k^5 R_{\max}^2 (h(M) + k) h(M) w(M)}{\delta_{k-i}^k (1 - \delta_{i-1})}\right)$ .*

The key to proving Theorem 29 is to show that the weight of an  $i$ -link lies almost entirely on level  $i$  of the HD-Level-Set Decomposition. To show this, we rely on another connection between regularity and eposet parameters for non-lazy posets.

► **Claim 30.** Let  $(X, \Pi)$  be a  $d$ -dimensional  $(\delta, \gamma)$ -eposet. Then for every  $1 \leq k \leq d$  and  $0 \leq i \leq k$ , the following relation between the eposet and regularity parameters holds:

$$\lambda_i(N_k^1) \in \frac{R(k, i)}{R(k+1, i)} \pm (\gamma_{k-i}^k + R(k, i) \delta_{k-i}^k \gamma).$$

We prove this relation in Appendix A. With this in hand, we can show links project mostly onto their corresponding level. We defer the proof to the full version, but the key idea is to express the (non)-expansion of the link both directly using the regularity parameters and using the approximate eigenvalues in the HD-Level-Set decomposition to argue that the only way these quantities can be equal is if the desired conclusion holds.

► **Lemma 31.** *Let  $(X, \Pi)$  be a  $d$ -dimensional  $(\delta, \gamma)$ -eposet with  $\gamma \leq O\left(\frac{\max_i \{\delta_i (1 - \delta_{i-1})\}}{k^5 R_{\max}^2}\right)$ . Then for all  $0 \leq i \leq k < d$  and  $\tau \in X(i)$ , for all  $j \neq i$ :*

$$\left| \frac{\langle \mathbb{1}_{X_\tau, i}, \mathbb{1}_{X_\tau, j} \rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau} \rangle} \right| \leq O\left(\frac{k^3 R_{\max}}{\delta_{k-i}^k (1 - \delta_{i-1})} \gamma\right).$$

We note that the above is the only result in our work that truly relies on non-laziness (it is used only to replace  $\rho$  with regularity in all other results). It is possible to recover the upper bound in Theorem 29 for general eposets via arguments used in [6], but the lower bound remains open for concentrated posets. With that in mind, we now prove Theorem 29.

**Proof of Theorem 29.** By the previous lemma, we have

$$\left| \frac{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau, j} \rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau} \rangle} \right| \leq O \left( \frac{1}{\delta_{k-i}^k (1 - \delta_{i-1})} \cdot \left( \frac{k^3}{\rho_{\min}} \gamma + R(k, i) \gamma \right) \right).$$

Expanding out  $\bar{\Phi}(\mathbb{1}_{X_\tau})$  then gives:

$$\begin{aligned} \bar{\Phi}(\mathbb{1}_{X_\tau}) &= \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau} \rangle} \sum_{j=0}^i \langle \mathbb{1}_{X_\tau}, M \mathbb{1}_{X_\tau, j} \rangle \\ &\leq \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau} \rangle} \sum_{j=0}^i \lambda_i(M) \langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau, j} \rangle + c_2 \gamma \\ &\leq \lambda_i(M) \frac{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau, i} \rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau} \rangle} + err_1 \\ &\leq \lambda_i(M) + err_2. \end{aligned}$$

where  $c_2, err_1, err_2 \leq O \left( \frac{k}{\delta_{k-i}^k (1 - \delta_{i-1})} \left( \frac{k^2(h(M)+k)h(M)w(M)}{\rho_{\min}} \gamma + R(k, i) \gamma \right) \right)$  and the last step follows from the previous lemma. The conclusion then follows from applying Claim 19.  $\blacktriangleleft$

Thus, for sufficiently nice expanding posets, the expansion of any  $i$ -link with respect to an HD-walk is almost exactly  $1 - \lambda_i(M)$ . As HD-walks are generally poor expanders (have large  $\lambda_1(M)$ ), Theorem 29 implies that links are examples of small, non-expanding sets. Following BHKL, we now prove a converse to this result: any non-expanding set must be explained by some structure inside links. To give a precise statement, we need the following definition:

► **Definition 32** (Stripped Threshold Rank [6]). *Let  $(X, \Pi)$  be a  $(\delta, \gamma)$ -eposet and  $M$  a  $k$ -dimensional HD-walk with  $\gamma$  small enough that the HD-Level-Set Decomposition has a corresponding decomposition of disjoint eigenstrips  $C_k = \bigoplus W_k^i$ . The ST-Rank of  $M$  with respect to  $\eta$  is the number of strips containing an eigenvector with eigenvalue at least  $\eta$ :*

$$R_\eta(M) = |\{W_k^i : \exists f \in V^i, Mf = \lambda f, \lambda > \eta\}|.$$

With this definition, we provide a converse to Theorem 29 in both  $\ell_2$  and  $\ell_\infty$  senses:

► **Theorem 33.** *Let  $(X, \Pi)$   $(\delta, \gamma)$ -eposet,  $M$  a  $k$ -dimensional, monotonic HD-walk, and  $\gamma$  small enough that the eigenstrip intervals of Theorem 17 are disjoint. For any  $\eta > 0$ , let  $r = R_\eta(M) - 1$ . Then the expansion of a set  $S \subset X(k)$  of density  $\alpha$  is at least:*

$$\Phi(S) \geq 1 - \alpha - (1 - \alpha)\eta - \sum_{i=1}^r (\lambda_i(M) - \eta) R(k, i) \varepsilon_i - c\gamma$$

where  $S$  is  $(\varepsilon_1, \dots, \varepsilon_r)$ -pseudorandom and  $c \leq O \left( \frac{k^5 R_{\max}^2(h(M)+k)h(M)w(M)}{\max_i \{\delta_i(1 - \delta_{i-1})\}} \right)$ .

The argument is similar to [6] for simplicial complexes and relies on similar manipulations to Theorem 29, so we defer the proof to the full version. Theorem 33 recovers the analogous result for simplicial complex in [6] with the appropriate value  $R(k, i) = \binom{k}{i}$ . BHKL also prove this special case is tight in multiple senses; see the discussion there for more details.

## 6 The Grassmann and $q$ -eposets

In this section, we specialize to expanding subsets of the Grassmann poset. We will show that our analysis is tight in this regime.

**Spectra.** We start by examining the spectrum of HD-walks on the Grassmann and  $q$ -eposets, focusing on the most widely used walks in the literature, the canonical and partial-swap walks. To start, recall that the Grassmann poset itself is a 0-eposet. Plugging the parameters in Equation (1) into Proposition 18, along with an analogous calculation for  $q$ -eposets, gives a nice exact form for the spectra of canonical walks.

► **Corollary 34** ( $N_k^j$  Spectra). *Let  $X = G_q(n, d)$  be the Grassmann Poset,  $k + j \leq d$ , and  $f_\ell = U_\ell^k g_\ell$  for some  $g_\ell \in H^\ell$ . Then  $N_k^j f_\ell = \lambda_\ell f_\ell$ , where*

$$\lambda_\ell = q^{\ell j} \frac{\binom{k+j-\ell}{j}_q \binom{n-k-\ell}{j}_q}{\binom{k+j}{j}_q \binom{n-k}{j}_q} \approx q^{-\ell j}.$$

Similarly, let  $(X, \Pi)$  be a  $d$ -dimensional  $\gamma$ - $q$ -eposet with  $\gamma \leq q^{-\Omega(k^2)}$ ,  $k + j \leq d$ , and  $f_\ell = U_\ell^{k-1} g_\ell$  for some  $g_\ell \in H^\ell$ . Then:

$$\left\| N_k^j f_\ell - \frac{\binom{k+j-\ell}{j}_q}{\binom{k+j}{j}_q} f_\ell \right\| \leq O \left( j(j+k) \binom{k}{\ell}_q \right) \gamma \|f_\ell\|$$

The first equation above recovers a very simple proof of classical results to this effect (see e.g. [14]). Note that for small enough  $\gamma$ , Theorem 17 implies that the true spectra is then concentrated around these values as well. These eigenvalues are, as one would expect, the natural  $q$ -analog of the corresponding eigenvalues on simplicial complexes.

Moreover, this carries over to the class of partial-swap walks, which were originally analyzed by AJT on simplicial complexes [1]. To see this, we first need to show (in Appendix B) these walks are indeed HD-walks, which follows from the  $q$ -analog of statements in AJT [1].

► **Lemma 35.** *Let  $(X, \Pi)$  be a pure, measured  $q$ -simplicial complex. Then:*

$$N_k^j = \sum_{i=0}^j q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} S_k^i, \text{ and } S_k^j = \frac{1}{q^{j^2} \binom{k}{k-j}_q} \sum_{i=0}^j (-1)^{j-i} q^{\binom{j-i}{2}} \binom{j}{i}_q \binom{k+i}{i}_q N_k^i.$$

This is unsurprisingly the  $q$ -analog of the analogous statement on simplicial complexes (see [1, Corollary 4.13]). Finally, combining the previous result with Corollary 20 and Corollary 34, it is possible to show that the eigenvalues of partial-swap walks on  $q$ -simplicial complexes are given by the natural  $q$ -analog of the simplicial complex case (see the full version for details):

► **Corollary 36.** *Let  $(X, \Pi)$  be a  $d$ -dimensional  $\gamma$ - $q$ -eposet with  $\gamma$  sufficiently small,  $k + j \leq d$ , and  $f_\ell = U_\ell^k g_\ell$  for some  $g_\ell \in H^\ell$ . Then:*

$$\left\| S_k^j f_\ell - \frac{\binom{k-j}{\ell}_q}{\binom{k}{\ell}_q} f_\ell \right\| \leq O \left( \left( \frac{q}{q-1} \right)^{\min(j, k-j)+2} k^2 \binom{k}{\ell}_q \right) \gamma \|f_\ell\|$$

Again, since the swap walks are self-adjoint Theorem 17 implies that for small enough  $\gamma$  the true spectra is closely concentrated around these values as well.

**Pseudorandom Functions and Small Set Expansion.** With an understanding of the spectra of HD-walks on  $q$ -simplicial complexes, we move to studying its combinatorial structure. By direct computation, it is not hard to show that on  $q$ -eposets,  $\rho_\ell^k = \binom{k}{\ell}_q^{-1}$  (Claim 19 would only imply this is approximately true). As a result, we get a level- $i$  inequality for  $q$ -simplicial complexes that is the natural  $q$ -analog of BHKL's inequality for basic simplicial complexes.

► **Theorem 37.** *Let  $(X, \Pi)$  be a  $\gamma$ - $q$ -eposet with  $\gamma \leq q^{-\Omega(k^2)}$  and let  $S \subseteq X(k)$ . If  $\mathbb{1}_S$  is  $(\varepsilon_1, \dots, \varepsilon_\ell)$ - $\ell_\infty$ -pseudorandom, then for  $c \leq q^{O(k^2)}$ ,*

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| \leq \left( \binom{k}{i}_q \varepsilon_i + c\gamma \right) \mathbb{E}[\mathbb{1}_S] \quad \forall 1 \leq i \leq \ell.$$

For large enough  $q, \gamma^{-1}$ , this result is exactly tight. The key to showing this fact is to examine a local structure unique to the Grassmann called *co-links*. The co-link of an element  $W \in X(k')$ , is all of the subspaces contained in  $W$ , i.e.  $\bar{X}_W = \{V \in X(k) : V \subseteq W\}$ . Just like links, it turns out that co-links of dimension  $i$  (that is  $k' = d - i$ ) also come through levels 0 through  $i$  of the complex, although this is somewhat trickier to see. The essential observation is that co-links satisfy enough symmetries to explicitly construct a function in  $C_i$  whose image under  $U_i^k$  yields the desired function; see the full version for the construction.

► **Lemma 38 (HD-Level-Set Decomposition of Co-Links).** *Let  $X = G_q(d, k)$  and  $S = \bar{X}_W$  be a co-link of dimension  $i$  for  $W \in X(d - i)$ . Then, we have  $\mathbb{1}_S \in V_k^0 \oplus \dots \oplus V_k^i$ .*

Using this fact, we can show that our level- $i$  inequality is exactly tight on co-links, deferring the proof to Appendix B.

► **Proposition 39.** *Let  $X = G_q(d, k)$  be the Grassmann poset. For any  $i \leq k \in \mathbb{N}$  and  $c < 1$ , there exist large enough  $q, d$  and a  $(i, \varepsilon_i)$ -pseudorandom set  $S \subset X(k)$  such that*

$$\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle > c \binom{k}{i}_q \varepsilon_i \langle \mathbb{1}_S, \mathbb{1}_S \rangle.$$

Finally, Theorem 37 directly implies that for the canonical and partial-swap walks, sufficiently pseudorandom functions expand near perfectly.

► **Corollary 40 ( $q$ -Eposets Edge-Expansion).** *Let  $(X, \Pi)$  be a  $d$ -dimensional  $\gamma$ - $q$ -eposet,  $S \subset X(k)$  a subset whose indicator function  $\mathbb{1}_S$  is  $(\varepsilon_1, \dots, \varepsilon_\ell)$ -pseudorandom. Then the edge expansion of  $S$  with respect to the canonical walk  $N_k^j$  is bounded by:*

$$\Phi_{\pi_k}(N_k^j, S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \sum_{i=1}^{\ell} \frac{\binom{k+j-i}{j}_q}{\binom{k+j}{j}_q} \binom{k}{i}_q \varepsilon_i - q^{-(\ell+1)j} - q^{O(k^2)}\gamma.$$

Further, the edge expansion of  $S$  with respect to the partial-swap walk  $S_k^j$  is bounded by:

$$\Phi_{\pi_k}(S_k^j, S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \sum_{i=1}^{\ell} \binom{k-j}{i}_q \varepsilon_i - q^{-(\ell+1)j} - q^{O(k^2)}\gamma.$$

Note that  $S_k^j$  on  $q$ -eposets is a generalization of the Grassmann Graphs (and are equivalent when  $X$  is the Grassmann Poset). While our definition of pseudorandomness is weaker than that of [44] and therefore necessarily depends on the dimension  $k$ , we take the above as evidence that the framework of expanding posets may be important for making further progress on the Unique Games Conjecture. In particular, combined with recent works removing this  $k$ -dependence on simplicial complexes [7, 25], it seems plausible that the framework of expanding posets may lead to a more general understanding of the structure underlying the Unique Games Conjecture.

## References

- 1 Vedat Levi Alev, Fernando Granha Jeronimo, and Madhur Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–201. IEEE, 2019.
- 2 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. *arXiv preprint*, 2020. [arXiv:2001.02827](#).
- 3 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *arXiv preprint*, 2020. [arXiv:2001.00303](#).
- 4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.
- 5 Mitali Bafna, Boaz Barak, Pravesh Kothari, Tselil Schramm, and David Steurer. Playing unique games on certified small-set expanders. *arXiv preprint*, 2020. [arXiv:2006.09969](#).
- 6 Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. High dimensional expanders: Eigenstripping, pseudorandomness, and unique games. *arXiv e-prints*, 2020. [arXiv:2011.04658](#).
- 7 Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. Hypercontractivity on high dimensional expanders: a local-to-global approach for higher moments. *arXiv preprint*, 2021. [arXiv:2111.09444](#).
- 8 Boaz Barak, Pravesh K Kothari, and David Steurer. Small-set expansion in shortcode graph and the 2-to-2 conjecture. *arXiv preprint*, 2018. [arXiv:1804.08662](#).
- 9 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. *arXiv preprint*, 2021. [arXiv:2103.07459](#).
- 10 Andries E Brouwer and Willem H Haemers. Distance-regular graphs. In *Spectra of graphs*, pages 177–185. Springer, 2012.
- 11 Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1548–1557. SIAM, 2021.
- 12 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of glauher dynamics up to uniqueness via contraction. *arXiv preprint*, 2020. [arXiv:2004.09083](#).
- 13 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauher dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1537–1550, 2021.
- 14 Philippe Delsarte. Association schemes and t-designs in regular semilattices. *Journal of Combinatorial Theory, Series A*, 20(2):230–243, 1976.
- 15 Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1495–1524. IEEE, 2019.
- 16 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 17 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. *arXiv preprint*, 2021. [arXiv:2111.04808](#).
- 18 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. *arXiv preprint*, 2021. [arXiv:2111.04808](#).
- 19 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985. IEEE, 2017.



- 20 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in grassmann graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 940–951, 2018.
- 21 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 376–389, 2018.
- 22 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the boolean domain. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1558–1577. SIAM, 2021.
- 23 Ernst Fischer. Über quadratische formen mit reellen koeffizienten. *Monatshefte für Mathematik und Physik*, 16(1):234–249, 1905.
- 24 Jason Gaitonde, Max Hopkins, Tali Kaufman, Shachar Lovett, and Ruizhe Zhang. Eigenstripping, spectral decay, and edge-expansion on posets. *CoRR*, abs/2205.00644, 2022. doi:10.48550/arXiv.2205.00644.
- 25 Tom Gur, Noam Lifshitz, and Siqi Liu. Hypercontractivity on high dimensional expanders: Approximate efroonstein decompositions for epsilon-product spaces. *arXiv preprint*, 2021. arXiv:2111.09375.
- 26 Max Hopkins and Ting-Chun Lin. Explicit lower bounds against  $\omega(n)$ -rounds of sum-of-squares. *arXiv preprint*, 2022. arXiv:2204.11469.
- 27 Roger A Horn, Noah H Rhee, and So Wasin. Eigenvalue inequalities and equalities. *Linear Algebra and its Applications*, 270(1-3):29–44, 1998.
- 28 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Spectral independence, coupling with the stationary distribution, and the spectral gap of the glauher dynamics. *arXiv preprint*, 2021. arXiv:2105.01201.
- 29 Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit epsilon-balanced codes near the gilbert-varshamov bound. *arXiv preprint*, 2020. arXiv:2011.05500.
- 30 Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of ta-shma’s codes via splittable regularity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1527–1536, 2021.
- 31 Tali Kaufman and David Mass. High dimensional combinatorial random walks and colorful expansion. *arXiv preprint*, 2016. arXiv:1604.02947.
- 32 Tali Kaufman and David Mass. Local-to-global agreement expansion via the variance method. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 33 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Combinatorica*, pages 1–37, 2020.
- 34 Tali Kaufman and Ran J Tessler. Local to global high dimensional expansion and garland’s method for general posets. *arXiv preprint*, 2021. arXiv:2101.12621.
- 35 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.
- 36 Subhash Khot, Dor Minzer, Dana Moshkovitz, and Muli Safra. Small set expansion in the johnson graph. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 25, page 78, 2018.
- 37 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 576–589, 2017.
- 38 Subhash Khot and Nisheeth K Vishnoi. On the unique games conjecture. In *FOCS*, volume 5, page 3. Citeseer, 2005.
- 39 Anthony Leverrier and Gilles Zémor. Quantum tanner codes. *arXiv preprint*, 2022. arXiv:2202.13641.



- 40 Ting-Chun Lin and Min-Hsiu Hsieh. c3-local testable codes from lossless expanders. *arXiv preprint*, 2022. [arXiv:2201.11369](#).
- 41 Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. *arXiv preprint*, 2021. [arXiv:2103.11609](#).
- 42 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical ldpc codes. *arXiv preprint*, 2021. [arXiv:2111.03654](#).
- 43 Richard P Stanley. Differential posets. *Journal of the American Mathematical Society*, 1(4):919–961, 1988.
- 44 Khot Subhash, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601. IEEE, 2018.
- 45 Qing Zou. The q-binomial inverse formula and a recurrence relation for the q-catalan–qi numbers. *J. Math. Anal.*, 8(1):176–182, 2017.

## A Eposet Parameters and Regularity

In this section, we discuss connections between notions of regularity, the averaging operators, and eposet parameters. To start, we’ll show that downward and middle regularity (which are defined only on adjacent levels of the poset) imply extended regularity between any two levels.

► **Proposition 41.** *Let  $(X, \Pi)$  be a  $d$ -dimensional regular measured poset. Then for any  $i < k \leq d$ , there exist regularity constant  $R(k, i)$  such that for any  $x_k \in X(k)$ , there are exactly  $R(k, i)$  elements  $x_i \in X(i)$  such that  $x_k > x_i$ .*

**Proof.** Given any element  $x_k \in X(k)$ , downward regularity promises there are exactly  $\prod_{j=i+1}^k R(j)$  unique chains  $x_k < x_{k-1} < \dots < x_{i+1} < x_i$ . By middle regularity, any fixed  $x_i \in X(i)$  which appears in this fashion appears in exactly  $m(k, i)$  chains. Noting that  $x_i < x_k$  if and only if  $x_i$  appears in such a chain, the total number of  $x_i < x_k$  must be exactly  $R(k, i) = \frac{\prod_{j=i+1}^k R(j)}{m(k, i)}$ . ◀

A similar argument shows that regularity allows the up operators to compose in the natural way.

► **Proposition 42.** *Let  $(X, \Pi)$  be a  $d$ -dimensional regular measured poset. Then for any  $i < k \leq d$  we have:*

$$U_i^k f(x_k) = \frac{1}{R(k, i)} \sum_{x_i < x_k} f(x_i)$$

**Proof.** Expanding out  $U_i^k f(y)$  gives:

$$U_i^k f(x_k) = \frac{1}{\prod_{j=i+1}^k R(j)} \sum_{x_{k-1} < x_k} \dots \sum_{x_i < x_{i+1}} f(x_i)$$

The number of times each  $x_i$  appears in this sum is exactly the number of chains starting at  $x_k$  and ending at  $x_i$ , so by middle regularity:

$$\begin{aligned} \frac{1}{\prod_{j=i+1}^k R(j)} \sum_{x_{k-1} < x_k} \dots \sum_{x_{i+1} < x_i} f(x_i) &= \frac{m(k, i)}{\prod_{j=i+1}^k R(j)} \sum_{x_i < x_k} f(x_i) \\ &= \frac{1}{R(k, i)} \sum_{x_i < x_k} f(x_i). \end{aligned} \quad \blacktriangleleft$$

## 16:20 Eigenstripping, Spectral Decay, and Edge-Expansion on Posets

We'll now take a look at the connection between eposet parameters and regularity. It is convenient to first start with a lemma stating that non-laziness is equivalent to bounding the maximum transition probability of the lower walk.

► **Lemma 43.** *Let  $(X, \Pi)$  be a  $d$ -dimensional measured poset. Then for any  $0 < i \leq d$ , the maximum laziness of the lower walk is also the maximum transition probability:*

$$\max_{\sigma \in X(i)} \{ \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\sigma \} = \max_{\sigma, \tau \in X(i)} \{ \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\tau \}.$$

**Proof.** Assume that  $\tau \neq \sigma$ . Then the transition probability from  $\tau$  to  $\sigma$  is exactly

$$\begin{aligned} \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\tau &= \frac{\pi_\tau(\sigma \setminus \tau)}{R(i, i-1)} \\ &\leq \frac{1}{R(i, i-1)} \sum_{\tau < \sigma} \pi_\tau(\sigma \setminus \tau) \\ &= \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\sigma, \end{aligned}$$

which implies the result. ◀

We now prove our two claims relating the eposet parameters to regularity.

▷ **Claim 44.** Let  $(X, \Pi)$  be a  $d$ -dimensional  $(\delta, \gamma)$ -eposet. Then for every  $1 \leq k \leq d$  and  $0 \leq i \leq k$ , the following approximate relation between the eposet and regularity parameters holds:

$$\lambda_i(N_k^1) \in \frac{R(k, i)}{R(k+1, i)} \pm (\gamma_{k-i}^k + R(k, i) \delta_{k-i}^k \gamma)$$

where we recall  $\lambda_i(N_k^1) = 1 - \prod_{j=i}^k \delta_j$ .

**Proof.** We require a refinement of [16, Claim 8.8] given in [6, Lemma A.1]:<sup>7</sup>

$$D_{k+1} U_i^{k+1} = (1 - \delta_{k-i}^k) U_i^k + \delta_{k-i}^k U_{i-1}^k D_i + \sum_{j=-1}^{k-i-1} U_{k-j-1}^k \Gamma_j U_i^{k-j-1} \quad (3)$$

where  $\sum \|\Gamma_j\| \leq \gamma_{k-i}^k$ . The idea is now to examine the “laziness” of the two sides of this equality. In other words, given a starting  $k$ -face  $\tau$ , what is the probability that the resulting  $i$ -face  $\sigma$  satisfies  $\sigma < \tau$ ?

To start, we'll argue that the laziness of the lefthand side is exactly  $\frac{R(k, i)}{R(k+1, i)}$ . This follows from noting that there are  $R(k, i)$   $i$ -faces  $\sigma$  satisfying  $\sigma < \tau$ , and  $R(k+1, i)$  options after taking the initial up-step of the walk to  $\tau' > \tau$ . After the down-steps, the resulting  $i$ -face is uniformly distributed over these  $R(k+1, i)$  options  $\sigma < \tau'$ , and since every  $\sigma < \tau < \tau'$ , all original  $R(k, i)$  lazy options are still viable after the up-step to  $\tau'$ .

Analyzing the right-hand side is a bit trickier. The initial term  $(1 - \delta_{k-i}^k) U_i^k$  is completely lazy, so it contributes exactly  $(1 - \delta_{k-i}^k) = \lambda_i(N_k^1)$ . We'll break the second term into two steps: walking from  $X(k)$  to  $X(i)$  via  $U_i^k$ , then from  $X(i)$  to  $X(i)$  via the lower walk  $U_{i-1} D_i$ . Starting at a  $k$ -face  $\tau$ , notice that after applying the down step  $U_i^k$  we are uniformly spread over  $\sigma < \tau$ . Computing the laziness then amounts to asking what the probability of staying

<sup>7</sup> Formally the result is only stated for simplicial complexes in [6], but the same proof holds for eposets.

in this set is after the application of  $UD$ , which one can naively bound by the maximum transition probability times the set size  $R(k, i)$ . By non-laziness, the maximum transition probability is at most  $\gamma$  (see Lemma 43).

The third term can be handled similarly. The first down step  $U_{k-j-1}^k$  spreads  $\tau$  evenly across  $\sigma < \tau$  in  $X(k-j-1)$ . The resulting  $i$ -face  $\sigma'$  after applying  $\Gamma_j U_i^{k-j-1}$  is less than  $\tau$  if and only if the intermediary  $(k-j-1)$ -face after applying  $\Gamma_j$  is less than  $\tau$ , which is bounded by the spectral norm  $\|\Gamma_j\|$ .<sup>8</sup>

Putting everything together, since both sides of Equation (3) must have equivalent laziness, we get that  $\lambda_i(N_k^1)$  must be within  $\sum \|\Gamma_j\| + \delta_{k-i}^k R(k, i)\gamma$  as desired.  $\triangleleft$

Claim 19 and Theorem 22 can both be proving an analogous theorem for the upper walk.

$\triangleright$  **Claim 45 (Regularity and Upper Walk Spectrum).** Let  $(X, \Pi)$  be a  $d$ -dimensional  $(\delta, \gamma)$ -eposet. Then for any  $j \leq i \leq k < d$ , we have:

$$\lambda_j(N_i^{k-i}) \in \frac{R(i, j)}{R(k, i)} \pm \text{err},$$

where  $\text{err} \leq O\left(\frac{i^4 k^2 R_{\max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$ .

*Proof.* This follows almost immediately from the fact that  $i$ -links lie almost entirely on the  $i$ th eigenstrip (Lemma 31). In particular, it is enough to examine the expansion of  $i$ -links with respect to the upper canonical walk  $N_i^{k-i}$ . On the one hand, for any  $j \leq i$  and  $\tau \in X(j)$  we have:

$$\begin{aligned} \bar{\Phi}(X_\tau^i) &= \frac{\langle \mathbb{1}_{X_\tau^i}, N_i^{k-i} \mathbb{1}_{X_\tau^i} \rangle}{\langle \mathbb{1}_{X_\tau^i}, \mathbb{1}_{X_\tau^i} \rangle} \\ &= \frac{\langle U_j^k \mathbb{1}_\tau, U_j^k \mathbb{1}_\tau \rangle}{\langle U_j^i \mathbb{1}_\tau, U_j^i \mathbb{1}_\tau \rangle} \\ &= \frac{R(i, j)^2 \langle \mathbb{1}_{X_\tau^k}, \mathbb{1}_{X_\tau^k} \rangle}{R(k, i)^2 \langle \mathbb{1}_{X_\tau^i}, \mathbb{1}_{X_\tau^i} \rangle} \\ &= \frac{R(i, j) \langle \mathbb{1}_\tau, \mathbb{1}_\tau \rangle}{R(k, i) \langle \mathbb{1}_\tau, \mathbb{1}_\tau \rangle} \\ &= \frac{R(i, j)}{R(k, i)}. \end{aligned}$$

where we have applied the fact that  $\langle X_\tau^\ell, X_\tau^\ell \rangle = R(\ell, j) \langle \mathbb{1}_\tau, \mathbb{1}_\tau \rangle$ . On the other hand, by Lemma 31 we also have that:

$$\begin{aligned} \bar{\Phi}(\mathbb{1}_{X_\tau^i}) &= \frac{1}{\langle \mathbb{1}_{X_\tau^i}, \mathbb{1}_{X_\tau^i} \rangle} \sum_{\ell=0}^i \langle \mathbb{1}_{X_\tau^i}, N_i^{k-i} \mathbb{1}_{X_\tau^i, \ell} \rangle \\ &\in \frac{1}{\langle \mathbb{1}_\tau, \mathbb{1}_\tau \rangle} \sum_{\ell=0}^i \lambda_j(N_i^{k-i}) \langle \mathbb{1}_{X_\tau^i}, \mathbb{1}_{X_\tau^i, \ell} \rangle + c\gamma \\ &\in \lambda_j(N_i^{k-i}) \frac{\langle \mathbb{1}_\tau, \mathbb{1}_{\tau, j} \rangle}{\langle \mathbb{1}_\tau, \mathbb{1}_\tau \rangle} + \sum_{j=0}^i \text{err}_1 \\ &\in \lambda_j(N_i^{k-i}) + \text{err}_2 \end{aligned}$$

where as in the proof of Theorem 29,  $c, \text{err}_1, \text{err}_2 \leq O\left(\frac{i^4 k^2 R_{\max}}{\delta_{i-j}^i(1-\delta_{j-1})}\gamma\right)$ .  $\triangleleft$

<sup>8</sup> We note that  $\Gamma_j$  is not stochastic, but it is self-adjoint and an easy exercise to see that the analogous reasoning still holds.

Claim 19 follows immediately from observing that  $\rho_i^k = \lambda_i(N_i^{k-i})$  (by Proposition 18). Theorem 22 follows from observing that  $\hat{N}_i^{k-i}$  and  $\check{N}_k^{k-i}$  have the same approximate eigenvalues (similarly by Proposition 18).

Finally we close out the section by discussing the connection between non-laziness and a variant of eposets called local-spectral expanders [34].

► **Definition 46** (Local-Spectral Expander [19, 34]). *A  $d$ -dimensional measured poset  $(X, \Pi)$  is a  $\gamma$ -local-spectral expander if the graph underlying every link<sup>9</sup> of dimension at most  $d - 2$  is a  $\gamma$ -spectral expander.<sup>10</sup>*

Under suitable regularity conditions (see [34]), local-spectral expansion is equivalent to the notion of expanding posets used in this work. A simple argument shows that  $\gamma$ -local-spectral expanders are  $\gamma$ -non-lazy.

► **Lemma 47.** *Let  $(X, \Pi)$  be a  $d$ -dimensional  $\gamma$ -local-spectral expander, and  $0 < i < d$ . The laziness of the lower walk on level  $i$  is at most:*

$$\max_{\sigma \in X(i)} \left\{ \frac{\langle \mathbb{1}_\sigma, U_{i-1} D_i \mathbb{1}_\sigma \rangle}{\langle \mathbb{1}_\sigma, \mathbb{1}_\sigma \rangle} \right\} \leq \gamma.$$

**Proof.** Through direct computation, the laziness probability of the lower walk at  $\sigma \in X(i)$  is exactly

$$\frac{\langle \mathbb{1}_\sigma, U_{i-1} D_i \mathbb{1}_\sigma \rangle}{\langle \mathbb{1}_\sigma, \mathbb{1}_\sigma \rangle} = \frac{1}{R(i, i-1)} \sum_{\tau \leq \sigma} \pi_\tau(\sigma \setminus \tau)$$

It is therefore enough to argue that  $\pi_\tau(\sigma \setminus \tau) \leq \gamma$ , as the graph underlying the link  $X_\tau$  is a  $\gamma$ -spectral expander. Recall that an equivalent formulation of this definition states that:

$$\|A_\tau - U D_\tau\| \leq \gamma,$$

where  $A_\tau$  is the standard (non-lazy upper) walk and  $U D_\tau$  is the lower walk on the graph underlying  $X_\tau$ . This implies that the weight of any vertex  $v$  in the graph is at most  $\gamma$ , as:

$$\frac{\langle \mathbb{1}_v, U D_\tau \mathbb{1}_v \rangle}{\langle \mathbb{1}_v, \mathbb{1}_v \rangle} = \frac{\langle \mathbb{1}_v, (U D_\tau - A_\tau) \mathbb{1}_v \rangle}{\langle \mathbb{1}_v, \mathbb{1}_v \rangle} \leq \|A_\tau - U D_\tau\| \leq \gamma$$

where we have used the fact that  $A_\tau$  is non-lazy by definition. Since  $\pi_\tau(\sigma \setminus \tau)$  is exactly the weight of the vertex  $\sigma \setminus \tau$  in  $X_\tau$ , this completes the proof. ◀

## B Deferred Proofs

**Proof of Lemma 35.** We follow the structure and notation of [1, Lemma 4.11]. Assume that the canonical walk starts at a subspace  $V \in X(k)$ , and walks up to  $W \in X(k+j)$ . We wish to analyze the probability that upon walking back down to level  $k$ , a subspace  $V'$  with intersection  $k-i$  is chosen, i.e.  $\dim(V \cap V') = k-i$ . Let such an event be denoted  $\mathcal{E}_i(W)$ . It follows from elementary  $q$ -combinatorics (see e.g. [10, Lemma 9.3.2]) that

$$\Pr_{V' \subset W}[\mathcal{E}_i(W) \mid W] = q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q},$$

<sup>9</sup> Here the link of  $\tau$  is not just its top level faces, but the complex given by taking this set, removing  $\tau$  from each face, and downward closing.

<sup>10</sup> A graph is a  $\gamma$ -spectral expander if its weighted adjacency matrix has no non-trivial eigenvalues greater than  $\gamma$  in absolute value.

where  $V' \in X(k)$  is drawn uniformly from the  $k$ -dimensional subspaces of  $W$ . To relate this process to the swap walk  $S_k^i$ , note that while the swap walk (as defined) only walks up to  $X(k+i)$ , walking up to  $X(k+j)$  and conditioning on intersection  $i$ , a process called the  $i$ -swapping  $j$ -walk by [1], is exactly the same due to symmetry (via the regularity condition, see [Proposition 4.9 of [1]] for a more detailed explanation). Thus consider the  $i$ -swapping  $j$ -walk, and let  $T'_i$  denote the variable standing for the subspace chosen by the walk. Conditioned on picking the same  $W$  as the canonical walk in its ascent, we may relate  $T'_i$  to the canonical walk:

$$\Pr[T'_i = T \mid W] = \Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)]$$

We may now decompose the canonical walk by intersection size:

$$\begin{aligned} N_k^j(V, T) &= \sum_{i=0}^j \sum_{W \in X(k+j)} \Pr[W] \Pr[\mathcal{E}_i(W) \mid W] \Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)] \\ &= \sum_{i=0}^j \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \mathbb{E}_{W \supset V} [\Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)]] \\ &= \sum_{i=0}^j \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \mathbb{E}_{W \supset V} [\Pr[T'_i = T \mid W]] \\ &= \sum_{i=0}^j \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \Pr[T'_i = T] \\ &= \sum_{i=0}^j \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} S_k^i(V, T). \end{aligned}$$

From this point, the claim can be obtained by applying a  $q$ -binomial inversion theorem (Theorem 2.1 of [45]), see the full version for details.  $\blacktriangleleft$

**Proof of Proposition 39.** For  $W \in X(d-i)$ , consider the co-link  $\bar{X}_W = \{V \in X(k) : V \subset W\}$ . For simplicity, let  $S := \bar{X}_W$ . The density of  $S$  in any  $j$ -link  $X_V$  is:

$$\alpha_j = \frac{(q^{d-i-j}-1) \dots (q^{d-k+1-i}-1)}{(q^{d-j}-1) \dots (q^{d-k+1}-1)} = q^{-i(k-j)} + o_{q,d}(1).$$

The idea is now to examine the (non)-expansion of the co-link with respect to the lower walk  $U_{k-1}D_k$ . By direct computation, the probability of returning to  $\bar{X}_W$  after moving to a  $(k-1)$ -dimensional subspace is exactly:

$$\bar{\Phi}(\bar{X}_W) = \frac{q^{d-i} - q^{k-1}}{q^d - q^{k-1}} = q^{-i} \pm q^{-\Omega(d)} \quad (4)$$

By Proposition 18, the approximate eigenvalues of the lower walk are

$$\lambda_j = \frac{q^{k-j} - 1}{q^k - 1} = q^{-j} - O(q^{-k})$$

Since a dimension- $i$  co-link has no projection onto levels  $i+1$  through  $k$ , it also holds that:

$$\bar{\Phi}(\bar{X}_W) = \frac{1}{\langle \mathbb{1}_S, \mathbb{1}_S \rangle} \sum_{j=0}^i q^{-j} \langle \mathbb{1}_S, \mathbb{1}_{S,j} \rangle - O(q^{-k})$$

## 16:24 Eigenstripping, Spectral Decay, and Edge-Expansion on Posets

for large enough  $q, d$ . Combined with Equation (4), there exists a universal constant  $c'$  such that for large enough  $q$  and  $d$ ,  $\mathbb{1}_{\bar{X}_W}$  cannot have more than a  $\frac{c'}{q}$  fraction of its mass on levels 1 through  $i - 1$ . Finally, noticing that  $\binom{k}{i}_q \alpha_i = 1 + o_q(1)$ , we obtain

$$\frac{\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle}{\langle \mathbb{1}_S, \mathbb{1}_S \rangle} \geq \frac{q - c'}{q} \geq c \binom{k}{i}_q \alpha_i$$

since the latter is strictly bounded away from 1 for large enough  $q$ . This completes the result since  $\bar{X}_W$  is  $(\alpha_i, i)$ -pseudorandom.  $\blacktriangleleft$

# Accelerating Polarization via Alphabet Extension

Iwan Duursma   



University of Illinois Urbana-Champaign, IL, USA

Ryan Gabrys   

University of California San Diego, CA, USA

Venkatesan Guruswami   

University of California, Berkeley, CA, USA

Ting-Chun Lin  

University of California San Diego, CA, USA

Hon Hai (Foxconn) Research Institute, Taipei, Taiwan

Hsin-Po Wang   

University of California San Diego, CA, USA

---

## Abstract

Polarization is an unprecedented coding technique in that it not only achieves channel capacity, but also does so at a faster speed of convergence than any other technique. This speed is measured by the “scaling exponent” and its importance is three-fold. Firstly, estimating the scaling exponent is challenging and demands a deeper understanding of the dynamics of communication channels. Secondly, scaling exponents serve as a benchmark for different variants of polar codes that helps us select the proper variant for real-life applications. Thirdly, the need to optimize for the scaling exponent sheds light on how to reinforce the design of polar code.

In this paper, we generalize the binary erasure channel (BEC), the simplest communication channel and the protagonist of many polar code studies, to the “tetrahedral erasure channel” (TEC). We then invoke Mori–Tanaka’s  $2 \times 2$  matrix over  $\mathbb{F}_4$  to construct polar codes over TEC. Our main contribution is showing that the dynamic of TECs converges to an almost-one-parameter family of channels, which then leads to an upper bound of 3.328 on the scaling exponent. This is the first non-binary matrix whose scaling exponent is upper-bounded. It also polarizes BEC faster than all known binary matrices up to  $23 \times 23$  in size. Our result indicates that expanding the alphabet is a more effective and practical alternative to enlarging the matrix in order to achieve faster polarization.

**2012 ACM Subject Classification** Mathematics of computing → Coding theory; Theory of computation → Error-correcting codes

**Keywords and phrases** polar code, scaling exponent

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.17

**Category** RANDOM

**Related Version** Full Version: <https://arxiv.org/abs/2207.04522> [14]

**Funding** Ryan Gabrys: NSF CCF-2107346.

Venkatesan Guruswami: NSF CCF-2210823 and Simons Investigator Award.

Hsin-Po Wang: NSF CCF-1764104.

## 1 Introduction

A fundamental question at the center of the theory of communication is whether we can fully utilize a noisy channel to transmit information. In modern terminology, can error correcting codes achieve channel capacity? The answer is positive; in fact, multiple code constructions do so. Among them, polar code is a special one as it achieves capacity faster than any other known code.



© Iwan Duursma, Ryan Gabrys, Venkatesan Guruswami, Ting-Chun Lin, and Hsin-Po Wang; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 17; pp. 17:1–17:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Polar coding was invented by Arikan around 2008 [2]. During that time, Arikan was experimenting with channel combining and splitting. By treating two independent binary channels as a single quaternary channel (combining) and tasking ourselves with guessing certain linear combinations of the inputs (splitting), he synthesized two channels, denoted by  $W^\perp$  and  $W^\circ$ , out of the original channel  $W$ . Arikan realized that, when combining and splitting is applied recursively, the channels undergo an intriguing dynamic that ultimately results in most synthetic channels being either almost noiseless or extremely noisy. This is *channel polarization*, the first ingredient underlying polar codes.

The second ingredient of polar codes, also given by Arikan in said seminal paper, is the relation between the dynamic of synthetic channels and the construction and performance of the code. Arikan's insight was that synthetic channels that become almost noiseless can be used to transmit information bits, and synthetic channels that become extremely noisy can be "frozen" to some fixed values. The rate at which we communicate meaningful bits is then the proportion of synthetic channels that are almost noiseless. So, whether we can achieve channel capacity becomes a problem of counting the numbers of good and bad synthetic channels.

It then became apparent, perhaps even appealing, that one can study the dynamic of synthetic channels by means of stochastic processes. Take the *binary erasure channel* (BEC) as an example. Let  $W$  be  $\text{BEC}(\varepsilon)$ , the BEC with erasure probability  $\varepsilon$ , where  $0 < \varepsilon < 1$ . The channels  $W^\perp$  and  $W^\circ$  are  $\text{BEC}(2\varepsilon - \varepsilon^2)$  and  $\text{BEC}(\varepsilon^2)$ , respectively. A process  $\{H_n\}_n$  is thus defined by having  $H_0 := \varepsilon$  and  $H_{n+1} := 2H_n - H_n^2$  or  $H_n^2$  with equal probability. It can be shown that if

$$\mathbb{P}\{H_n \leq f(n)\} = 1 - H_0 - g(n),$$

where  $f, g > 0$  are functions in  $n$ , then there is a polar code of length  $2^n$ , miscommunication probability  $2^n f(n)$ , and gap to capacity  $g(n)$ .

It was at this point that the study of polar codes branched. On one branch, called the *error exponent regime*,  $g$  is a constant and the asymptotics of  $f$  is examined. On the other branch, called the *scaling exponent regime*,  $f$  is a constant<sup>1</sup> and the asymptotics of  $g$  is examined. On the error exponent branch, it was shown that  $f(n)$  is roughly  $\exp(-e^{\beta n})$ , where  $\beta > 0$  is a constant depending on the matrix used in the code construction. The task of determining  $\beta$  for each matrix has been fully resolved; interested readers are referred to [3, 26, 22, 33].

On the scaling exponent branch, making progress is harder and slower. For BECs, [21, 25] managed to estimate that  $g(n) \approx 2^{-n/3.527}$ . For binary memoryless symmetric (BMS) channels, it was first shown that  $g(n) < 2^{-n/\mu}$  for some constant  $0 < \mu < \infty$  [20]. This makes polar codes the only known code family that converges to capacity at a polynomial rate in the block length. More realistic estimates of  $\mu$  were given later:  $3.553 < \mu$  [18],  $3.579 < \mu < 6$  [23],  $\mu < 5.702$  [17],  $\mu < 4.714$  [31], and very recently  $\mu < 4.63$  [49]. Now that we know the  $\mu$  for polar code and the optimal value being  $\mu \approx 2$  for random code [4, 24, 37], the discrepancy begs the question: Can one modify polar code to reach a smaller scaling exponent?

<sup>1</sup> Not always; sometimes  $f \rightarrow 0$  but only exponentially fast in  $n$ . Note that  $2^n f(n)$ , the upper bound on the miscommunication probability, is allowed to exceed 1, so the corresponding code can be meaningless. Yet the asymptotics of  $g$  capture the behaviors of other meaningful codes.



The answer is positive: Arikan used the matrix  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  (this is called the *kernel* in literature) to combine and split channels. Instead, one can use a larger matrix, for instance

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

to combine and split channels. In [16, 50, 46, 45, 5, 28], binary matrices ranging from  $3 \times 3$  to  $64 \times 64$  are deployed and the scaling exponents over BECs are estimated. The best scaling exponent up to every matrix size is plotted in Figure 2. There are also meta-asymptotic results stating that  $\mu \approx 2$  can be achieved using larger and larger matrices. This statement was proved over  $q$ -ary erasure channels [36], binary erasure channels [15], all BMS channels [19], and finally discrete memoryless channels [48].

As much as we want to lower polar code's scaling exponent, there is one caveat that renders large matrices impractical: the smallest matrix whose scaling exponent is strictly better than  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  is the  $8 \times 8$  matrix above. Using this matrix takes twice more time to decode (estimate based on the method of [9]), whereas the benefit we gain is that  $\mu$  slightly decreases from 3.627 to 3.577. As the matrix gets larger and deviates more from the tensor powers of  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ , the time complexity grows drastically. For this reason, it is unlikely that we will ever see polar code based on large matrices (unless it is for other concerns [6]).

Large matrix aside, many other techniques emerge with empirical evidence that they improve polar code – concatenation, cyclic redundancy check, and list decoder to name a few. But none of them sees a proof of improvements in the scaling exponent; in fact, quite the opposite was reported [30]. So we are back to the starting point where we want to improve polar codes' scaling exponent while minimizing the complexity penalty.

One approach that seems promising, albeit very little is known due to its innate technical difficulty, is to use a non-binary input alphabet. This line of research started from Şaşoğlu [42, 41, 13], wherein the goal was to find at least one way to polarize arbitrary finite alphabets regardless of the speed. In particular, the usual matrix  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  is known to polarize prime fields. Later, Sahebi-Pradhan [40] and Park-Barg [35] showed that  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  cannot polarize non-prime fields. Then, Mori-Tanaka [33] classified all matrices that can polarize finite fields (i.e., the alphabet size must be a prime power). One step forward, Nasser [34] classified all binary operators (i.e., bivariate functions) that can polarize arbitrary finite alphabets. In [7, 8], the authors showed that, for any polarizing matrix over prime fields, one has  $\mu < \infty$ . In [48], the authors showed that  $\mu \approx 2$  is reachable over arbitrary finite alphabets.

Why is a non-binary input alphabet attractive? There are at least three reasons. First, modulation<sup>2</sup>: For quadrature amplitude modulation (QAM) and amplitude and phase-shift keying (APSK), a constellation point is more likely to be confused with constellation points nearer to it. A non-binary channel models this proximity relation more naturally than a series of correlated binary channels do [44, 11]. Second, two-stage polarization: If we weakly-polarize a binary channel with  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ , treat two binary channels as a quaternary channel, and strongly-polarize the quaternary channel with the  $4 \times 4$  Reed-Solomon matrix, we can improve the asymptotics of  $f(n)$  from  $\exp(-2^{0.5n})$  to  $\exp(-2^{0.5731n})$  [38] (see also [1, 12]). Third, and most importantly, scaling exponent: Several works have observed that non-binary matrices of the form  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$  just polarize faster than  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  [51, 29, 43]. Could it be that the non-binary scaling exponents are smaller?

<sup>2</sup> Modulation means translating digital signals to analog signals. A digital signal will be mapped to a point on the complex plane, which represents a sine wave with certain amplitude and phase; such a point is called a constellation point, the union of all points a constellation diagram.

Consider [39]’s technique that uses  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  to polarize non-binary channels; their result has an implication that non-binary channels’ scaling exponent is at least as good as binary channels’. In this paper, we aim to answer the question of whether the former is strictly better than the latter. By defining a toy model that contains a pair of BECs as a special case and estimating the scaling exponent of  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$ , we provide a proof of concept result that an expansion in alphabet size does result in an improvement in scaling exponent. Recall that BECs form a one-parameter family and that this property makes its scaling behavior easy to analyze. This paper’s overall strategy is to show that the descendants of a quaternary channel converge to an almost-one-parameter family; we then analyze the scaling behavior of this family and conclude the following.

► **Theorem 1 (main theorem).** *Treating a pair of BECs as a quaternary channel, the  $2 \times 2$  matrix  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$  over  $\mathbb{F}_4$  induces a scaling exponent less than 3.451. Here,  $\omega^2 + \omega + 1 = 0$ .*

This paper is organized as follows. Section 2 reviews the essence of polar code. Section 3 defines tetrahedral erasure channels (TECs), defines balanced TECs to be those that possess some symmetry, and defines edge-heavy TECs to be those that will be polarized faster. Section 4 defines serial combination and parallel combination that will be used to polarize TECs. Section 5 shows that unbalanced TECs tend to become very close to balanced TECs, so it suffices to consider the speed of polarization of the latter. Section 6 shows that edge-light TECs tend to become very close to edge-heavy TECs, so it suffices to consider the speed of polarization of the latter. Section 7 shows the speed of polarization of a generic TEC is faster than the classical BEC.

## 2 Polar Code

Readers who are familiar with polar code can safely skip this section. This section serves a simplified, high-level summary of classical polar code. More details are found in [47, Chapter 2]. We assume BEC throughout the section.

Let  $X \in \mathbb{F}_2$  be a random variable following the uniform distribution. Let  $Y \in \mathbb{F}_2 \cup \{?\}$  be a random variable with  $\mathbb{P}\{Y = X\} = 1 - \varepsilon$  and  $\mathbb{P}\{Y = ?\} = \varepsilon$ . Here,  $\varepsilon \in [0, 1]$  is called the *erasure probability*. The pair  $(X|Y)$  is called a *binary erasure channel* (BEC) and denoted by  $\text{BEC}(\varepsilon)$ . The entropy  $H(\text{BEC}(\varepsilon)) = H(X|Y) = \varepsilon$  is defined through Shannon’s mean.

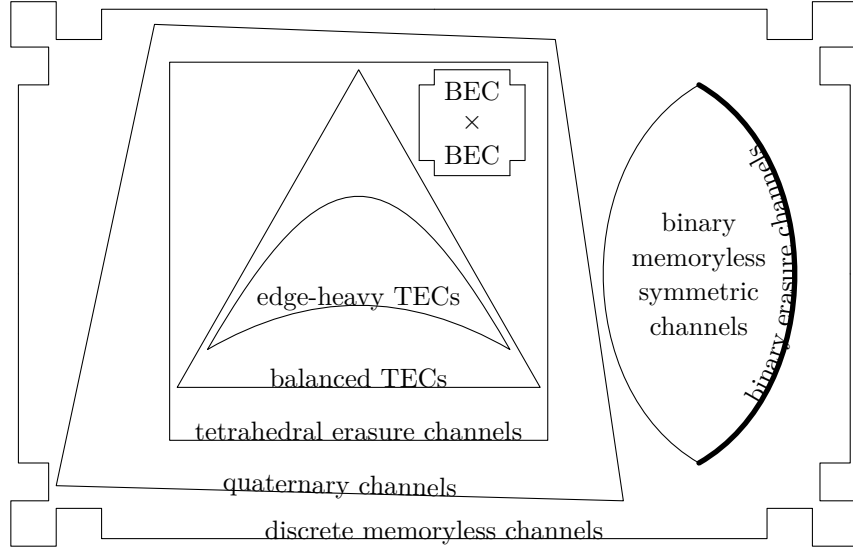
Let  $(X_1|Y_1)$  and  $(X_2|Y_2)$  be two iid copies of  $\text{BEC}(\varepsilon)$ . Define the serial combination  $\text{BEC}(\varepsilon)^\top$  to be  $(X_1 + X_2|Y_1, Y_2)$ . That is, what do we know about  $X_1 + X_2$  when given  $Y_1$  and  $Y_2$ ? One sees that it is information theoretically equivalent to  $\text{BEC}(2\varepsilon - \varepsilon^2)$ . Define the parallel combination  $\text{BEC}(\varepsilon)^\circ$  to be  $(X_1|Y_1, Y_2, X_1 + X_2)$ . That is, what do we know about  $X_1$  when given  $Y_1, Y_2$ , and  $X_1 + X_2$ ? One sees that it is information theoretically equivalent to  $\text{BEC}(\varepsilon^2)$ .

Serial and parallel combinations apply recursively. A polar code of block length  $2^n$  consists of a subset of strings  $\mathcal{I} \subseteq \{\top, \circ\}^n$ . In this code, a synthetic channel

$$\left( \dots \left( (\text{BEC}(\varepsilon)^{c_1})^{c_2} \right) \dots \right)^{c_n} \quad (1)$$

will be used to transmit useful information iff  $(c_1, c_1, \dots, c_n) \in \mathcal{I}$ . The code rate of this polar code is  $|\mathcal{I}|/2^n$ . The exact miscommunication probability of this polar code is hard to find, but has an upper bound of

$$\sum_{\mathcal{I}} H \left( \left( \dots \left( (\text{BEC}(\varepsilon)^{c_1})^{c_2} \right) \dots \right)^{c_n} \right).$$



■ **Figure 1** The Euler diagram of channels featured in this paper. The cross is the set of pairs of BECs; it will converge to the set of balanced TECs (Section 5). The balanced TECs will then converge to edge-heavy TECs (Section 6). And then edge-heavy TECs polarize faster than BECs. Note that BECs are a one-parameter family of extreme BMS channels, hence the thick curve.

To define a good  $\mathcal{I}$ , choose a function  $f(n)$  and collect all strings  $(c_1, c_1, \dots, c_n) \in \{\square, \circ\}^n$  such that  $H(\text{formula (1)})$  is less than  $f(n)$ . The fact that the erasure probabilities undergo simple evolutions  $\varepsilon \mapsto 2\varepsilon - \varepsilon^2$  and  $\varepsilon \mapsto \varepsilon^2$  motivates the following stochastic process: define  $\{H_n\}_n$  by initial value  $H_0 := \varepsilon$  and evolution rule  $H_{n+1} := 2H_n - H_n^2$  or  $H_n^2$  with equal probability. Then the code rate  $|\mathcal{I}|/2^n$  coincides with  $\mathbb{P}\{H_n \leq f(n)\}$ . The gap to capacity  $g(n) := 1 - H_0 - |\mathcal{I}|/2^n = 1 - H_0 - \mathbb{P}\{H_n \leq f(n)\}$  is thus motivated.

In a way, the study of polar code over BEC is the study of the cdf of  $H_n$ , with emphasis put on the hard threshold at  $1 - H_0$ . Abusing the same logic, this paper is a study of a stochastic process  $\{W_n\}_n$  that lives in  $[0, 1]^5 \cap \{p + q + r + s + t = 1\}$ , which happens to have peculiar implications in coding theory.

### 3 A New Channel Model

We are to define a type of quaternary channels in this section. This should be the smallest possible set of quaternary channels that meet the following: (a) it should model a pair of BECs as a special case; and (b) it should be closed under pre-processing the input using invertible linear transformations.

#### 3.1 Tetrahedral erasure channel

Let the input alphabet be  $\mathbb{F}_2^2$ ; and we assume the uniform input distribution throughout the paper. For any input  $(x_1, x_2) \in \mathbb{F}_2^2$ , the output will be in  $(\mathbb{F}_2 \cup \{?\})^2$  and assume one of the following five erasure patterns:

- $(x_1, x_1 + x_2, x_2)$  with probability  $p$ ;
- $(x_1, ?, ?)$  with probability  $q$ ;
- $(?, x_1 + x_2, ?)$  with probability  $r$ ;
- $(?, ?, x_2)$  with probability  $s$ ;
- $(?, ?, ?)$  with probability  $t$ .

Here we call  $p, q, r, s, t$  the *subspace erasure probabilities* and they sum to 1. Such a channel is denoted by  $\text{TEC}(p, q, r, s, t)$ . For brevity, we say a TEC outputs  $(x_1, x_2)$ , outputs  $x_1$ , outputs  $x_1 + x_2$ , outputs  $x_2$ , and outputs nothing to represent the five erasure patterns.

A TEC can be related to a tetrahedron whose vertices are  $(0, 0, 0)$ ,  $(1, 1, 0)$ ,  $(1, 0, 1)$ , and  $(0, 1, 1)$ . Outputting  $(x_1, x_2)$  corresponds to the vertex  $(x_1, x_1 + x_2, x_2)$ . Outputting  $x_1$  corresponds to the edge  $(x_1, x_1, 0) - (x_1, 1 - x_1, 1)$ . Outputting nothing corresponds to the tetrahedron per se. That is to say, a TEC takes a vertex as an input and outputs the same vertex with probability  $p$ , outputs an edge attached to that vertex with probability  $q + r + s$ , and output the entire tetrahedron with probability  $t$ .

There is another way to interpret a TEC. Consider  $\mathbb{F}_4$  and let  $\omega$  be a primitive element therein. A TEC takes  $x := x_1\omega + x_2 \in \mathbb{F}_4$  as an input and outputs  $x$ ,  $\text{tr}(x)$ ,  $\text{tr}(\omega x)$ ,  $\text{tr}(x/\omega)$ , or nothing, each with probability  $p, q, r, s$ , and  $t$ . Here,  $\text{tr}: \mathbb{F}_4 \rightarrow \mathbb{F}_2$  is the field trace. It is the matrix trace if we use the matrices  $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ ,  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ,  $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$  to represent  $0, 1, \omega, 1 + \omega \in \mathbb{F}_4$ .

TEC is not an ad hoc channel that we happen to know how to deal with. It relates to other channels that have been discussed in literature.

► **Proposition 2.** *The “ $q$ -ary erasure channel with erasure probability  $\varepsilon$ ” [32, 36], when  $q = 4$ , is a TEC of the form  $\text{TEC}(1 - \varepsilon, 0, 0, 0, \varepsilon)$ .*

► **Proposition 3.** *When transmitting two bits  $x_1$  and  $x_2$  through  $\text{BEC}(\delta)$  and  $\text{BEC}(\varepsilon)$ , respectively, the outputs can be simulated by  $\text{TEC}((1 - \delta)(1 - \varepsilon), (1 - \delta)\varepsilon, 0, \delta(1 - \varepsilon), \delta\varepsilon)$ .*

The proofs are trivial. The propositions imply that any scaling exponent estimate for TEC immediately generalizes to 4-ary erasure channels and BECs.

### 3.2 Channel functionals

The *conditional entropy* (hereafter *entropy*) of a TEC is defined by the following; it is meant to be compatible with Shannon’s definition:

$$H(\text{TEC}(p, q, r, s, t)) := \frac{q + r + s}{2} + t.$$

The *edge mass* of a TEC is defined by the following; it measures the “polarizability” of a TEC:

$$E(\text{TEC}(p, q, r, s, t)) := q + r + s.$$

The *Quetelet index* of a TEC  $W$  is defined by

$$Q(W) := \frac{E(W)}{H(W)(1 - H(W))}.$$

Clearly,  $0 \leq E(W) \leq 2 \min(H(W), 1 - H(W))$  and  $0 \leq Q(W) \leq 4$ . We call a TEC  $W$  *edge-heavy* if  $Q(W) \geq 2\sqrt{7} - 4$ . Adolphe Quetelet invented the body mass index that determines if a person is overweight or underweight. Here, we use Quetelet index to determine if a TEC possesses too much edge mass (easy to polarize) or too little (hard to polarize).

A TEC is *balanced* if  $q = r = s$ . Put it another way, the edges of the tetrahedron weigh the same. It is not hard to see that  $H$  and  $E$  uniquely determine a balanced TEC by

$$p = 1 - H(W) - \frac{E(W)}{2}, \quad q = r = s = \frac{E(W)}{3}, \quad t = H(W) - \frac{E(W)}{2}.$$

The *moment of inertia* of a TEC is defined by

$$A(\text{TEC}(p, q, r, s, t)) := (q - r)^2 + (r - s)^2 + (s - q)^2.$$

A TEC is balanced iff its moment of inertia vanishes. See also the “symmetric over the product” condition in [10] and the “equidistance” condition in [42].

## 4 Channel Synthesis

TECs can be serially combined or parallelly combined as in the theory of density evolution [27].

### 4.1 Serial combination

Let  $U := \text{TEC}(p, q, r, s, t)$  and  $V := \text{TEC}(p', q', r', s', t')$  be two TECs. The *serial combination* of  $U$  and  $V$  is defined to be the task of guessing  $(u_1 + v_1, u_2 + v_2)$  given the output of inputting  $(u_1, u_2)$  into  $U$  and the output of inputting  $(v_1, v_2)$  into  $V$ . Let us go over all 25 erasure patterns that are classified into five scenarios.

Scenario one –  $U$  outputs  $(u_1, u_2)$  and  $V$  outputs  $(v_1, v_2)$ : Now we know  $(u_1 + v_1, u_2 + v_2)$  in its entirety. This scenario happens with probability  $pp'$ .

Scenario two –  $U$  outputs  $u_1$  with or without  $u_2$ , and  $V$  outputs  $v_1$  with or without  $v_2$ , but either  $u_2$  or  $v_2$  is missing: In this case, we can infer  $u_1 + v_1$ , but we cannot infer  $u_2 + v_2$ . So this case feels like  $(x_1, x_2) := (u_1 + v_1, u_2 + v_2)$  underwent a TEC and only  $x_1$  went through. The probability that only  $x_1$  went through is  $pq' + qq' + qp'$ .

Scenario three –  $U$  outputs  $(u_1, u_2)$  or  $u_1 + u_2$ , and  $V$  outputs  $(v_1, v_2)$  or  $v_1 + v_2$ , but scenario one does not happen: For this case, we know neither  $u_1 + v_1$  nor  $u_2 + v_2$ . But we can infer  $(u_1 + v_1) + (u_2 + v_2)$ . So this case feels like  $(x_1, x_2) := (u_1 + v_1, u_2 + v_2)$  underwent a TEC and only  $x_1 + x_2$  went through. The probability that only  $x_1 + x_2$  went through is  $pr' + rr' + rp'$ .

Scenario four –  $U$  outputs  $u_2$  with or without  $u_1$ , and  $V$  outputs  $v_2$  with or without  $v_1$ , but either  $u_1$  or  $v_1$  is missing: In this case, we can infer  $x_2 := u_2 + v_2$  but not  $x_1 := u_1 + v_1$ . So this case feels like  $x_1$  is erased. This scenario happens with probability  $ps' + ss' + sp'$ .

Scenario five –  $U$  outputs one bit ( $u_1$  or  $u_1 + u_2$  or  $u_2$ ) and  $V$  outputs one bit ( $v_1$  or  $v_1 + v_2$  or  $v_2$ ) but the erasure patterns do not match; or at least one of  $U$  and  $V$  outputs nothing: We cannot infer  $u_1 + v_1$  because either  $u_1$  or  $v_1$  is missing. We cannot infer  $u_2 + v_2$  because either  $u_2$  or  $v_2$  is missing. We cannot infer  $(u_1 + v_1) + (u_2 + v_2)$ , either. So this case feels like both  $x_1 := u_1 + v_1$  and  $x_2 := u_2 + v_2$  are erased, so is  $x_1 + x_2$ . The probability that we learn nothing about  $(x_1, x_2)$  is  $(q + r + s)(q' + r' + s') - qq' - rr' - ss' + t + t' - tt'$ .

Note that these five scenarios correspond to the five erasure patterns in the definition of TEC. Denote by  $U \boxtimes V$  the serial combination of  $U$  and  $V$ ; it is a TEC with subspace erasure probabilities

$$U \boxtimes V := \text{TEC}(pp', pq' + qq' + qp', pr' + rr' + rp', ps' + ss' + sp', 1 - \text{the four to the left}).$$

### 4.2 Parallel combination

The *parallel combination* of  $U := \text{TEC}(p, q, r, s, t)$  and  $V := \text{TEC}(p', q', r', s', t')$  is defined to be the task of guessing  $(u_1, u_2)$  given  $(u_1 + v_1, u_2 + v_2)$  (the perfect output of  $U \boxplus V$ ), the result of feeding  $(u_1, u_2)$  into  $U$ , and the result of feeding  $(v_1, v_2)$  into  $V$ .

Denote by  $U \oplus V$  the parallel combination of  $U$  and  $V$ . One can go over its erasure scenarios like the previous subsection does. For instance, if  $U$  outputs  $u_1$  and  $V$  outputs  $v_1 + v_2$ , then we can infer  $v_1$  (using  $u_1$  and  $u_1 + v_1$ ), followed by  $v_2$  (using  $v_1$  and  $v_1 + v_2$ ), and finally  $u_2$  (using  $v_2$  and  $u_2 + v_2$ ); and hence we can completely recover  $u_1$  and  $u_2$ . Details omitted, it can be shown that  $U \oplus V$  is a TEC with subspace erasure probabilities

$$U \oplus V := \text{TEC}(1 - \text{the four to the right}, tq' + qq' + qt', tr' + rr' + rt', ts' + ss' + st', tt').$$

Note that there is a duality between  $\text{TEC}(p, q, r, s, t)$  and  $\text{TEC}(t, s, r, q, p)$  that keeps  $E$  as is, maps  $H$  to  $1 - H$ , and swaps parallel and serial combinations. The duality grants us the convenience of proving half of a theorem and the other half follows by symmetry.

### 4.3 Mori–Tanaka’s twisting kernel

A  $2 \times 2$  polarization *kernel*  $K$  over  $\mathbb{F}_4$  is defined with a “twist” as follows: For a pair of inputs  $u, v \in \mathbb{F}_4$ , let  $K$  be the linear transformation that reads  $(u, v) \mapsto (u + \omega v, v)$  or, equivalently,

$$\begin{bmatrix} u & v \end{bmatrix} \mapsto \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}.$$

This kernel was studied by Mori–Tanaka [33] and is shown to be polarizing. If we treat  $\mathbb{F}_4$  as  $\mathbb{F}_2^2$ , then  $K$  reads  $((u_1, u_2), (v_1, v_2)) \mapsto ((u_1 + v_1 + v_2, u_2 + v_1), (v_1, v_2))$  or, equivalently,

$$\begin{bmatrix} u_1 & u_2 & v_1 & v_2 \end{bmatrix} \mapsto \begin{bmatrix} u_1 & u_2 & v_1 & v_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix},$$

where  $u_1, u_2, v_1, v_2 \in \mathbb{F}_2$ . The kernel  $K$  combines two TECs  $U$  and  $V$  to synthesize  $U \boxtimes (V\omega)$  and  $U \boxplus (V\omega)$ , where  $V\omega$  is the channel that multiplies the input by  $\omega$  before feeding it into  $V$ . For brevity,  $W \boxtimes (W\omega)$  and  $W \boxplus (W\omega)$  are denoted by  $W^\perp$  and  $W^\circ$ , respectively.

Multiplying a TEC by  $\omega$  behaves like a rotation of order 3 (after all,  $\omega^3 = 1$  and it is rotating the tetrahedron). It maps  $\text{TEC}(p, q, r, s, t)$  to  $\text{TEC}(p, s, q, r, t)$ . If  $W$  is balanced, rotation does not alter it:  $W = W\omega$ . If it is not balanced, then the rotation helps mis-match  $q, r, s$  so that a large probability is paired with a small probability. More precisely,

$$\text{TEC}(p, q, r, s, t)^\perp := \text{TEC}(p^2, ps + sq + qp, pq + qr + rp, pr + rs + sp, 1 - \text{the other four}),$$

$$\text{TEC}(p, q, r, s, t)^\circ := \text{TEC}(1 - \text{the other four}, ts + sq + qt, tq + qr + rt, tr + rs + st, t^2).$$

Twisting makes it easier to reduce  $q, r$ , and  $s$  and redistribute the mass to  $p$  and  $t$ .

### 4.4 Channel process

For a TEC  $W$ , we call  $W^\perp$  the *serial-child* of  $W$  and  $W^\circ$  the *parallel-child* of  $W$ . Together, they are the *children* of  $W$ . The *descendants* of  $W$  are the children of  $W$  together with the descendants of the children of  $W$ . The  $n$ th-generation descendants of  $W$  are the  $(n - 1)$ th-generation descendants of the children of  $W$ ; the 0th is  $W$  itself.

When  $W$  is understood from the context, let  $W_0$  be  $W$ . For  $n$  a positive integer, let  $W_n$  be a random child of  $W_{n-1}$  with equal probability.

The common strategy used to estimate the scaling exponent concerns a concave function  $\psi: [0, 1] \rightarrow \mathbb{R}$  such that  $\psi(0) = \psi(1) = 0$  and is positive elsewhere. With  $\psi$ , one finds a  $0 < \mu < \infty$  such that

$$\frac{\psi(H(W^\perp)) + \psi(H(W^\circ))}{2\psi(H(W))} \leq 2^{-1/\mu}$$

With this “eigenvalue,” a routine argument [47, Sections 5.8–5.10] will show that

$$\mathbb{P}\{H(W_n) < \exp(-e^{n^{1/3}})\} > 1 - H(W_0) - 2^{-n/\mu}.$$

## 5 Unbalanced TEC Becomes Balanced

In this section, we argue that TECs undergoing the polarization process tend to become more balanced than before. We do so by showing that the moments of inertia are decreasing.

► **Theorem 4** (uniform loss of inertia).  $A(W^\square), A(W^\circ) \leq A(W)(1 - A(W)/3)$  for any TEC  $W$ .

A proof of the theorem is in Appendix A.1 of the full version [14]. Now the recurrence relation  $A(W_{n+1}) \leq A(W_n)(1 - A(W_n)/3)$  is equivalent to  $A(W_{n+1}) - A(W_n) \leq -A(W_n)^2/3$  and analogous to the ordinary differential equation  $f'(n) \leq -f(n)^2/3$ . Solving it, we get  $f(n) = O(1/n)$ ; analogously,  $A(W_n) = O(1/n)$ .

► **Corollary 5** (ultimate loss of inertia). *Fix a TEC  $W$ , then  $A(W_n) = O(1/n)$  as  $n \rightarrow \infty$ .*

Another way to look at it is the average decay of  $A(W_n)$ .

► **Proposition 6** (average loss of inertia).  $A(W^\square) + A(W^\circ) \leq A(W)$  for any TEC  $W$ .

A proof of the proposition is in Appendix A.2 of the full version [14]. By the proposition,  $A(W) \geq A(W^\perp) + A(W^\circ) \geq A(W^{\perp\perp}) + A(W^{\perp\circ}) + A(W^{\circ\perp}) + A(W^{\circ\circ}) \geq A(W^{\perp\perp\perp}) \dots$ . Hence the expectation of  $A(W_n)$  over all  $W_n$  is at most  $A(W)/2^n$ .

Corollary 5 and Proposition 6 imply that any unbalanced TEC will promptly become very similar to a balanced one.<sup>3</sup> The speed of polarization of unbalanced TECs is thus dominated by that of balanced TECs. We now turn to the analysis of the polarization speed of balanced TECs.

## 6 Balanced TECs Hoard Edge Mass

In this section, we argue that the Quetelet index  $Q(W_n) := E(W_n)/H(W_n)(1 - H(W_n))$  of a sufficiently deep descendant is about 1.6. Put another way, there is a “trap” that constrains the relation between  $E(W_n)$  and  $H(W_n)$ .

Recall that a balanced TEC  $W$  is edge-heavy if  $Q(W) \geq 2\sqrt{7} - 4$ . Let  $\alpha := 2\sqrt{7} - 4 \approx 1.3$ .

► **Theorem 7** (trapping region). *If  $W$  is balanced and edge-heavy, then its children are edge-heavy.*

A proof of the theorem is in Appendix B.1 of the full version [14]. The theorem implies that all descendants of an edge-heavy are edge-heavy. For a TEC that is not edge-heavy, its descendants will still become “edge-heavier” by the following theorem.

► **Theorem 8** (attraction toward the trap). *Fix any  $\varepsilon > 0$ ; choose  $\delta := 3\varepsilon/8$ . Let  $W$  be any balanced TEC. We have that  $Q(W) \leq \alpha - \varepsilon$  implies  $Q(W^\square) \geq Q(W)(1 + H(W)\delta)$  and  $Q(W^\circ) \geq Q(W)(1 + (1 - H(W))\delta)$ .*

A proof of the theorem is in Appendix B.2 of the full version [14]. It is clear that the factors  $H(W)$  and  $1 - H(W)$  before  $\delta$  slow down the rate at which  $Q(W_n)$  approaches  $2\sqrt{7} - 4$ , especially when  $H(W)$  is close to 0 or 1, respectively. These factors cannot be optimized away. To see why, suppose that  $H(W) = x \approx 1$  and  $E(W) = y \approx 0$ . Then

<sup>3</sup> Note that Corollary 5 is a weak statement about every single descendant of  $W$ , while Proposition 6 implies a strong statement about  $A(W_n)$  averaged over all  $n$ th-generation descendants. Only Corollary 5 will be used later.



## 17:10 Accelerating Polarization via Alphabet Extension

$H(W^\circ)$  is about  $x^2 + O(y^2)$  and  $E(W^\circ)$  is about  $2xy + O(y^2)$ . Hence  $Q(W^\circ)$  is about  $2xy/x^2(1-x^2) \approx y/x(1-x) = Q(W)$ . That being the case, we would like to add that TECs whose Quetelet index can hardly be improved are already polarized, so we shall not worry about them. Besides, we can prove uniform attraction using Theorem 8.

► **Theorem 9** (uniform attraction). *Fix any  $\varepsilon > 0$ . For any balanced TEC  $W$  such that  $Q(W) \leq \alpha - \varepsilon$ , there exists an integer  $m > 0$  such that  $Q(W_n) \geq Q(W)(1 + \varepsilon/8)$  for all  $n \geq m$ .*

A proof of the theorem is in Appendix B.3 of the full version [14]. Uniform attraction means that every child is at least making some positive progress toward the trap. Small steps of the descendants accumulate to a giant leap of the family.

► **Corollary 10** (ultimate attraction). *For any  $\varepsilon > 0$  and any balanced TEC  $W$  such that  $Q(W) > 0$ , there exists an integer  $m > 0$  such that  $Q(W_n) \geq \alpha - \varepsilon$  for all  $n \geq m$ .*

**Proof.** Apply the uniform attraction theorem repeatedly. Every application improves the Quetelet index by a factor of  $1 + (\alpha - Q(W_n))/8$ . So after a finite number of applications the Quetelet index can be made  $\geq \alpha - \varepsilon$ . ◀

To summarize this and the previous section, we have two trends: unbalanced TECs tend to become balanced; and “edge-light” TECs tend to become edge-heavy.

The following proposition is a bound on Quetelet index in the opposite direction.

► **Proposition 11** (attraction on the other side). *Let  $W$  be a balanced TEC with  $Q(W) \leq 2$ . Then  $Q(W^\square) \leq 2$  and  $Q(W^\circ) \leq 2$ .*

Some comments on how to prove this proposition is in Appendix B.4 of the full version [14].

The following proposition gives a tighter trapping region than Theorem 8 and Proposition 11 do. A proof is omitted but similar to those of Theorem 8 and Proposition 11. For the optimal trapping region, see the discussion in Appendix D of the full version [14].

► **Proposition 12.** *Let  $f(x) := x(1-x)(1.66 - 0.38x(1-x))$ . Then  $E(W) \leq f(H(W))$  implies  $E(W^\square) \leq f(H(W^\square))$  and  $E(W^\circ) \leq f(H(W^\circ))$ . Let  $g(x) := x(1-x)(2 - 2x(1-x)/3)$ . Then  $E(W) \geq g(H(W))$  implies  $E(W^\square) \geq g(H(W^\square))$  and  $E(W^\circ) \geq g(H(W^\circ))$ .*

## 7 Edge-heavy TECs Polarize Faster

Let  $W$  be any balanced TEC with a fixed  $H(W) = x$  and a variable  $E(W) = y$ . Then  $H(W^\square) = 2x - x^2 + y^2/12$  is increasing in  $y$  and  $H(W^\circ) = x^2 - y^2/12$  is decreasing in  $y$ .

The monotonicity has two applications. Application one: If we know too little to lower bound  $Q(W)$ , we will upper bound  $H(W^\circ)$  using  $x^2$ . In this case, the speed of polarization is at least  $\mu \approx 3.627$ , the number induced by the standard polar code. Application two: If we know  $Q(W) \geq \alpha$ , we will upper bound  $H(W^\circ)$  using  $x^2 - (\alpha x(1-x))^2/12$ . This time,  $H(W^\circ)$  and  $H(W^\square)$  are more separated so the speed of polarization is strictly better than  $\mu \approx 3.627$ . Any positive  $\alpha$ , not necessarily  $2\sqrt{7} - 4$ , can improve the scaling. This is demonstrated by the following lemma that uses  $9/7$  in place of  $\alpha$ .

► **Lemma 13** (eigenfunction and eigenvalue). *Let  $\psi(x) := (x(1-x))^{0.697}(5 - \sqrt{x(1-x)})$ . For balanced TECs with  $Q(W) \geq 9/7$ ,*

$$\frac{\psi(H(W^\square)) + \psi(H(W^\circ))}{2\psi(H(W))} < 0.818.$$



Comments on how to verify the lemma is in Appendix C of the full version [14].

► **Theorem 14** (main theorem). *Consider a pair of BECs treated as a TEC, or consider any TEC where  $pqrst > 0$ . The  $2 \times 2$  matrix  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$  over  $\mathbb{F}_4$  induces a scaling exponent less than 3.451.*

**Proof.** Two iid copies of  $\text{BEC}(\varepsilon)$  can be seen as  $W := \text{TEC}((1-\varepsilon)^2, (1-\varepsilon)\varepsilon, 0, \varepsilon(1-\varepsilon), \varepsilon^2)$ . If  $\varepsilon$  is 0 or 1, there is nothing to prove. Suppose  $0 < \varepsilon < 1$ , then both  $W^\perp$  and  $W^\circ$  have five positive subspace erasure probabilities. (That is, their “ $p, q, r, s, t$ ” are all positive). The descendants of a TEC with five positive subspace erasure probabilities satisfy the same property. In particular, all descendants have positive Quetelet index.

Let  $W$  be a TEC whose descendants all have positive  $Q$ . By Corollary 5, it takes  $W$  a finite number of generations to become very similar to a balanced TEC. That is, for any  $\delta > 0$  there exists an  $m > 0$  such that  $A(W_m) < \delta$ . Although  $W_n$  is never balanced, what we proved about balanced TECs still hold for “almost-balanced” TECs up to a diminishing error term. So we may proceed as if  $W_n$  is balanced for  $n \geq m$ .

By Corollary 10, it takes another finite number of generations to become “almost edge-heavy.” In particular, there exists an  $m'$  such that  $Q(W_{m'}) \geq 9/7$  (note that  $9/7 \approx 1.286$  and  $2\sqrt{7} - 4 \approx 1.291$ ).

Before the  $m'$ th generation, the eigenvalues of the form

$$\frac{\psi(H(W_n^\Gamma)) + \psi(H(W_n^\circ))}{2\psi(H(W_n))}$$

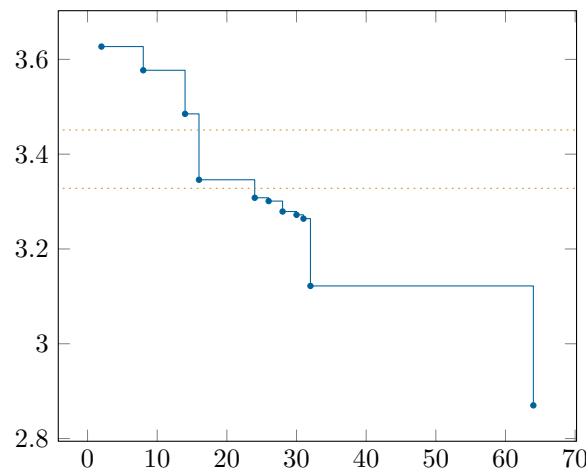
was less than 1. After the  $m'$ th generation, the eigenvalues of said form will be less than  $0.818 < 2^{-1/3.451}$ , by Lemma 13. As  $n$  goes to infinity, 0.818 dominates the overall scaling behavior. Hence  $W$ , and hence any BEC, enjoys scaling exponent less than 3.451. ◀

In the abstract, we claim that the scaling exponent of  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$  over TECs (and hence BECs) is  $< 3.328$ . This number will be derived in Appendix D of the full version [14] with more intense numerical calculations. In particular, there is a new trapping region that is bounded by two linear splines and is significantly smaller than the region bounded by  $ax(1-x)$  for  $a = 2\sqrt{7} - 4$  and 2; the attraction toward the new trap is witnessed by sampling TECs with low edge-mass. In Appendix E of the full version [14], we also examine the actual values of  $H(W_n)$  and its asymptotic behavior aligns with the estimate 3.328.

## 8 Conclusions

In this paper, we argue that  $\begin{bmatrix} 1 & 0 \\ \omega & 1 \end{bmatrix}$  polarizes BECs faster than  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  does. We first show that a pair of BECs will be transformed into balanced TECs. We then show that balanced TECs will be transformed into edge-heavy TECs. Finally, we show that edge-heavy TECs assume a better scaling exponent.

Our rigorous overestimate of the scaling exponent is 3.451; there is another overestimate of 3.328 with strong numerical evidence. Compared to Arıkan’s  $2 \times 2$  matrix with  $\mu \approx 3.627$ , Fazeli–Vardy’s  $8 \times 8$  matrix with  $\mu \approx 3.577$  [16], Trofimiuk–Trifonov’s  $16 \times 16$  matrix with  $\mu \approx 3.346$  [46], and Yao–Fazeli–Vardy’s  $32 \times 32$  matrix with  $\mu \approx 3.122$  [50], our result suggests that one should consider expanding the alphabet size prior to enlarging the matrix size. More precisely, the rigorous estimate is analogous to a  $15 \times 15$  binary matrix; the more accurate estimate is analogous to a  $20 \times 20$  binary matrix (see Figure 2).



■ **Figure 2** Horizontal axis: matrix size; vertical axis: scaling exponent of the best known matrix [16, 50, 46, 45, 5]. A matrix size will be skipped if no known matrix outruns all smaller matrices. Underlying channel is BEC. Our estimates 3.451 and 3.328 are marked as dotted lines.

## References

- 1 Fariba Abbasi, Hessam Mahdavi, and Emanuele Viterbo. Hybrid non-binary repeated polar codes. *IEEE Transactions on Wireless Communications*, pages 1–1, 2022. doi:10.1109/TWC.2022.3159807.
- 2 Erdal Arıkan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009. doi:10.1109/TIT.2009.2021379.
- 3 Erdal Arıkan and Emre Telatar. On the rate of channel polarization. In *2009 IEEE International Symposium on Information Theory*, pages 1493–1495, June 2009. doi:10.1109/ISIT.2009.5205856.
- 4 D. Baron, M.A. Khojastepour, and R.G. Baraniuk. How quickly can we approach channel capacity? In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, volume 1, pages 1096–1100 Vol.1, November 2004. doi:10.1109/ACSSC.2004.1399310.
- 5 Manan Bhandari, Ishan Bansal, and V. Lalitha. On the polarizing behavior and scaling exponent of polar codes with product kernels. In *2020 National Conference on Communications (NCC)*, pages 1–6, February 2020. doi:10.1109/NCC48643.2020.9056096.
- 6 Valerio Bioglio, Frédéric Gabry, Ingmar Land, and Jean-Claude Belfiore. Multi-kernel polar codes: Concept and design principles. *IEEE Transactions on Communications*, 68(9):5350–5362, September 2020. doi:10.1109/TCOMM.2020.3006212.
- 7 Jarosław Błasiok, Venkatesan Guruswami, Preetum Nakkiran, Atri Rudra, and Madhu Sudan. General strong polarization. *J. ACM*, 69(2), March 2022. doi:10.1145/3491390.
- 8 Jarosław Błasiok, Venkatesan Guruswami, and Madhu Sudan. Polar Codes with Exponentially Small Error at Finite Block Length. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.34.
- 9 Sarit Buzaglo, Arman Fazeli, Paul H. Siegel, Veeresh Taranalli, and Alexander Vardy. Permuted successive cancellation decoding for polar codes. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2618–2622, June 2017. doi:10.1109/ISIT.2017.8007003.

- 10 Eduardo Camps, Hiram H. López, Gretchen L. Matthews, and Eliseo Sarmiento. Polar decreasing monomial-cartesian codes. *IEEE Transactions on Information Theory*, 67(6):3664–3674, June 2021. doi:10.1109/TIT.2020.3047624.
- 11 Semih Cayci, Toshiaki Koike-Akino, and Ye Wang. Nonbinary polar coding for multilevel modulation. In *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2019.
- 12 Peiyao Chen, Baoming Bai, and Xiao Ma. Two-stage polarization-based nonbinary polar codes for 5g urllc, 2018. doi:10.48550/ARXIV.1801.08059.
- 13 Mao-Ching Chiu. Non-binary polar codes with channel symbol permutations. In *2014 International Symposium on Information Theory and its Applications*, pages 433–437, October 2014.
- 14 Iwan Duursma, Ryan Gabrys, Venkatesan Guruswami, Ting-Chun Lin, and Hsin-Po Wang. Accelerating polarization via alphabet extension, 2022. doi:10.48550/ARXIV.2207.04522.
- 15 Arman Fazeli, Hamed Hassani, Marco Mondelli, and Alexander Vardy. Binary linear codes with optimal scaling: Polar codes with large kernels. *IEEE Transactions on Information Theory*, 67(9):5693–5710, September 2021. doi:10.1109/TIT.2020.3038806.
- 16 Arman Fazeli and Alexander Vardy. On the scaling exponent of binary polarization kernels. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 797–804, September 2014. doi:10.1109/ALLERTON.2014.7028536.
- 17 Dina Goldin and David Burshtein. Improved bounds on the finite length scaling of polar codes. *IEEE Transactions on Information Theory*, 60(11):6966–6978, November 2014. doi:10.1109/TIT.2014.2359197.
- 18 Ali Goli, S. Hamed Hassani, and Rüdiger Urbanke. Universal bounds on the scaling behavior of polar codes. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1957–1961, July 2012. doi:10.1109/ISIT.2012.6283641.
- 19 Venkatesan Guruswami, Andrii Riazanov, and Min Ye. Arıkan meets shannon: Polar codes with near-optimal convergence to channel capacity. *IEEE Transactions on Information Theory*, pages 1–1, 2022. doi:10.1109/TIT.2022.3146786.
- 20 Venkatesan Guruswami and Patrick Xia. Polar codes: Speed of polarization and polynomial gap to capacity. *IEEE Transactions on Information Theory*, 61(1):3–16, January 2015. doi:10.1109/TIT.2014.2371819.
- 21 S. Hamed Hassani, Kasra Alishahi, and Rudiger Urbanke. On the scaling of polar codes: Ii. the behavior of un-polarized channels. In *2010 IEEE International Symposium on Information Theory*, pages 879–883, June 2010. doi:10.1109/ISIT.2010.5513585.
- 22 S. Hamed Hassani, Ryuhei Mori, Toshiyuki Tanaka, and Rüdiger L. Urbanke. Rate-dependent analysis of the asymptotic behavior of channel polarization. *IEEE Transactions on Information Theory*, 59(4):2267–2276, April 2013. doi:10.1109/TIT.2012.2228295.
- 23 Seyed Hamed Hassani, Kasra Alishahi, and Rüdiger L. Urbanke. Finite-length scaling for polar codes. *IEEE Transactions on Information Theory*, 60(10):5875–5898, October 2014. doi:10.1109/TIT.2014.2341919.
- 24 Masahito Hayashi. Information spectrum approach to second-order coding rate in channel coding. *IEEE Transactions on Information Theory*, 55(11):4947–4966, November 2009. doi:10.1109/TIT.2009.2030478.
- 25 Satish Babu Korada, Andrea Montanari, Emre Telatar, and Rüdiger Urbanke. An empirical scaling law for polar codes. In *2010 IEEE International Symposium on Information Theory*, pages 884–888, June 2010. doi:10.1109/ISIT.2010.5513579.
- 26 Satish Babu Korada, Eren Şaşoğlu, and Rüdiger Urbanke. Polar codes: Characterization of exponent, bounds, and constructions. *IEEE Transactions on Information Theory*, 56(12):6253–6264, December 2010. doi:10.1109/TIT.2010.2080990.
- 27 Ingmar Land and Johannes Huber. Information combining. *Foundations and Trends® in Communications and Information Theory*, 3(3):227–330, 2006. doi:10.1561/0100000013.

- 28 Liping Lin. On the construction of the kernel matrix by primitive bch codes for polar codes. *Communications and Network*, 14(1):23–35, 2021.
- 29 Guan-Chen Liu and Qi-Yue Yu. Non-binary polar coded system for the two-user multiple-access channel, 2021. doi:10.48550/ARXIV.2111.03839.
- 30 Marco Mondelli, S. Hamed Hassani, and Rüdiger L. Urbanke. Scaling exponent of list decoders with applications to polar codes. *IEEE Transactions on Information Theory*, 61(9):4838–4851, September 2015. doi:10.1109/TIT.2015.2453315.
- 31 Marco Mondelli, S. Hamed Hassani, and Rüdiger L. Urbanke. Unified scaling of polar codes: Error exponent, scaling exponent, moderate deviations, and error floors. *IEEE Transactions on Information Theory*, 62(12):6698–6712, December 2016. doi:10.1109/TIT.2016.2616117.
- 32 Ryuhei Mori and Toshiyuki Tanaka. Non-binary polar codes using reed-solomon codes and algebraic geometry codes. In *2010 IEEE Information Theory Workshop*, pages 1–5, August 2010. doi:10.1109/CIG.2010.5592755.
- 33 Ryuhei Mori and Toshiyuki Tanaka. Source and channel polarization over finite fields and reed-solomon matrices. *IEEE Transactions on Information Theory*, 60(5):2720–2736, May 2014. doi:10.1109/TIT.2014.2312181.
- 34 Rajai Nasser. An ergodic theory of binary operations—part i: Key properties. *IEEE Transactions on Information Theory*, 62(12):6931–6952, December 2016. doi:10.1109/TIT.2016.2616642.
- 35 Woomyoung Park and Alexander Barg. Polar codes for q-ary channels,  $q = 2^r$ . *IEEE Transactions on Information Theory*, 59(2):955–969, 2013. doi:10.1109/TIT.2012.2219035.
- 36 Henry D. Pfister and Rüdiger Urbanke. Near-optimal finite-length scaling for polar codes over large alphabets. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 215–219, July 2016. doi:10.1109/ISIT.2016.7541292.
- 37 Yury Polyanskiy, H. Vincent Poor, and Sergio Verdu. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, May 2010. doi:10.1109/TIT.2010.2043769.
- 38 Noam Presman, Ofer Shapira, and Simon Litsyn. Mixed-kernels constructions of polar codes. *IEEE Journal on Selected Areas in Communications*, 34(2):239–253, February 2016. doi:10.1109/JSAC.2015.2504278.
- 39 Constantin Runge, Thomas Wiegart, Diego Lentner, and Tobias Prinz. Multilevel binary polar-coded modulation achieving the capacity of asymmetric channels, 2022. arXiv:2202.04010.
- 40 Aria G. Sahebi and S. Sandeep Pradhan. Multilevel polarization of polar codes over arbitrary discrete memoryless channels. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1718–1725, September 2011. doi:10.1109/Allerton.2011.6120375.
- 41 Eren Şaçoğlu. Polar codes for discrete alphabets. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 2137–2141, July 2012. doi:10.1109/ISIT.2012.6283740.
- 42 Eren Şaçoğlu, Emre Telatar, and Erdal Arıkan. Polarization for arbitrary discrete memoryless channels. In *2009 IEEE Information Theory Workshop*, pages 144–148, October 2009. doi:10.1109/ITW.2009.5351487.
- 43 Valentin Savin. Non-binary polar codes for spread-spectrum modulations. In *2021 11th International Symposium on Topics in Coding (ISTC)*, pages 1–5, August 2021. doi:10.1109/ISTC49272.2021.9594166.
- 44 Mathis Seidl, Andreas Schenk, Clemens Stierstorfer, and Johannes B. Huber. Polar-coded modulation. *IEEE Transactions on Communications*, 61(10):4108–4119, October 2013. doi:10.1109/TCOMM.2013.090513.130433.
- 45 Grigori Trofimuk. Shortened polarization kernels. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, December 2021. doi:10.1109/GCWkshps52748.2021.9681982.

- 46 Grigorii Trofimiuk and Peter Trifonov. Window processing of binary polarization kernels. *IEEE Transactions on Communications*, 69(7):4294–4305, July 2021. doi:10.1109/TCOMM.2021.3072730.
- 47 Hsin-Po Wang. Complexity and second moment of the mathematical theory of communication, 2021. arXiv:2107.06420.
- 48 Hsin-Po Wang and Iwan M. Duursma. Polar codes’ simplicity, random codes’ durability. *IEEE Transactions on Information Theory*, 67(3):1478–1508, 2021.
- 49 Hsin-Po Wang, Ting-Chun Lin, Alexander Vardy, and Ryan Gabrys. Sub-4.7 scaling exponent of polar codes, 2022. doi:10.48550/ARXIV.2204.11683.
- 50 Hanwen Yao, Arman Fazeli, and Alexander Vardy. Explicit polar codes with small scaling exponent. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1757–1761, 2019. doi:10.1109/ISIT.2019.8849741.
- 51 Peihong Yuan and Fabian Steiner. Construction and decoding algorithms for polar codes based on  $2 \times 2$  non-binary kernels. In *2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, pages 1–5, December 2018. doi:10.1109/ISTC.2018.8625284.



# Sketching Distances in Monotone Graph Classes

Louis Esperet   

Univ. Grenoble Alpes, CNRS, Laboratoire G-SCOP, Grenoble, France

Nathaniel Harms   

University of Waterloo, Canada

Andrey Kupavskii 

Univ. Grenoble Alpes, CNRS, Laboratoire G-SCOP, Grenoble, France

Moscow Institute of Physics and Technology, Russia

Huawei R&D Moscow, Russia

---

## Abstract

We study the problems of adjacency sketching, small-distance sketching, and approximate distance threshold (ADT) sketching for monotone classes of graphs. The algorithmic problem is to assign random sketches to the vertices of any graph  $G$  in the class, so that adjacency, exact distance thresholds, or approximate distance thresholds of two vertices  $u, v$  can be decided (with probability at least  $2/3$ ) from the sketches of  $u$  and  $v$ , by a decoder that does not know the graph. The goal is to determine when sketches of *constant size* exist.

Our main results are that, for monotone classes of graphs: constant-size adjacency sketches exist if and only if the class has bounded arboricity; constant-size small-distance sketches exist if and only if the class has bounded expansion; constant-size ADT sketches imply that the class has bounded expansion; any class of constant expansion (i.e. any proper minor closed class) has a constant-size ADT sketch; and a class may have arbitrarily small expansion without admitting a constant-size ADT sketch.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures; Theory of computation  $\rightarrow$  Graph algorithms analysis; Theory of computation  $\rightarrow$  Distributed algorithms; Theory of computation  $\rightarrow$  Sketching and sampling

**Keywords and phrases** adjacency labelling, informative labelling, distance sketching, adjacency sketching, communication complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.18

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2202.09253>

**Funding** *Louis Esperet:* supported by the French ANR Projects GATO (ANR-16-CE40-0009-01), GrR (ANR-18-CE40-0032), TWIN-WIDTH (ANR-21-CE48-0014-01), and by LabEx PERSYVAL-lab (ANR-11-LABX-0025).

**Acknowledgements** We thank Gwenaél Joret for many helpful discussions. We thank Viktor Zamaraev for leading us to Corollary 3.6 and carefully proofreading our manuscript. We thank Alexandr Andoni for a helpful discussion and for sharing with us the manuscript [10]. We thank Renato Ferreira Pinto Jr. and Sebastian Wild for comments on the presentation of this article.

## 1 Introduction

A common type of problem, with many theoretical and practical uses in computer science, is to assign short *labels* to each of  $n$  elements of a space, so that certain “local” information can be deduced from the labels. The Boolean hypercube graph of size  $n = 2^d$ , with vertex set  $\{0, 1\}^d$  and edges  $(x, y)$  where  $x, y \in \{0, 1\}^d$  differ on exactly 1 coordinate, has the trivial but useful property that one can assign to each vertex  $x \in \{0, 1\}^d$  a label of  $d = \log n$  bits,



© Louis Esperet, Nathaniel Harms, and Andrey Kupavskii;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 18; pp. 18:1–18:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

so that any function  $f(x, y)$  on vertex pairs can be decided given only the labels of  $x$  and  $y$ . Sometimes, the label size can be improved drastically by allowing *randomized* labels, which we refer to as *sketches*. For example:

1. Adjacency in the hypercube can be computed (with probability at least  $2/3$ ) from sketches of constant size (which follows from the Hamming distance communication protocol [42]);
2. Distinguishing between  $\text{dist}(x, y) \leq r$  and  $\text{dist}(x, y) > r$  can be done with sketches of size depending only on  $r$  (which also follows from the Hamming distance protocol);
3. Distinguishing between  $\text{dist}(x, y) \leq r$  and  $\text{dist}(x, y) > \alpha r$  (for constant  $\alpha > 1$ ) can be done with sketches of size independent of  $r$  and  $n$  [51].

We call these *adjacency sketches*, *small-distance sketches*, and *approximate distance threshold (ADT) sketches*, respectively (see Section 1.2 for formal definitions). It is natural to ask which classes of graphs, other than the hypercubes, admit similarly efficient sketches. Motivated by a connection between communication complexity, sketching, and graph labelling schemes, recent work [40] asked which *hereditary* classes of graphs admit constant-size adjacency sketches, and also gave some examples of constant-size (i.e. independent of the number of vertices) small-distance sketches, including for planar graphs, answering a question of [38]. Sketches for deciding  $\text{dist}(x, y) \leq r$  vs.  $\text{dist}(x, y) > \alpha r$  are well-studied, and characterizing the metrics which admit this type of sketch is a well-known open problem [62, 10, 46, 61], but little is known about the natural case of path-distance metrics in graphs.

We study the relationships between these three types of sketches for the important special case of *monotone* classes of graphs. A class of graphs is a set of (labelled<sup>1</sup>) graphs closed under isomorphism. It is *hereditary* if it is closed under taking induced subgraphs, and *monotone* if it is closed under taking subgraphs. Monotone graph classes are ubiquitous: typical examples include minor-closed classes, graphs avoiding some subgraph  $H$ , or graphs with bounded chromatic number.

In this paper, we completely determine the monotone graph classes which admit constant-size adjacency sketches and constant-size (i.e. independent of the number of vertices) small-distance sketches, and show that constant-size (i.e. independent of the number of vertices and the parameter  $r$ ) ADT sketches imply the existence of constant-size small-distance sketches. We show that the classes which admit constant-size adjacency sketches are exactly the classes with bounded arboricity, and the classes which admit constant-size small-distance sketches are exactly the classes with bounded *expansion*<sup>2</sup>. Classes which admit constant-size ADT sketches must also have bounded expansion, and any class with constant expansion (i.e. any proper minor-closed class) has a constant-size ADT sketch, but on the other hand a class can have expansion growing arbitrarily slowly and yet does *not* admit a constant-size ADT sketch. We describe these results in more detail below.

## 1.1 Motivation & Prior Work

Labelling schemes and sketches are important primitives for distributed computing, streaming, communication, data structures for approximate nearest neighbors, and even classical algorithms (see e.g. [47, 30, 63, 60, 20], and [4, 44, 10, 61, 11] and references therein). As such, a great deal of research has been done on finding other spaces having nice sketching and labelling properties.

<sup>1</sup> Standard terminology is that a *labelled*  $n$ -vertex graph is one with vertex set  $[n]$ ; not to be confused with *informative labelling schemes*.

<sup>2</sup> We mean bounded expansion in the sense of sparsity theory [57], which is distinct from expansion in the context of expander graphs.



One direction of research investigates the metric spaces which admit approximate distance threshold (ADT) sketches, of the third type described above, as defined in [62]. This is a well-known open problem in sublinear algorithms (see e.g. [10, 46, 61]). Here,  $n$  points  $X \subseteq \mathcal{X}$  in a metric space  $(\mathcal{X}, \text{dist})$ , should be assigned random sketches  $\text{sk} : X \rightarrow \{0, 1\}^*$  such that  $\text{dist}(x, y) \leq r$  or  $\text{dist}(x, y) \geq \alpha r$  can be determined (with probability at least  $2/3$ ) from  $\text{sk}(x)$  and  $\text{sk}(y)$ . The goal is to obtain sketches whose size depends only on  $\alpha$ . This problem is fairly well-understood when the metric is a norm: there is a constant-size sketch for the  $\ell_p$  (quasi-)norm, for any  $0 < p \leq 2$  [44], so any metric that can be embedded into such an  $\ell_p$  is sketchable; conversely, sketching a norm is equivalent to embedding it into  $\ell_{1-\varepsilon}$  [11]. Outside of norms, the problem is less well-understood: there are sketchable metrics that are not embeddable into  $\ell_{1-\varepsilon}$  [48].

Another direction of research investigates the classes  $\mathcal{F}$  of graphs that admit (deterministic) labelling schemes for various functions, generally called *informative labelling schemes* [60]. The most well-studied labelling schemes are for adjacency, introduced in [47, 55]. The main open problem is to identify the hereditary classes of graphs that admit adjacency labelling schemes of size  $O(\log n)$ . A solution was suggested in [47] and later conjectured in [63], but recently refuted in a breakthrough of [41], leaving the problem wide open. *Randomized adjacency labelling* (i.e. *adjacency sketching*) was studied in [24, 38, 40]. It was observed in [38, 40] that a constant-size sketch implies an  $O(\log n)$  labelling scheme, as desired in the above open problem, and it was further observed in [40] that the set of hereditary graph classes which admit constant-size adjacency sketches is equivalent to the set of Boolean-valued communication problems that admit constant-cost public-coin protocols, whose structure is unknown [37]. This raises the following question, which was the main motivation of [40]:

► **Question 1.** *Which hereditary classes of graphs admit constant-size adjacency sketches?*

Perhaps the next most commonly studied graph labelling problem is *distance labelling* [31], where the goal is to compute  $\text{dist}(x, y)$  from the labels (see e.g. [7, 9, 25, 32]). Intermediate between distance and adjacency labelling is the decision version of distance labelling: for given  $r$ , decide whether  $\text{dist}(x, y) \leq r$  from the labels. We call this *small-distance labelling*, following the terminology of [6, 29]. For  $r = 1$ , this coincides with adjacency labelling. The natural generalization of constant-size adjacency sketches is to ask for small-distance sketches whose size depends only on  $r$ ; it was shown in [38] that such sketches exist for trees, and in [40] that they exist for any Cartesian product graphs and any *stable*<sup>3</sup> class of bounded twin-width (including, for example, planar graphs or any proper minor-closed class; see [27]).

► **Question 2.** *Which hereditary classes of graphs admit small-distance sketches whose size depends only on  $r$ ?*

It is common to weaken distance labelling to *approximate distance labelling* [28], where the goal is to approximate  $\text{dist}(x, y)$  up to a constant factor (see e.g. [65, 1, 8]). The decision version is to distinguish, for a given  $r$ , between  $\text{dist}(x, y) \leq r$  and  $\text{dist}(x, y) > \alpha r$ ; we will call this problem  $\alpha$ -*approximate distance threshold (ADT) labelling and sketching*. This is a similar formulation as the distance sketching problem mentioned above, with the  $n$  points from the metric space  $\mathcal{X}$  being replaced with a size  $n$  graph from a class  $\mathcal{F}$ . Despite significant interest in distance sketching and labelling, the only prior work explicitly relating the two, or studying *randomized ADT labelling*, appears to be the unpublished manuscript [10] (although there is extensive literature on the related problem of embedding graph metrics into normed

<sup>3</sup> See [40] for a discussion of stability, which is not necessary for the current paper.

spaces [54, Chapter 15]; embedding planar graphs into  $\ell_1$  with constant distortion is a major open problem [35]). This raises the following question, which is a special case of the open problem of identifying sketchable metrics:

► **Question 3.** *Which classes of graphs admit constant-size ADT sketches?*

It holds by definition (see definitions below) that a small-distance sketchable class  $\mathcal{F}$  is adjacency sketchable, but the relationships between other types of sketching are otherwise unclear, *a priori*. It seems reasonable to suspect that these three types of sketching require similar conditions on the graph class  $\mathcal{F}$ ; so we ask:

► **Question 4.** *What is the relationship between adjacency, small-distance, and ADT sketching?*

## 1.2 Our Results

In this paper, we resolve Questions 1, 2, and 4 for *monotone* classes of graphs, and make progress towards Question 3. The sketches we obtain usually do not assume that the classes under consideration are monotone, but our lower bounds crucially rely on this assumption. We first formally define the main three types of *sketchability* that we are concerned with. We will generalize these definitions in Section 2.2. For a graph class  $\mathcal{F}$ , we say:

1.  $\mathcal{F}$  admits an *adjacency sketch of size  $s(n)$*  if there is a function  $D : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$  such that  $\forall G \in \mathcal{F}$  with size  $n$ , there is a random function  $\text{sk} : V(G) \rightarrow \{0, 1\}^{s(n)}$  satisfying

$$\forall x, y \in V(G) : \quad \Pr [D(\text{sk}(x), \text{sk}(y)) = 1 \iff x, y \text{ are adjacent}] \geq 2/3.$$

$\mathcal{F}$  is *adjacency sketchable* if it admits an adjacency sketch of constant size.

2.  $\mathcal{F}$  admits a *small-distance sketch of size  $s(n, r)$*  if for every  $r \in \mathbb{N}$  there is a function  $D_r : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$  such that  $\forall G \in \mathcal{F}$  with size  $n$ , there is a random function  $\text{sk} : V(G) \rightarrow \{0, 1\}^{s(n, r)}$  satisfying

$$\forall x, y \in V(G) : \quad \Pr [D_r(\text{sk}(x), \text{sk}(y)) = 1 \iff \text{dist}_G(x, y) \leq r] \geq 2/3.$$

$\mathcal{F}$  is *small-distance sketchable* if it admits a small-distance sketch of size independent of  $n$ .

3. For constant  $\alpha > 1$ ,  $\mathcal{F}$  admits an  $\alpha$ -*ADT sketch of size  $s(n)$*  if for every  $r \in \mathbb{N}$  there is a function  $D_r : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$  such that  $\forall G \in \mathcal{F}$  with size  $n$ , there is a random function  $\text{sk} : V(G) \rightarrow \{0, 1\}^{s(n)}$  satisfying

$$\begin{aligned} \forall x, y \in V(G) : \quad & \text{dist}(x, y) \leq r \implies \Pr [D_r(\text{sk}(x), \text{sk}(y)) = 1] \geq 2/3 \\ & \text{dist}(x, y) > \alpha r \implies \Pr [D_r(\text{sk}(x), \text{sk}(y)) = 0] \geq 2/3. \end{aligned}$$

For a constant  $\alpha > 1$ , we say that  $\mathcal{F}$  is  $\alpha$ -*ADT sketchable* if  $\mathcal{F}$  admits an  $\alpha$ -ADT sketch with size independent of  $n$ .  $\mathcal{F}$  is *ADT sketchable* if there is a constant  $\alpha > 1$  such that  $\mathcal{F}$  is  $\alpha$ -ADT sketchable.

Our results imply the following hierarchy, which answers Question 4 for monotone classes of graphs. Let ADJ be the adjacency sketchable monotone graph classes, SD the small-distance sketchable monotone graph classes, and ADT the ADT sketchable monotone graph classes. Then

$$\text{ADT} \subsetneq \text{SD} \subsetneq \text{ADJ}.$$

That  $\text{SD} \subseteq \text{ADJ}$  follows by definition, and  $\text{SD} \neq \text{ADJ}$  is witnessed by the arboricity-2 graphs (as observed in [38]). Our contribution to this hierarchy is  $\text{ADT} \subsetneq \text{SD}$  (which does not necessarily hold for non-monotone classes), a complete characterization of the sets SD and ADJ, and some results towards a characterization of ADT.

### 1.2.1 Adjacency Sketching

We resolve Question 1 for monotone classes by showing that they are adjacency sketchable if and only if they have bounded arboricity. A graph  $G$  has arboricity  $k$  if its edges can be partitioned into  $k$  forests. A class  $\mathcal{F}$  has arboricity  $k$  if all graphs  $G \in \mathcal{F}$  have arboricity at most  $k$ . If there exists some constant  $k$  such that  $\mathcal{F}$  has arboricity  $k$ , we say  $\mathcal{F}$  has *bounded arboricity*.

► **Theorem 1.1.** *Let  $\mathcal{F}$  be a monotone class of graphs. Then  $\mathcal{F}$  is adjacency sketchable if and only if  $\mathcal{F}$  has bounded arboricity.*

All proofs for adjacency sketching are in Section 3. Using standard random hashing and the adjacency labelling scheme of [47], it is easy to see that any class of bounded arboricity is adjacency sketchable; this was stated explicitly in [38, 40] (the latter giving slightly improved sketch size). We prove the converse for monotone classes (which does not hold for hereditary classes in general [40]). We use the probabilistic method to find a subgraph of small discrepancy in any class of unbounded arboricity, inspired by the recent proof of [36, 37] that refuted the main conjecture of [40], and we find that the subgraphs of the hypercube are an easier-to-define counterexample to the conjecture of [40]).

It is interesting that the hashing-based sketch uses randomization only to compute EQUALITY subproblems; i.e. it can be simulated by a constant-cost *deterministic* communication protocol with access to a unit-cost EQUALITY oracle. This type of sketch is called *equality-based* in [40]. Equality-based sketches imply some structural properties of the graph class, such as the strong Erdős-Hajnal property [37]. Recent work has studied the power of the EQUALITY oracle and found that it does not capture the full power of randomization [15, 37, 40]; in particular, the Boolean hypercubes (and any Cartesian product graphs) are adjacency sketchable, but not with an equality-based sketch [37, 40]. Our result shows that EQUALITY captures the power of randomization for sketching monotone classes of graphs. In fact, it is only necessary to compute a *disjunction* of equality checks, which we think of as the simplest possible type of sketch.

We remark that sketches (especially small-distance or ADT sketches) which compute a disjunction of equality checks can be used to obtain *locality-sensitive hashes*, a widely-used algorithmic tool introduced in [45]. Almost all of our positive results are of this type. See Remark 2.6.

### 1.2.2 Small-Distance Sketching

We answer Question 2 by proving that the monotone graph classes that are small-distance sketchable are exactly those with *bounded expansion* (as in [57]; see our Definition 2.2). Informally, bounded expansion means that the edge density of a graph increases only as a function of  $r$  when contracting subgraphs of radius  $r$  into a single vertex. Many graph classes of theoretical and practical importance have bounded expansion, including bounded-degree graphs, proper minor-closed graph classes, and graphs of bounded genus [57], along with many random graph models and real-world graphs [17].

To state our theorem, we briefly describe another type of sketch that generalizes small-distance sketching, called *first-order* sketching. A graph class  $\mathcal{F}$  is *first-order sketchable* if any first-order (FO) formula  $\phi(x, y)$  over the vertices and edge relation of the graph (with two free variables whose domain is the set of vertices) is sketchable (see Section 2.2). This type of sketch was introduced in [40] and generalizes small-distance sketching, along with (for example) testing whether vertices  $x, y$  belong to a subgraph isomorphic to some fixed graph  $H$ . We show that, for monotone graph classes, first-order sketchability is equivalent to small-distance sketchability. All proofs for small-distance sketching are in Section A.

► **Theorem 1.2.** *Let  $\mathcal{F}$  be a monotone class of graphs. Then the following are equivalent:*

1.  $\mathcal{F}$  is small-distance sketchable;
2.  $\mathcal{F}$  is first-order sketchable;
3.  $\mathcal{F}$  has bounded expansion.

The implications (3)  $\implies$  (2)  $\implies$  (1) do not require monotonicity. (2)  $\implies$  (1) holds by definition. The proof of (3)  $\implies$  (2) is straightforward, but relies on a structural result of [26] whose proof is highly technical. We actually get the stronger result that any *first-order transduction* of a class with bounded expansion is first-order sketchable, which improves the results of [40]. It was proved in [40], using structural results of [27], that any stable class of bounded twin-width is first-order sketchable. A stable class has bounded twin-width if and only if it is a transduction of a class of bounded sparse twin-width [27]. Every class of bounded sparse twin-width has bounded expansion, but the converse does not hold (e.g. for cubic graphs) [14], so our result generalizes the result of [40]. It essentially follows from using the structural results of [26] instead of [27].

Our proof of (1)  $\implies$  (3) (Section A.4) requires our proof of Theorem 1.1 and some results in sparsity theory [50, 57]. We actually prove a stronger statement: for any monotone class  $\mathcal{F}$ , the existence of a sketch for deciding  $\text{dist}(x, y) \leq r$  vs.  $\text{dist}(x, y) > 5r - 1$ , with size depending only on  $r$ , implies bounded expansion. Under a conjecture of Thomassen [64], we can replace the constant 5 with any arbitrarily large constant; see the remark after Conjecture A.17. Note that, even with a constant-factor gap between distance thresholds, this problem is distinct from ADT sketching, since the small-distance sketch size is allowed to depend on  $r$ . If we could replace the constant 5 with any arbitrarily large constant, this would immediately imply  $\text{ADT} \subseteq \text{SD}$ .

We also present a more direct proof of (3)  $\implies$  (1), without going through first-order sketching, that allows for quantitative results. Going through first-order sketching (as was also done in [40]) proves the existence of a function  $f(r)$  bounding the sketch size, without giving it explicitly. We obtain explicit bounds in terms of the *weak coloring number* [57], written as  $\text{wcol}_r(\mathcal{F})$  for any  $r \in \mathbb{N}$  (Definition A.2). Using known bounds on the weak coloring number [66], we obtain the following corollary. As was the case for adjacency sketching, we observe that this proof (unlike the more general one for first-order sketching) produces sketches that only use randomization to compute a disjunction of EQUALITY checks, establishing that this extremely simple type of sketch suffices for monotone classes.

► **Corollary 1.3.** *Any graph class  $\mathcal{F}$  with bounded  $\text{wcol}_r(\mathcal{F})$  admits a small-distance sketch of size  $O(r + \text{wcol}_r(\mathcal{F}) \log(\text{wcol}_r(\mathcal{F})))$ . In particular, planar graphs admit a small-distance sketch of size  $O(r^3 \log r)$ , and the class of  $K_t$ -minor-free graphs admits a small-distance sketch of size  $O(r^{t-1} \log r)$ . Furthermore, planar graphs admit a small-distance labelling scheme of size  $O(r^3 \log n)$  and  $K_t$ -minor-free graphs admit a small-distance labelling scheme of size  $O(r^{t-1} \log n)$ .*

### 1.2.3 Approximate Distance Sketching

In light of Theorem 1.2, a reasonable question is whether ADT sketching for monotone classes is also determined by expansion. Our first result is that bounded expansion is necessary. All proofs on approximate distance sketching are omitted here due to space limitations, but can be found in the full version of the paper.

► **Theorem 1.4.** *If a monotone class  $\mathcal{F}$  is ADT sketchable, then it has bounded expansion.*

Combined with Theorem 1.2, this proves  $\text{ADT} \subseteq \text{SD}$ . Our proof uses a recent and fairly involved result in extremal graph theory [53], along with the theory of sparsity [57], to show that an  $\alpha$ -ADT sketch for a monotone class  $\mathcal{F}$  of unbounded expansion could be used to get a constant-size sketch for deciding  $\text{dist}(x, y) \leq 1$  vs.  $\text{dist}(x, y) > \alpha$  in arbitrary graphs, which (as we show) is a contradiction.

We are then concerned with the converse. We show that the class of max-degree 3 graphs, which has expansion exponential in  $r$  [56], is not ADT sketchable. After proving this theorem, we learned of an unpublished result [10] which proves a  $\Theta(\log(n)/\alpha)$  bound for one-way communication of the  $\alpha$ -ADT problem on degree-3 expander graphs. This could be used in place of our theorem to get the same qualitative (constant vs. non-constant) results, but not the quantitative bound: note that communication complexity cannot give sketching or labelling lower bounds better than  $\Theta(\log n)$ .

► **Theorem 1.5.** *For any  $\alpha > 1$ , any  $\alpha$ -ADT sketch for the class of graphs with maximum degree 3 has size at least  $\Omega(n^{\frac{1}{4\alpha}-\varepsilon})$ , for any constant  $\varepsilon > 0$ .*

This establishes that  $\text{ADT} \neq \text{SD}$  (and negatively answers open problem 2 of [2] about approximate distance labels for bounded-degree graphs, which [10] does not). But max-degree 3 graphs have exponential expansion. Smaller bounds on the expansion are associated with structural properties: for example, in monotone classes, polynomial expansion is equivalent to the existence of strongly sublinear separators [19]. One may then wonder if smaller bounds on the expansion suffice to guarantee ADT sketchability. We prove that this is not the case for two natural examples: subgraphs of the 3-dimensional grid (with polynomial expansion [57]), and subgraphs of the 2-dimensional grid with crosses (with linear expansion [18]) are not ADT sketchable. For this we require our Theorem 1.5.

► **Proposition 1.6.** *For the class of subgraphs of the 3-dimensional grid (the Cartesian product of 3 paths), and the class of subgraphs of the 2-dimensional grid (the strong product of 2 paths), an  $\alpha$ -ADT sketch requires size at least  $n^{\Omega(1/\alpha)}$ .*

We strengthen this result by showing that one can obtain monotone classes of graphs with expansion that grows arbitrarily slowly, which are not ADT sketchable.

► **Theorem 1.7.** *For any function  $\rho$  tending to infinity, there exists a monotone class of expansion  $r \mapsto \rho(r)$  that is not ADT sketchable. Moreover, for any  $\varepsilon > 0$ , there exists a monotone class  $\mathcal{F}$  of expansion  $r \mapsto O(r^\varepsilon)$ , such that, if  $\mathcal{F}$  admits an  $\alpha$ -ADT sketch of size  $s(n)$ , then we must have  $s(n) = n^{\Omega(1/\alpha)}$ .*

We conclude with a brief discussion of upper bounds for ADT sketching. A number of concepts have been introduced in the literature that can be used to obtain ADT sketches, including sparse covers [12] and padded decompositions [49].

Using the sketches obtained from sparse covers, combined with results of [23] on sparse covers (based on [49, 22]), we obtain the following, which complements our Theorem 1.7; note that the graph classes with constant expansion are exactly the proper minor-closed classes [57].

► **Corollary 1.8.** *For any  $t \geq 4$ , the class of  $K_t$ -minor-free graphs has a  $O(2^t)$ -ADT sketch of size  $O(t^2 \log t)$ . The sketch is equality-based and has one-sided error. As a consequence, every monotone class of constant expansion is ADT sketchable.*

It is also relatively straightforward to obtain ADT sketches from padded decompositions, with an interesting difference. These sketches may not have one-sided error and, unlike all other positive examples of sketches in this paper, they may not be equality-based. On the other hand, they are extremely small. We can use constructions of padded decompositions due to [52, 3] to obtain the following remarkable corollary:

► **Corollary 1.9.** *For any  $t \geq 4$ , the class of  $K_t$ -minor-free graphs has an  $O(t)$ -ADT sketch of size 2. For  $g \geq 0$ , the class of graphs embeddable on a surface of Euler genus  $g$  has an  $O(\log g)$ -ADT sketch of size 2.*

### 1.3 Discussion & Open Problems

The main problem left open by this paper is Question 3 for monotone classes of graphs; we have shown that a constant bound on the expansion implies ADT sketchability, while arbitrarily small non-constant bounds do not, but this does not rule out a monotone, ADT sketchable class with non-constant expansion.

We have examples showing that ADT sketching does not imply small-distance sketching, in general. But our examples are not even hereditary. Is there a hereditary class that is ADT sketchable, but not small-distance or adjacency sketchable?

Our Theorem 1.2 shows that bounded expansion implies first-order sketchability, and that for monotone classes the converse also holds. We showed more generally that classes of *structurally bounded expansion* are first-order sketchable. To extend our study of sketchability beyond monotone classes, it would be interesting to investigate whether the converse of this statement holds: does first-order sketchability of a *hereditary* class imply structurally bounded expansion?

In the preprint of this paper, we asked whether the class of subgraphs of hypercubes is a counterexample to the Implicit Graph Conjecture (IGC), which asks for deterministic adjacency labels of size  $O(\log n)$ . This conjecture was refuted recently in [41] by a non-constructive argument, and it would be interesting to find a more natural class that refutes the conjecture. The *induced* subgraphs of hypercubes are adjacency sketchable and therefore admit adjacency labels of size  $O(\log n)$  (see e.g. [40]), but our Corollary 3.6 shows that the subgraphs are not adjacency sketchable. Prior work (e.g. [16]) has not succeeded in finding labeling schemes of size  $O(\log n)$  for this class. These observations made it plausible to us that efficient labeling schemes for this class do not exist. However, efficient adjacency labels for this class have since been found in [21]. A related question is whether we may characterize the monotone classes of graphs which admit adjacency labeling schemes of size  $O(\log n)$ .

We have focused on determining whether there *exists* a constant  $\alpha$  such that a class is  $\alpha$ -ADT sketchable. It is also of interest to obtain sketches for arbitrarily small  $\alpha > 1$ , with sketch size depending on  $\alpha$ . One strategy is to embed the graph isometrically into  $\ell_1$ , but this is not always the best option. We obtained a  $(1 + \varepsilon)$ -ADT sketch for the class of forests with size  $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ , but this result appeared earlier in [10]; this sketch is more efficient than the one obtained by embedding the trees isometrically in  $\ell_1$ . We remark that a class (monotone or not) that admits a  $(1 + \varepsilon)$ -ADT sketch for  $\varepsilon < 1$  must also admit an adjacency sketch.

Finally, we point out an interesting conjecture of [37], that all constant-cost public-coin communication problems contain a large monochromatic rectangle. In our terminology, using the equivalence between constant-cost communication and adjacency sketching from [40], this conjecture states that all adjacency sketchable graph classes have the strong Erdős-Hajnal property.



## 2 Preliminaries

### 2.1 Notation

Throughout the paper,  $\log$  denotes the logarithm base 2, while  $\ln$  denotes the natural logarithm.

We will write  $\mathbf{1}[E]$  for the indicator variable for the event  $E$ , which takes value 1 if  $E$  is true. Given a graph  $G$ , the length of a path  $P$  in  $G$  is the number of edges of  $P$ . Given two vertices  $x, y \in V(G)$ , we define  $\text{dist}_G(x, y)$  to be the infimum of the length of a path between  $x$  and  $y$  in  $G$ ; we define  $\text{dist}_G(x, y) = \infty$  if there exists no path between  $x$  and  $y$ . Notice that  $(V(G), \text{dist}_G)$  is a metric space (with possibly infinite distances between pairs of vertices if  $G$  is disconnected).

The *girth* of a graph  $G$  is defined as the size of a shortest cycle in  $G$  (if  $G$  is acyclic, its girth is infinite).

### 2.2 Distance and First-Order Sketching

We will require more general notions of sketching than those introduced above. For a class  $\mathcal{F}$  of graphs, we will say that a sequence  $\{f_G\}_{G \in \mathcal{F}}$  of partial functions  $f_G : V(G) \times V(G) \rightarrow \{0, 1, *\}$  is a *partial function  $f$  parameterized by graphs  $G \in \mathcal{F}$* . We will write  $f$  to refer to this sequence.

For a graph class  $\mathcal{F}$ , we define an  *$f$ -sketch* for  $\mathcal{F}$  as a decoder  $D : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$ , such that for every  $G \in \mathcal{F}$  the following holds. There is a probability distribution over functions  $\text{sk} : V(G) \rightarrow \{0, 1\}^*$ , such that for all  $x, y \in V(G)$ ,

$$f_G(x, y) \neq * \implies \Pr[D(\text{sk}(x), \text{sk}(y)) = f_G(x, y)] \geq 2/3.$$

We define the *size* of the sketch as

$$\max_{G \in \mathcal{F}_n} \sup_{\text{sk}} \max_{x \in V(G)} |\text{sk}(x)|,$$

where the supremum is over the set of functions  $\text{sk} : V(G) \rightarrow \{0, 1\}^*$  in the support of the distribution defined for  $G$ , and  $|\text{sk}(x)|$  is the number of bits of  $\text{sk}(x)$ . We will say that a class  $\mathcal{F}$  is  *$f$ -sketchable* if there exists an  $f$ -sketch for  $\mathcal{F}$  with size that does not depend on the number of vertices  $n$ .

For a graph class  $\mathcal{F}$ , we also define an  *$f$ -labelling scheme* for  $\mathcal{F}$  similar to above, except that for every  $G \in \mathcal{F}$  there is a *deterministic* function  $\ell : V(G) \rightarrow \{0, 1\}^*$  such that for all  $x, y \in V(G)$ ,

$$f_G(x, y) \neq * \implies D(\ell(x), \ell(y)) = f_G(x, y).$$

The following simple proposition (observed in [38, 40]) relates sketches to labelling schemes:

► **Proposition 2.1.** *If  $\mathcal{F}$  admits an  $f$ -sketch of size  $s(n)$ , then it admits an  $f$ -labelling scheme of size  $O(s(n) \log n)$ .*

We now define certain important types of  $f$ -sketches. Let  $\mathcal{F}$  be a class of graphs. For any  $r_1 \leq r_2$ , a *distance- $(r_1, r_2)$  sketch* for  $\mathcal{F}$  is an  $f$ -sketch, as defined above, when for any graph  $G$  we define the function

$$f_G(x, y) = \begin{cases} 1 & \text{if } \text{dist}_G(x, y) \leq r_1 \\ 0 & \text{if } \text{dist}_G(x, y) > r_2 \\ * & \text{otherwise.} \end{cases}$$

The size of such a sketch may depend on  $r_1$ ,  $r_2$ , the number of vertices  $n$ , or other graph parameters.

Recall the definitions of *adjacency sketchable*, *small-distance sketchable*, and *ADT sketchable*. It is clear that:

1. A class  $\mathcal{F}$  is *adjacency sketchable* if it is distance- $(1, 1)$  sketchable;
2. A class  $\mathcal{F}$  is *small-distance sketchable* if for every  $r \geq 1$  it is distance- $(r, r)$  sketchable.
3. A class  $\mathcal{F}$  is  $\alpha$ -*ADT sketchable* if for every  $r \geq 1$  it is distance- $(r, \alpha r)$  sketchable, and furthermore the size of the sketch does not depend on  $r$ .

Following [40], we will also define *FO-sketchable* classes, for which we require some terminology (see e.g. [59] for more on the following terminology). A *relational vocabulary*  $\Sigma$  is a set of relation symbols, with each  $R \in \Sigma$  having an *arity*  $\text{arity}(R) \in \mathbb{N} \setminus \{0\}$ . A  $\Sigma$ -structure  $\mathcal{A}$  consists of a *domain*  $A$ , and for each relation symbol  $R \in \Sigma$  an *interpretation*  $R^{\mathcal{A}} \subseteq A^{\text{arity}(R)}$ , which is a relation. Fix a countably infinite set  $X$  of *variables*. *Atomic formulas of vocabulary*  $\Sigma$  are of the form

- $x = y$  for  $x, y \in X$ ; or,
- $R(x_1, \dots, x_r)$  for  $x_1, \dots, x_r \in X$ ,  $R \in \Sigma$  and  $r = \text{arity}(R)$ , which evaluates to true when  $(x_1, \dots, x_r) \in R$ .

*First-order (FO) formulas of vocabulary*  $\Sigma$  are inductively defined as either atomic formulas, or a formula of the form  $\neg\phi$ ,  $\phi \wedge \psi$ ,  $\phi \vee \psi$ , or  $\exists x.\phi$  or  $\forall x.\psi$ , where  $\phi$  and  $\psi$  are each FO formulas. A *free variable* of a formula  $\phi$  is one which is not bound by a quantifier. We will write  $\phi(x_1, x_2, \dots, x_k)$  to show that the free variables of  $\phi$  are  $x_1, \dots, x_k \in X$ . For a value  $u \in A$ , we write  $\phi[u/x]$  for the formula obtained by substituting the constant  $u$  for the free variable  $x$ .

Let  $\phi(x, y)$  be any formula with two free variables and relational vocabulary  $\Sigma = \{E', R_1, \dots, R_k\}$  where  $E'$  is symmetric of arity 2 and each  $R_i$  is unary (i.e. of arity 1). We will say that a graph class  $\mathcal{F}$  is  $\phi$ -*sketchable* if it is  $f$ -sketchable for any  $f$  chosen as follows. For any graph  $G = (V, E)$ , we choose any  $\Sigma$ -structure with domain  $V$  where  $E$  is the interpretation of the symbol  $E'$ . Then set  $f_G(u, v) = 1$  if and only if  $\phi(u/x, v/y)$  evaluates to true.

We remark that for any graph  $G$ , there are many ways to choose a  $\Sigma$ -structure with domain  $V$  with  $E$  being the interpretation of  $E'$ . To be first-order sketchable, a class  $\mathcal{F}$  must be  $f$ -sketchable for *every* such choice of functions  $f_G$ . A concrete example is that, for any  $r \in \mathbb{N}$ , we can choose the formula

$$\phi(x, y) = \exists u_1, u_2, \dots, u_{r-1} : (E'(x, u_1) \vee x = u_1) \wedge (E'(u_1, u_2) \vee u_1 = u_2) \wedge \dots \wedge (E'(u_{r-1}, y) \vee u_{r-1} = y),$$

which evaluates to true if and only if  $\text{dist}_G(x, y) \leq r$ .

## 2.3 Bounded expansion

Here we introduce the notion of expansion from sparsity theory, as discussed in [57].

► **Definition 2.2** (Bounded Expansion). *Given a graph  $G$  and an integer  $r \geq 0$ , a depth- $r$  minor of  $G$  is a graph obtained by contracting pairwise disjoint connected subgraphs of radius at most  $r$  in a subgraph of  $G$ . For any function  $f$ , we say that a class of graphs  $\mathcal{G}$  has expansion at most  $f$  if any depth- $r$  minor of a graph of  $\mathcal{G}$  has average degree at most  $f(r)$  (see [57] for more details on this notion). We say that a class  $\mathcal{G}$  has bounded expansion if there is a function  $f$  such that  $\mathcal{G}$  has expansion at most  $f$ .*

Note that, for example, every proper minor-closed family has constant expansion.



## 2.4 Equality-Based Labelling Schemes and Sketches

An equality-based labelling scheme is one which assigns to each vertex a deterministic label, comprising a data structure of size  $s$  that holds  $k$  “equality codes”; which can be used only for checking equality. These labelling schemes: 1) capture the constant-cost randomized communication protocols that can be simulated by a constant-cost *deterministic* communication protocol with access to an EQUALITY oracle (as studied in e.g. [15, 13, 37, 40]); and 2) capture a common type of adjacency labels, including those of [47] for bounded arboricity graphs (see [40] for others).

One might formalize these schemes in a few ways; we slightly adapt the definition from [40]. This definition is intended to simplify notation rather than optimize label size, since we care mainly about constant vs. non-constant.

► **Definition 2.3** (Equality-Based Labeling Scheme). *Let  $\mathcal{F}$  be a class of graphs and let  $f : \mathbb{N} \times \mathbb{N} \times \mathcal{F} \rightarrow \{0, 1, *\}$  be a partial function. An  $(s, t, k)$ -equality-based  $f$ -labelling scheme for  $\mathcal{F}$  is an algorithm  $D$ , called a decoder, which satisfies the following. For every  $G \in \mathcal{F}$  with vertex set  $[n]$  and every  $x \in [n]$ , there is a sequence of the form*

$$\ell_G(x) = [(p_1(x) \mid \vec{q}_1(x)), (p_2(x) \mid \vec{q}_2(x)), \dots, (p_t(x) \mid \vec{q}_t(x))],$$

where the vectors  $p_i(x) \in \{0, 1\}^*$  are called the prefixes, and the entries of the vectors  $\vec{q}_i(x) \in \mathbb{N}^*$  are called equality codes (which we may assume are positive integers). We must have  $\sum_{i=1}^t |p_i(x)| \leq s$  and  $\sum_{i=1}^t |\vec{q}_i(x)| \leq k$  (recall that given a vector  $v$  of binary numbers or integers,  $|v|$  denotes the number of entries of  $v$ ). We insist on the fact that  $k$  bounds the total number of equality codes associated with any vertex  $x$ , but not necessarily the total number of bits needed to store these codes (see Example 2.4 below, where  $k = 2$  but storing the codes would require  $2 \log n$  bits per vertex). On inputs  $\ell_G(x), \ell_G(y)$ , the algorithm  $D$  chooses a function  $D_{p(x), p(y)}$ , where  $p(x) = (p_1(x), \dots, p_t(x))$ , and outputs

$$D_{p(x), p(y)}(Q_{x,y}),$$

where

$$Q_{x,y}(i_1, i_2, j_1, j_2) = \mathbf{1}[(\vec{q}_{i_1}(x))_{j_1} = (\vec{q}_{i_2}(x))_{j_2}] \quad (1)$$

is the set of equality values for every pair of equality codes. It is required that, for every  $G \in \mathcal{F}$  and  $x, y \in V(G)$ ,

$$f(x, y, G) \neq * \implies D_{p(x), p(y)}(Q_{x,y}) = f(x, y, G).$$

We make the further distinction of calling a labelling scheme  $(s, k)$ -disjunctive if it is an  $(s, t, k)$ -equality-based labelling scheme, where each function  $D_{p(x), p(y)}$  is simply a disjunction over a subset of values  $Q_{x,y}(i_1, i_2, j_1, j_2)$ .

When an element  $(p_i(x) \mid \vec{q}_i(x))$  in an equality-based label has  $p_i(x)$  of size 0, we will write  $(- \mid \vec{q}_i(x))$ ; similarly, we write  $(p_1(x) \mid -)$  when  $\vec{q}_i(x)$  is empty.

► **Example 2.4.** The adjacency labelling scheme of [47] for forests can be written as an equality-based labelling scheme. For each  $x$  in an  $n$ -vertex forest  $G$  with arbitrarily rooted trees, which we assume has vertex set  $[n]$ , we assign the label  $\ell_G(x) = [(- \mid (x, p(x)))]$  where  $p(x)$  is the parent of  $x$  if it has one, or 0 otherwise. Here  $\vec{q}_1(x) = (x, p(x)) \in \mathbb{N}^2$ . The decoder simply outputs the disjunction of  $p(x) = y$  or  $p(y) = x$ , so in fact this is a  $(0, 1, 2)$ -disjunctive labeling scheme.

An equality-based labelling scheme is easily transformed into a standard deterministic labelling scheme or a sketch. The following simple proposition was observed in [40]. We sketch the proof for the sake of clarity.

► **Proposition 2.5.** *Let  $\mathcal{F}$  be a class of graphs and  $f : \mathbb{N} \times \mathbb{N} \times \mathcal{F} \rightarrow \{0, 1, *\}$  be a partial function. If there is an  $(s, t, k)$ -equality-based  $f$ -labelling scheme for  $\mathcal{F}$  then there is an  $f$ -sketch for  $\mathcal{F}$  of size at most  $O(s + t + k \log k)$ . If the scheme is disjunctive, the sketch has one-sided error: when  $f(x, y, G) = 1$ , the sketch will produce the wrong output with probability 0.*

**Proof sketch.** Choose a random function  $\xi : \mathbb{N} \rightarrow [w]$  for  $w = 3k^2$ . For any vertex  $x$  of a graph  $G$ , replace each vector  $\vec{q}_i(x) = (q_{i,1}(x), \dots, q_{i,m}(x))$  with  $(\xi(q_{i,1}(x)), \dots, \xi(q_{i,m}(x)))$ . We have replaced each of the (at most)  $k$  equality codes  $(\vec{q}_i(x))_j$  with  $\xi((q_i(x))_j)$ , using  $k \log w = O(k \log k)$  bits in total. The sketch has size  $O(s + t + k \log k)$  since we must include each  $p_i(x)$  (using  $s$  bits in total), the  $O(k \log k)$  bits for the equality codes, and  $O(t)$  bits to encode the symbols  $(\cdot)$ .

For two vertices  $x, y$ , write  $Q_{x,y}^\xi(i_1, i_2, j_1, j_2) = \mathbf{1}[\xi((\vec{q}_{i_1}(x))_{j_2}) = \xi((\vec{q}_{j_1}(y))_{j_2})]$ . Since there are at most  $k$  equality codes in each label, there are at most  $k^2$  equality comparisons. By the union bound, the probability that any of these comparisons have

$$\mathbf{1}[\xi((\vec{q}_{i_1}(x))_{j_2}) = \xi((\vec{q}_{j_1}(y))_{j_2})] \neq \mathbf{1}[(\vec{q}_{i_1}(x))_{j_2} = (\vec{q}_{j_1}(y))_{j_2}]$$

is at most  $k^2 \cdot (1/w) = 1/3$ , so with probability at least  $2/3$  all of the comparisons made by the decoder have the correct value, so the decoder will be correct. Note that when  $(\vec{q}_{i_1}(x))_{j_2} = (\vec{q}_{j_1}(y))_{j_2}$ , the random values under  $\xi$  will be equal with certainty. We conclude from this that disjunctive schemes will produce sketches with one-sided error. ◀

► **Remark 2.6.** Disjunctive labelling schemes with  $s = 0$  (i.e. the  $p$  values are empty) can be transformed into *locality-sensitive hashes (LSH)* [45]. A  $(r_1, r_2, \gamma_1, \gamma_2)$ -LSH must map any two points  $x, y$  with  $\text{dist}(x, y) \leq r_1$  to the same hash value with probability at least  $\gamma_1$ , and map any two points  $x, y$  with  $\text{dist}(x, y) > r_2$  to the same hash value with probability at most  $\gamma_2$ , where  $r_1 < r_2$  and  $\gamma_1 > \gamma_2$ . By boosting the success probability of each EQUALITY check in the disjunction, and then sampling a uniformly random term from the disjunction, one obtains an LSH with distance parameters that depend on the original sketch. All of the equality-based sketches presented in this paper, except the first-order sketches, are of this form.

### 3 Adjacency Sketching

In this section, we prove Theorem 1.1, and include the additional equivalent statement that  $\mathcal{F}$  admits a constant-size *disjunctive* adjacency sketch. We think of disjunctive sketches as the simplest possible use of randomization in a sketch, with the theorem establishing that the simplest possible sketches are sufficient for monotone classes.

► **Theorem 3.1.** *Let  $\mathcal{F}$  be a monotone class of graphs. Then the following are equivalent:*

1.  $\mathcal{F}$  is adjacency sketchable.
2.  $\mathcal{F}$  admits a constant-size disjunctive adjacency labelling scheme.
3.  $\mathcal{F}$  has bounded arboricity.

A disjunctive labelling scheme for graphs of arboricity  $k$  can be obtained from the adjacency labelling scheme of [47], as in Example 2.4. This leads to a sketch of size  $O(k \log k)$  by Proposition 2.5, which was improved slightly in [40]:

► **Proposition 3.2** ([40]). *Let  $\mathcal{F}$  be any class with arboricity at most  $k$ . Then  $\mathcal{F}$  admits a  $(0, 1, k + 1)$ -disjunctive adjacency labelling scheme, and an adjacency sketch of size  $O(k)$ .*

Therefore, to prove Theorem 3.1, it suffices to prove  $(1) \implies (3)$ , which we will prove by contrapositive. This proof will use the notion of discrepancy from communication complexity. Our proof is inspired by the recent proof of Hambardzumyan, Hatami, & Hatami [37], which refuted the main conjecture of [40]. Our proof also leads to another, more natural counterexample to the conjecture of [40]: the class of subgraphs of the hypercube.

Consider a graph  $G = (V, E)$ , let  $f : V \times V \rightarrow \{0, 1, *\}$  be a partial function, and let  $\mu$  be a probability distribution over  $V \times V$  that is supported on pairs  $(x, y)$  which satisfy  $f(x, y) \neq *$ . Let  $X, Y \subseteq V$ . Then we define the *discrepancy* of  $R = X \times Y$  as

$$\text{Disc}_{\mu, f}(G, R) = \left| \Pr_{\mu}[(x, y) \in R \cap f^{-1}(1)] - \Pr_{\mu}[(x, y) \in R \cap f^{-1}(0)] \right|,$$

where  $(x, y)$  is drawn from  $\mu$ . The discrepancy of  $G$  under  $\mu$  is defined as

$$\text{Disc}_{\mu, f}(G) = \max_R \text{Disc}_{\mu, f}(G, R),$$

where the maximum is over all sets  $R = X \times Y$  with  $X, Y \subseteq V$ . The following lemma is essentially a restatement of a standard lower-bound technique in communication complexity.

► **Lemma 3.3.** *Let  $G = (V, E)$  be any graph on  $n$  vertices, let  $\mathcal{F}$  be any class of graphs containing  $G$ , and let  $f$  be a partial function parameterized by graphs in  $\mathcal{F}$ . Let  $\mu$  be any probability distribution over  $V \times V$  supported on a subset of  $\{(x, y) : f_G(x, y) \neq *\}$ . Then any  $f$ -sketch for  $\mathcal{F}_n$  has size at least  $\frac{1}{2} \log \frac{1}{3 \text{Disc}_{\mu, f}(G)}$ .*

A *spanning subgraph* of a graph  $G = (V, E)$  is a subgraph of  $G$  with vertex set  $V$ . Our next lemma will give a lower bound on the adjacency sketch size for the class  $\mathcal{G}$  of spanning subgraphs of a graph  $G$  of minimum degree  $d$ . We will actually prove the lower bound for a weaker type of adjacency sketch, which is only required to be correct on pairs  $(x, y)$  that were originally edges in  $G$ . This stronger statement is not necessary for the current section, but will be used in the proof of Theorem A.16.

For a graph  $G = (V, E)$  and the class  $\mathcal{G}$  of spanning subgraphs of  $G$ , and any subgraph  $H \in \mathcal{G}$ , we will define the partial function  $\text{adj}_H^E : V \times V \rightarrow \{0, 1, *\}$  as

$$\text{adj}_H^E(x, y) = \begin{cases} \text{adj}_H(x, y) & \text{if } (x, y) \in E \\ * & \text{otherwise.} \end{cases}$$

In the remainder of this section, we view  $\text{adj}^E$  as the function  $(\text{adj}_H^E)_{H \in \mathcal{G}}$  parametrized by  $H \in \mathcal{G}$ . In particular, an  $\text{adj}^E$ -sketch for  $\mathcal{G}$  computes the partial function  $\text{adj}_H^E$  for each  $H \in \mathcal{G}$ .

We show by the probabilistic method that there is a distribution  $\mu$  and a subgraph of  $G$  with discrepancy  $O(1/\sqrt{d})$  with respect to  $\mu$ . We will require the standard Chernoff bound for the binomial distribution with parameters  $n$  and  $\frac{1}{2}$  (see Corollary A.1.2 in [5]): for any  $t > 0$ ,

$$\Pr(|\text{Bin}(n, \frac{1}{2}) - \frac{n}{2}| > t) < 2 \exp(-2t^2/n).$$

► **Lemma 3.4.** *Let  $G = (V, E)$  be a graph of minimum degree  $d$ , and let  $\mathcal{G}$  be the class of spanning subgraphs of  $G$ . Then any  $\text{adj}^E$ -sketch for  $\mathcal{G}$  requires size at least  $\Omega(\log d)$ .*

**Proof.** Let  $\mathbf{H}$  be a random spanning subgraph of  $G$  obtained by including each edge of  $G$  independently with probability  $1/2$ . Note that  $\mathbf{H} \in \mathcal{G}$  with probability 1. Let  $m = |E|$  and let  $\mu$  be the probability distribution over  $V \times V$  such that for every  $(x, y) \in V \times V$ , we have  $\mu((x, y)) = 1/m$  if  $(x, y) \in E$ , and  $\mu((x, y)) = 0$  otherwise (so that  $\mu$  is uniform over the edges of  $G$ ). For simplicity, write  $\text{Disc}_\mu$  for  $\text{Disc}_{\mu, f}$  where  $f = \text{adj}^E$ . We will prove that  $\text{Disc}_\mu(\mathbf{H})$  is small, with nonzero probability over  $\mathbf{H}$ .

Consider a set  $R = X \times Y$  with  $X, Y \subseteq V$ , and let  $k \leq m$  be the number of edges  $(x, y)$  of  $G$  with  $(x, y) \in R$ . Let  $H$  be any subgraph of  $G$  with  $|E(H) \cap R| = \ell \leq k$ . Then

$$\begin{aligned} \text{Disc}_\mu(H, R) &= \left| \Pr_\mu[(x, y) \in E(H) \cap R] - \Pr_\mu[(x, y) \in R \setminus E(H)] \right| \\ &= \left| \frac{\ell}{m} - \frac{k - \ell}{m} \right| = \frac{|2\ell - k|}{m}. \end{aligned}$$

For fixed  $R = X \times Y$ , it then holds that  $\text{Disc}_\mu(\mathbf{H}, R)$  is a random variable  $\frac{|2\ell - k|}{m}$ , where  $\ell \sim \text{Bin}(k, 1/2)$ . Then, by the Chernoff bound, we have for any  $\varepsilon > 0$  that

$$\Pr[\text{Disc}_\mu(\mathbf{H}, R) > \varepsilon] = \Pr\left[\left|\text{Bin}\left(k, \frac{1}{2}\right) - \frac{k}{2}\right| > \frac{1}{2}\varepsilon m\right] \leq 2 \exp\left(-\frac{\varepsilon^2 m^2}{2k}\right) \leq 2 \exp(-\varepsilon^2 m/2),$$

where the last inequality is due to  $k \leq m$ . There are at most  $2^{2n}$  sets  $R = X \times Y \subseteq V \times V$ , so by the union bound,

$$\begin{aligned} \Pr[\exists R = X \times Y \subseteq V \times V : \text{Disc}_\mu(\mathbf{H}, R) > \varepsilon] &\leq 2^{2n+1} \exp(-\varepsilon^2 m/2) \\ &= \exp((2n+1)\ln(2) - \varepsilon^2 m/2). \end{aligned}$$

Now, since  $G$  has minimum degree  $d$ , we have  $m \geq dn/2$ . Setting  $\varepsilon = \Omega\left(\frac{1}{\sqrt{d}}\right)$  with a sufficiently large implicit multiplicative constant, we get an upper bound on this probability of

$$\exp((2n+1)\ln(2) - \varepsilon^2 m/2) \leq \exp((2n+1)\ln(2) - \varepsilon^2 dn/4) < 1.$$

Therefore there exists a subgraph  $H$  with  $\text{Disc}_\mu(H) = O(1/\sqrt{d})$ . Applying Lemma 3.3, we see that any  $\text{adj}^E$ -sketch for  $\mathcal{G}$  must have size at least  $\Omega(\log(\sqrt{d})) = \Omega(\log d)$ . ◀

We may now complete the proof of Theorem 3.1. We aim to prove (1)  $\implies$  (3), which we will prove by contrapositive: i.e. that any class of unbounded arboricity has non-constant adjacency sketch size.

► **Lemma 3.5.** *Let  $\mathcal{F}$  be any monotone class of graphs with unbounded arboricity. Then  $\mathcal{F}$  does not admit a constant-size adjacency sketch.*

**Proof.** It is well-known that the degeneracy of a graph is within factor 2 of the arboricity, so the degeneracy of  $\mathcal{F}$  must also be unbounded. Then for any integer  $d \in \mathbb{N}$ , there is a graph  $G \in \mathcal{F}$  with degeneracy at least  $d$ . By definition,  $G$  contains a subgraph  $H$  of minimum degree at least  $d$ . Let  $\mathcal{G}$  be the class of spanning subgraphs of  $G$ . Since  $\mathcal{F}$  is monotone, we have  $\mathcal{G} \subseteq \mathcal{F}$ . Then by Lemma 3.4, any adjacency sketch for  $\mathcal{G}$  must have size at least  $\Omega(\log d)$ . Then for any integer  $d$ , we obtain a lower bound of  $\Omega(\log d)$  on the size of an adjacency sketch for  $\mathcal{F}$ ; it follows that any adjacency sketch for  $\mathcal{F}$  is of non-constant size. ◀

As a consequence, we obtain the following counterexample to the main conjecture of [40]. We remind the reader that the conjecture was already refuted in [36], using an interesting construction of a graph class that was originally used to establish a “proof barrier” in communication complexity [37]. Our counterexample, the subgraphs of the hypercube, is more easily defined. The following bound on the number of subgraphs of the hypercube was observed by Viktor Zamaraev (personal communication). See [40] for a definition of *stable*.

► **Corollary 3.6.** *Let  $\mathcal{F}$  be a class of subgraphs of the hypercube. Then:*

1.  $\mathcal{F}$  is stable, and there are at most  $2^{O(n \log n)}$  graphs on  $n$  vertices in  $\mathcal{F}$ .
2.  $\mathcal{F}$  is not adjacency sketchable.

**Proof.** Since the  $d$ -dimensional hypercube of size  $N = 2^d$  has minimum degree  $d = \log N$ ,  $\mathcal{F}$  has non-constant adjacency sketch size. To bound the number of  $n$ -vertex subgraphs of the hypercubes, we first observe that there are at most  $2^{O(n \log n)}$  induced subgraphs of the hypercube on  $n$  vertices, which follows from the  $O(\log n)$  adjacency labelling scheme for this class [38] (see a simpler exposition at [39]). It is known that any  $n$ -vertex induced subgraph of the hypercube has at most  $O(n \log n)$  edges [33], so each induced subgraph admits at most  $2^{O(n \log n)}$  spanning subgraphs. Therefore the number of  $n$ -vertex subgraphs of the hypercube is at most  $2^{O(n \log n)} \cdot 2^{O(n \log n)} = 2^{O(n \log n)}$ . Any monotone class of graphs which is not stable contains  $K_{t,t}$ , for every  $t \in \mathbb{N}$ , and therefore contains the class of all bipartite graphs. This does not hold for  $\mathcal{F}$  (or indeed for any class of factorial speed), so  $\mathcal{F}$  must be stable. ◀

---

## References

---

- 1 Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the forty-fourth annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 1199–1218, 2012.
- 2 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the twenty-fifth annual ACM Symposium on Principles of Distributed Computing (PODC 2006)*, pages 188–197, 2006.
- 3 Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. *SIAM Journal on Computing*, 48(3):1120–1145, 2019.
- 4 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.
- 5 Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.
- 6 Stephen Alstrup, Philip Bille, and Theis Rauhe. Labeling schemes for small distances in trees. *SIAM Journal on Discrete Mathematics*, 19(2):448–462, 2005.
- 7 Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear distance labeling. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.
- 8 Stephen Alstrup, Cyril Gavoille, Esben Bistrup Halvorsen, and Holger Petersen. Simpler, faster and shorter labels for distances in graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 338–350. SIAM, 2016.
- 9 Stephen Alstrup, Inge Li Gørtz, Esben Bistrup Halvorsen, and Ely Porat. Distance labeling schemes for trees. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.
- 10 Alexandr Andoni and Robert Krauthgamer. Distance estimation protocols for general metrics. <https://www.cs.columbia.edu/~andoni/papers/de.pdf>, 2008.
- 11 Alexandr Andoni, Robert Krauthgamer, and Ilya P. Razenshteyn. Sketching and embedding are equivalent for norms. *SIAM J. Comput.*, 47(3):890–916, 2018. doi:10.1137/15M1017958.
- 12 Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 503–513. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89571.
- 13 Alexander R. Block, Simina Branzei, Hemanta K. Maji, Himanshi Mehta, Tamalika Mukherjee, and Hai H. Nguyen.  $P_4$ -free partition and cover numbers and application. *Cryptology ePrint Archive*, Report 2020/1605, 2020. URL: <https://ia.cr/2020/1605>.

- 14 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1977–1996. SIAM, 2021.
- 15 Arkadev Chattopadhyay, Shachar Lovett, and Marc Vinyals. Equality alone does not simulate randomness. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.
- 16 Victor Chepoi, Arnaud Labourel, and Sébastien Ratel. On density of subgraphs of Cartesian products. *Journal of Graph Theory*, 93(1):64–87, 2020.
- 17 Erik D Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *arXiv preprint*, 2014. [arXiv:1406.2587](#).
- 18 Zdeněk Dvořák. A note on sublinear separators and expansion. *European J. Combin.*, 93:103273, 2021. [doi:10.1016/j.ejc.2020.103273](#).
- 19 Zdeněk Dvořák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM J. Discret. Math.*, 30(2):1095–1101, 2016. [doi:10.1137/15M1017569](#).
- 20 Talya Eden, Piotr Indyk, and Haike Xu. Embeddings and labeling schemes for  $A^*$ . In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215, page 62. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
- 21 Louis Esperet, Nathaniel Harms, and Viktor Zamaraev. Optimal adjacency labels for subgraphs of cartesian products. *arXiv preprint*, 2022. [arXiv:2206.02872](#).
- 22 Jittat Fakcharoenphol and Kunal Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 36–46. Springer, 2003.
- 23 Arnold Filtser. Scattering and sparse partitions, and their applications. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 24 Pierre Fraigniaud and Amos Korman. On randomized representations of graphs using short labels. In *Proceedings of the twenty-first annual Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pages 131–137, 2009.
- 25 Ofer Freedman, Paweł Gawrychowski, Patrick K Nicholson, and Oren Weimann. Optimal distance labeling schemes for trees. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2017)*, pages 185–194, 2017.
- 26 Jakub Gajarský, Stephan Kreutzer, Jaroslav Nešetřil, Patrice Ossona de Mendez, Michał Pilipczuk, Sebastian Siebertz, and Szymon Toruńczyk. First-order interpretations of bounded expansion classes. *ACM Trans. Comput. Logic*, 21(4), July 2020. [doi:10.1145/3382093](#).
- 27 Jakub Gajarský, Michał Pilipczuk, and Szymon Toruńczyk. Stable graphs of bounded twin-width. *arXiv preprint*, 2021. [arXiv:2107.03711](#).
- 28 Cyril Gavoille, Michał Katz, Nir A Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. In *European Symposium on Algorithms (ESA 2001)*, pages 476–487. Springer, 2001.
- 29 Cyril Gavoille and Arnaud Labourel. On local representation of distances in trees. In *Proceedings of the twenty-sixth annual ACM Symposium on Principles of Distributed Computing (PODC 2007)*, pages 352–353, 2007.
- 30 Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2):111–120, 2003.
- 31 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.
- 32 Paweł Gawrychowski and Przemysław Uznański. Better distance labeling for unweighted planar graphs. In *Workshop on Algorithms and Data Structures (WADS 2021)*, pages 428–441. Springer, 2021.
- 33 Ron L. Graham. On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170–186, 1970.



- 34 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM (JACM)*, 64(3):1–32, 2017.
- 35 Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Cuts, trees and  $\ell_1$ -embeddings of graphs. *Combinatorica*, 24(2):233–269, 2004.
- 36 Lianna Hambardzumyan, Hamed Hatami, and Pooya Hatami. A counter-example to the probabilistic universal graph conjecture via randomized communication complexity, 2021. [arXiv:2111.10436](https://arxiv.org/abs/2111.10436).
- 37 Lianna Hambardzumyan, Hamed Hatami, and Pooya Hatami. Dimension-free bounds and structural results in communication complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, page 66, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/066>.
- 38 Nathaniel Harms. Universal communication, universal graphs, and graph labeling. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, page 33. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
- 39 Nathaniel Harms. Adjacency labeling and sketching for induced subgraphs of the hypercube. [https://cs.uwaterloo.ca/~nharms/downloads/hypercube\\_sketch.pdf](https://cs.uwaterloo.ca/~nharms/downloads/hypercube_sketch.pdf), 2022. URL: [https://cs.uwaterloo.ca/~nharms/downloads/hypercube\\_sketch.pdf](https://cs.uwaterloo.ca/~nharms/downloads/hypercube_sketch.pdf).
- 40 Nathaniel Harms, Sebastian Wild, and Viktor Zamaraev. Randomized communication and implicit graph representations. In *Proceedings of the 54th annual ACM Symposium on Theory of Computing (STOC)*, 2022.
- 41 Hamed Hatami and Pooya Hatami. The implicit graph conjecture is false. *arXiv preprint arXiv:2111.13198*, 2021.
- 42 Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the hamming distance problem. *Information Processing Letters*, 99(4):149–153, 2006.
- 43 Tony Huynh, Bojan Mohar, Robert Šámal, Carsten Thomassen, and David R Wood. Universality in minor-closed graph classes. *arXiv preprint*, 2021. [arXiv:2109.00327](https://arxiv.org/abs/2109.00327).
- 44 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.
- 45 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 604–613, 1998.
- 46 T.S. Jayram. Problem 25: Communication complexity and metric spaces. <https://sublinear.info/25>, 2009. URL: <https://sublinear.info/25>.
- 47 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.
- 48 Subhash Khot and Assaf Naor. The Andoni–Krauthgamer–Razenshteyn characterization of sketchable norms fails for sketchable metrics. In *Proceedings of the thirtieth annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 1814–1824. SIAM, 2019.
- 49 Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM Symposium on Theory of Computing (STOC 1993)*, pages 682–690, 1993.
- 50 Daniela Kühn and Deryk Osthus. Every graph of sufficiently large average degree contains a  $C_4$ -free subgraph of large average degree. *Combinatorica*, 24(1):155–162, 2004.
- 51 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.
- 52 James R Lee and Anastasios Sidiropoulos. Genus and the geometry of the cut graph. In *Proceedings of the twenty-first annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 193–201. SIAM, 2010.
- 53 Hong Liu and Richard Montgomery. A solution to Erdős and Hajnal’s odd cycle problem. *arXiv preprint*, 2020. [arXiv:2010.15802](https://arxiv.org/abs/2010.15802).
- 54 Jiří Matoušek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.
- 55 John H Muller. Local structure in graph classes, 1989.

- 56 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, 2008.
- 57 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer-Verlag, 2012.
- 58 Jaroslav Nešetřil and Patrice Ossona de Mendez. On low tree-depth decompositions. *Graphs and combinatorics*, 31(6):1941–1963, 2015.
- 59 Jaroslav Nešetřil, Patrice Ossona de Mendez, and Sebastian Siebertz. Structural properties of the first-order transduction quasiorder. In *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
- 60 David Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.
- 61 Ilya Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness*. PhD thesis, Massachusetts Institute of Technology, 2017.
- 62 Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the thirty-fourth annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 360–369, 2002.
- 63 Jeremy P. Spinrad. *Efficient graph representations*. American Mathematical Society, 2003.
- 64 Carsten Thomassen. Girth in graphs. *Journal of Combinatorial Theory, Series B*, 35(2):129–141, 1983.
- 65 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.
- 66 Jan van den Heuvel, Patrice Ossona de Mendez, Daniel Quiroz, Roman Rabinovich, and Sebastian Siebertz. On the generalised colouring numbers of graphs that exclude a fixed minor. *European Journal of Combinatorics*, 66:129–144, 2017.

## A

 Small-Distance Sketching

In this section we prove Theorem 1.2. As in Theorem 3.1, we refine the theorem by showing that the sketches are in fact disjunctive.

► **Theorem A.1.** *Let  $\mathcal{F}$  be a monotone class of graphs. Then the following are equivalent:*

1.  $\mathcal{F}$  is small-distance sketchable.
2. For some function  $f : \mathbb{N} \rightarrow \mathbb{N}$  and every  $r \in \mathbb{N}$ ,  $\mathcal{F}$  admits a disjunctive small-distance labelling scheme of size  $f(r)$ .
3.  $\mathcal{F}$  is first-order sketchable.
4.  $\mathcal{F}$  has bounded expansion.

It holds by definition that (3)  $\implies$  (1) and (2)  $\implies$  (1), even without the assumption of monotonicity. We will prove (4)  $\implies$  (3) and (4)  $\implies$  (2) using different methods. We prove (4)  $\implies$  (3) (again without the assumption of monotonicity) in Section A.2 using the structural result of [26]. This proof does not give explicit bounds on the sketch size. (4)  $\implies$  (2) is proved in Section A.3 and gives explicit upper bounds on the sketch size. The final piece of the theorem, (1)  $\implies$  (4), is proved in Section A.4.

### A.1 Bounded expansion

► **Definition A.2** (Weakly  $r$ -reachable). *Given a total order  $(V, <)$  on the vertex set  $V$  of a graph  $G$  and an integer  $r \geq 0$ , we say that a vertex  $v \in V$  is weakly  $r$ -reachable from a vertex  $u \in V$  if there is a path of length at most  $r$  connecting  $v$  to  $u$  in  $G$ , and such that for any vertex  $w$  on the path,  $v \leq w$  (in words,  $v$  is the smallest vertex on the path with respect to  $(V, <)$ ). For a graph  $G$  and an integer  $r \geq 0$ , we denote by  $\text{wcol}_r(G)$  the smallest integer  $k$  for which the vertex set of  $G$  has a total order  $(V, <)$  such that for any vertex  $u \in V$ , at*



most  $k$  vertices are weakly  $r$ -reachable from  $u$  with respect to  $(V, <)$ . For a graph class  $\mathcal{F}$ , we write  $\text{wcol}_r(\mathcal{F})$  for the supremum of  $\text{wcol}_r(G)$ , for  $G \in \mathcal{F}$ .

► **Definition A.3** ( $(k, \ell)$ -Subdivisions). For a graph  $G$  and two integers  $0 \leq k \leq \ell$ , a  $(k, \ell)$ -subdivision of  $G$  is any graph obtained from  $G$  by subdividing each edge of  $G$  at least  $k$  times and at most  $\ell$  times (i.e. we replace each edge of  $G$  by a path with at least  $k$  and at most  $\ell$  internal vertices). A  $(k, k)$ -subdivision is also called a  $k$ -subdivision for simplicity;

► **Definition A.4** (Depth- $r$  Topological Minor). We say that  $H$  is a depth- $r$  topological minor of a graph  $G$  if  $G$  contains a  $(0, 2r)$ -subdivision of  $H$  as a subgraph. In the proof below it will be convenient to use the following equivalent definition of bounded expansion [57].

► **Theorem A.5.** For a class  $\mathcal{F}$  of graphs, the following are equivalent:

1.  $\mathcal{F}$  has bounded expansion.
2. There is a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for any  $r \in \mathbb{N}$ ,  $\text{wcol}_r(\mathcal{F}) \leq f(r)$ .
3. There is a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for any  $r \in \mathbb{N}$  and any  $G \in \mathcal{F}$ , any depth- $r$  topological minor of  $G$  has average degree at most  $f(r)$ .

We will also require the following standard fact about the expansion of monotone classes, which is a simple consequence of Theorem A.5 (see for instance [58]) combined with a result of Kühn & Osthus [50].

► **Corollary A.6.** Let  $\mathcal{F}$  be a monotone class of unbounded expansion. Then there is a constant  $r \geq 0$ , so that for any  $d \geq 0$ ,  $\mathcal{F}$  contains an  $r$ -subdivision of a bipartite graph of minimum degree at least  $d$  and girth at least 6.

The proof is omitted here due to the space limitation.

## A.2 Bounded Expansion Implies FO Labelling Schemes

To prove that any class of bounded expansion is first-order sketchable, we use the result of [26] that shows how to decompose any class of (structurally) bounded expansion into a number of graphs of bounded shrubdepth. We will require an adjacency sketch for classes of bounded shrubdepth, given below.

### A.2.1 Adjacency Sketching for Bounded Shrubdepth

We must first define shrubdepth. A *connection model* for a graph  $G$  is a rooted tree  $T$  whose nodes are colored with a bounded number of colors such that:

- the vertices of  $G$  are the leaves of  $T$ ; and
- for two vertices  $u, v \in V(G)$ , whether  $u$  and  $v$  are adjacent in  $G$  depends only on the colors of  $u$  and  $v$  in  $T$ , and the color of the lowest common ancestor of  $u$  and  $v$  in  $T$ .

To avoid ambiguity, we say  $G$  has *vertices* while  $T$  has *nodes*. Note that we can assume without loss of generality that all leaves are at the same distance from the root in  $T$ . A class  $\mathcal{G}$  has *bounded shrubdepth* if there are some  $d, k \in \mathbb{N}$  such that every  $G \in \mathcal{G}$  has a connection model of depth  $d$  with colors in  $[k]$  (we recall that the depth of a rooted tree  $T$  is the maximum number of vertices on a root-to leaf path in  $T$ ).

► **Lemma A.7.** Any class  $\mathcal{G}$  of bounded shrub-depth admits a constant-size equality-based adjacency labelling scheme.

**Proof.** Let  $d, k$  be such that any graph  $G \in \mathcal{G}$  has a connection model  $T_G$  of depth  $d$  using color set  $[k]$ . We denote by  $\varphi_G : [k]^3 \rightarrow \{0, 1\}$  the function such that if  $u$  has color  $a$ ,  $v$  has

color  $b$ , and the lowest common ancestor of  $u$  and  $v$  has color  $c$  in  $T_G$ , then  $u$  and  $v$  are adjacent in  $G$  if and only if  $\varphi_G(a, b, c) = 1$ . For every node  $u$  of  $T_G$ , write  $\chi(u)$  for the color of  $u$  in the connection model.

We now construct our equality-based labels for  $G$ . For any vertex  $x$ , let  $t_1(x), t_2(x), \dots, t_d(x)$  be the leaf-to-root path for  $x$ , where  $t_1(x) = x$  and  $t_d(x)$  is the root of  $T_G$ . Then the label for  $x$  is the sequence  $(\varphi_G \mid -), (\chi(t_1(x)) \mid t_1(x)), \dots, (\chi(t_d(x)) \mid t_d(x))$ .

On inputs

$$\begin{aligned} &(\varphi_G \mid -), (\chi(t_1(x)) \mid t_1(x)), \dots, (\chi(t_d(x)) \mid t_d(x)), \\ &(\varphi_G \mid -), (\chi(t_1(y)) \mid t_1(y)), \dots, (\chi(t_d(y)) \mid t_d(y)), \end{aligned}$$

the decoder operates as follows. It finds the smallest  $i \in [d]$  such that  $\mathbf{1}[t_i(x) = t_i(y)]$  and outputs  $\varphi_G(\chi(t_1(x)), \chi(t_1(y)), \chi(t_i(x)))$ .

The correctness of this labelling scheme follows from the fact that we will have  $t_i(x) = t_i(y)$  if and only if the node  $t_i(x) = t_i(y)$  is an ancestor of both  $x$  and  $y$  in  $T_G$ , so the smallest  $i \in [d]$  such that  $t_i(x) = t_i(y)$  identifies the lowest common ancestor of  $x$  and  $y$  in  $T_G$ . ◀

### A.2.2 Structurally Bounded Expansion Implies First-Order Sketching

Following [26], we say that a class of graphs has *structurally bounded expansion* if it can be obtained from a class of bounded expansion by first-order (FO) transductions. We omit the precise definition of FO transductions in this paper, as they are not necessary to our discussion, and instead refer the reader to [26]. We just note that a particular case of FO transduction is the notion of *FO interpretation*, which is of specific interest to us. Consider an FO formula  $\phi(x, y)$  with two free variables and relational vocabulary  $\Sigma = \{F, R_1, \dots, R_k\}$  where  $F$  is symmetric of arity 2. We will say that a graph class  $\mathcal{F}'$  is an FO interpretation of a graph class  $\mathcal{F}$  with respect to  $\phi$  if for any graph  $G' = (V, E') \in \mathcal{F}'$  there is a graph  $G = (V, E) \in \mathcal{F}$  and a  $\Sigma$ -structure with domain  $V$  where  $E$  is the interpretation of the symbol  $F$ , such that for any pair  $u, v \in V$ ,  $uv \in E'$  if and only if  $\phi(u/x, v/y)$  evaluates to true. For instance, if  $\phi(u/x, v/y)$  encodes the property  $\text{dist}_G(u, v) \leq r$  for some fixed integer  $r \geq 1$  (which can be written as an FO formula), then the corresponding FO interpretation of the class  $\mathcal{F}$  is the class of all graph powers  $\{G^r \mid G \in \mathcal{F}\}$ . FO transductions are slightly more involved, as it is allowed to consider a bounded number of copies of a graph before applying the formula, and then it is possible to delete vertices. We will use the following structural result for classes of structurally bounded expansion, proved in [26].

► **Theorem A.8** ([26]). *A class  $\mathcal{G}$  of graphs has structurally bounded expansion if and only if the following condition holds. For every  $p \in \mathbb{N}$ , there is a constant  $m = m(p)$  such that for every graph  $G \in \mathcal{G}$ , one can find a family  $\mathcal{F}(G)$  of vertex subsets of  $G$  with  $|\mathcal{F}(G)| \leq m$  and the following properties:*

- *for every  $X \subseteq V(G)$  with  $|X| \leq p$ , there is  $A \in \mathcal{F}(G)$  such that  $X \subseteq A$ ; and*
- *the class  $\{G[A] \mid G \in \mathcal{G}, A \in \mathcal{F}(G)\}$  of induced subgraphs has bounded shrubdepth.*

We directly deduce the following result.

► **Lemma A.9.** *Any class  $\mathcal{G}$  of structurally bounded expansion admits a constant-size equality-based adjacency labelling scheme.*

**Proof.** Let  $m$  and  $\mathcal{F}$  be given by applying Theorem A.8 to  $\mathcal{G}$  with  $p = 2$ . By definition, for every graph  $G \in \mathcal{G}$  and every pair of vertices  $u, v \in V(G)$ , there is a set  $A \in \mathcal{F}(G)$  containing  $u$  and  $v$ . Moreover,  $\mathcal{F}(G)$  contains at most  $m$  sets and the family  $\mathcal{C}$  of all graphs  $G[A]$ , for

$G \in \mathcal{G}$ , and  $A \in \mathcal{F}(G)$ , has bounded shrubdepth. It follows from Lemma A.7 that there is a constant-size equality-based adjacency labelling scheme for  $\mathcal{C}$ . We denote the decoder of this scheme by  $D$ , and the corresponding labels as  $\ell'_{G[A]}$ .

Consider some graph  $G \in \mathcal{G}$ , and let  $\mathcal{F}(G) = \{A_1, \dots, A_m\}$ . For each vertex  $x$  of  $G$  and  $i \in [m]$ , we write  $a(x) = (a_1(x), \dots, a_m(x))$  where  $a_i(x) = \mathbf{1}[x \in A_i]$ . Then we define the label for  $x$  by taking the prefix  $a(x)$  and appending the labels  $\ell'_{G[A_i]}(x)$  for each induced subgraph  $G[A_i] \in \mathcal{C}$  to which  $x$  belongs. Given the labels for vertices  $x$  and  $y$ , the decoder finds any  $i \in [m]$  such that  $a_i(x) = a_i(y) = 1$ ; and outputs  $D'(\ell'_{G[A_i]}(x), \ell'_{G[A_i]}(y))$ . Such a number  $i \in [m]$  always exists due to Theorem A.8. The correctness of this labelling scheme follows from Theorem A.8 and Lemma A.7.  $\blacktriangleleft$

Since FO-transductions compose (see e.g. [59]), sketching FO formulas in a class of structurally bounded expansion is equivalent to sketching adjacency in another class of structurally bounded expansion. We obtain the following direct corollary of Theorem A.9.

► **Corollary A.10.** *Any class  $\mathcal{G}$  of structurally bounded expansion is first-order sketchable.*

As the property  $\text{dist}_G(x, y) \leq r$  can be written as an FO formula, this directly implies that classes of bounded expansion are small-distance sketchable. However, this does not tell anything on the size of the sketches as a function of  $r$ , unlike the approach using weak coloring numbers described in the next section.

### A.3 Bounded Expansion Implies Small-Distance Sketching

Recall the definition of weak reachability from Definition A.2. We give a quantitative bound on the small-distance sketch of any graph class  $\mathcal{F}$  in terms of  $\text{wcol}_r(\mathcal{F})$ . Recall from Theorem A.5 that any class with bounded expansion has  $\text{wcol}_r(\mathcal{F}) \leq f(r)$  for some function  $f(r)$ ; therefore we obtain the existence of small-distance sketches for any class of bounded expansion.

► **Theorem A.11.** *For any  $r \in \mathbb{N}$ , any class  $\mathcal{F}$  has an  $(0, r, \text{wcol}_r(\mathcal{F}))$ -disjunctive distance- $(r, r)$  labelling scheme.*

**Proof.** Let  $G \in \mathcal{F}$ , and consider a total order  $(V, \prec)$  such that for any vertex  $x \in V$ , at most  $\text{wcol}_r(\mathcal{F})$  vertices are weakly  $r$ -reachable from  $x$  in  $G$  with respect to  $(V, \prec)$ . We say that vertex  $y \in V$  has  $x$ -rank  $k$  if  $y$  is weakly  $k$ -reachable from  $x$  but not weakly  $(k-1)$ -reachable from  $x$ . For each vertex  $x$  and  $k \in [r]$ , write  $S_k(x)$  for the set of vertices  $y$  with  $x$ -rank  $k$ .

We construct a disjunctive labelling scheme as follows. Each vertex  $x$  is assigned the label

$$(- \mid \vec{q}_1(x)), (- \mid \vec{q}_2(x)), \dots, (- \mid \vec{q}_{r'}(x))$$

where  $r' \leq r$  is the maximum number such that  $S_{r'}(x) \neq \emptyset$ , and the equality codes  $\vec{q}_i(x)$  are names of vertices in the set  $S_i(x)$ . Each label contains at most  $\text{wcol}_r(G)$  equality codes, plus a constant number of bits per equality code and  $O(r)$  bits to separate the elements of the list. Given labels for  $x$  and  $y$ , the decoder outputs 1 if and only if there exist  $0 \leq i, j \leq r$  such that  $i + j \leq r$  and  $S_i(x) \cap S_j(y) \neq \emptyset$ , which can be checked using the equality codes in  $\vec{q}_i(x)$  and  $\vec{q}_j(y)$ .

Suppose that  $\text{dist}_G(x, y) \leq r$  and let  $P \subseteq V(G)$  be a path of length  $\text{dist}_G(x, y)$ . Let  $z \in P$  be the minimal element of  $P$  with respect to  $\prec$ . Then  $z$  is weakly  $i$ -reachable from  $x$  and weakly  $j$ -reachable from  $y$ , for some values  $i, j$  such that  $i + j \leq r$ . Then  $z \in S_i(x) \cap S_j(y)$ , so the decoder will output 1 given the labels for  $x$  and  $y$ . On the other hand, if the decoder outputs 1, then there are values  $i, j$  such that  $i + j \leq r$  and  $S_i(x) \cap S_j(y) \neq \emptyset$ . Let  $z \in S_i(x) \cap S_j(y)$ , so that  $z$  is weakly  $i$ -reachable from  $x$  and weakly  $j$ -reachable from  $y$ . Then  $\text{dist}_G(x, y) \leq \text{dist}_G(x, z) + \text{dist}_G(z, y) \leq i + j \leq r$ .  $\blacktriangleleft$

We noticed after proving this result that a similar idea was used in [34, Lemma 6.10] to obtain sparse neighborhood covers in nowhere-dense classes.

We will need the following quantitative results for planar graphs and graphs avoiding some specific minor, due to [66].

► **Theorem A.12** ([66]). *For any planar graph  $G$ , and any integer  $r \geq 0$ ,  $\text{wcol}_r(G) \leq (2r+1)\binom{r+2}{2} = O(r^3)$ .*

► **Theorem A.13** ([66]). *For any integer  $t \geq 3$ , any graph  $G$  with no  $K_t$ -minor, and any integer  $r \geq 0$ ,  $\text{wcol}_r(G) \leq \binom{r+t-2}{t-2}(t-3)(2r+2) = O(r^{t-1})$ .*

In the proof of Theorem A.11, the equality codes are just the names of vertices; so we can use  $\lceil \log n \rceil$  bits to encode each of the  $\text{wcol}_r(\mathcal{F})$  equality codes to obtain an adjacency label. Then, combined with Proposition 2.5, we obtain the following corollary:

► **Corollary A.14.** *If a class  $\mathcal{F}$  has bounded expansion, then  $\mathcal{F}$  has a small-distance sketch of size at most  $O(r + \text{wcol}_r(\mathcal{F}) \log(\text{wcol}_r(\mathcal{F})))$ . If  $\mathcal{F}$  is the class of planar graphs, then the sketch has size  $O(r^3 \log r)$  and if  $\mathcal{F}$  is the class of  $K_t$ -minor free graphs for some fixed integer  $t \geq 3$ , then the sketch has size  $O(r^{t-1} \log r)$ . Furthermore,  $\mathcal{F}$  admits a distance- $(r, r)$  labelling scheme of size  $O(r + \text{wcol}_r(\mathcal{F}) \log n)$ ; if  $\mathcal{F}$  is the class of planar graphs, then the scheme has size  $O(r^3 \log n)$  and if  $\mathcal{F}$  is the class of  $K_t$ -minor free graphs, then the scheme has size  $O(r^{t-1} \log n)$ .*

► **Remark A.15.** The fact that the sketch size is independent of the number of vertices in Corollary A.14 implies that the scheme actually works for infinite graphs. It was proved in [43] that for infinite graphs  $G$ ,  $\text{wcol}_r(G)$  is the supremum of  $\text{wcol}_r(H)$  for all finite subgraphs  $H$  of  $G$  (this was actually proved explicitly for the strong coloring numbers instead of the weak coloring numbers, but the proof is the same). This shows that Theorems A.12 and A.13, and thus Corollary A.14, also hold for infinite graphs.

## A.4 Small-Distance Sketching Implies Bounded Expansion

To complete the proof of Theorem A.1, we must show that any monotone class of graphs that is small-distance sketchable has bounded expansion, which we do by contrapositive. In fact, we will prove a stronger statement: even having a weaker  $(r, 5r-1)$ -distance sketch of size  $f(r)$  implies bounded expansion.

► **Theorem A.16.** *Let  $\mathcal{F}$  be a monotone class of graphs and assume that there is a function  $f$  such that for any  $r \geq 1$ ,  $\mathcal{F}$  has a  $(r, 5r-1)$ -distance sketch of size  $f(r)$ . Then  $\mathcal{F}$  has bounded expansion.*

**Proof.** Assume for the sake of contradiction that  $\mathcal{F}$  has unbounded expansion. By Corollary A.6, there is a constant  $k$  such that for every  $d \geq 0$ ,  $\mathcal{F}$  contains a  $k$ -subdivision of some bipartite graph  $G = (V, E)$  of minimum degree at least  $d$  and girth at least 6. Let  $\mathcal{G}$  be the class consisting of the graph  $G$ , together with all its spanning subgraphs. By monotonicity,  $\mathcal{F}$  contains  $k$ -subdivisions of all the graphs of  $\mathcal{G}$ .

Recall the definition of the partial function  $\text{adj}^E$  parameterized by graphs  $H \in \mathcal{G}$ , from the discussion preceding Lemma 3.4. We will show that the  $(k+1, 5(k+1)-1)$ -distance sketch of size  $f(k+1)$  for  $\mathcal{F}$  can be used to obtain a  $\text{adj}^E$ -sketch for  $\mathcal{G}$ , which must have size  $\Omega(\log d)$  due to Lemma 3.4. This is a contradiction since we must have  $f(k) = \Omega(\log d)$  for arbitrarily large  $d$ , whereas  $f(k+1)$  is a constant independent of  $d$ .

Let  $H$  be any spanning subgraph of  $G$  and let  $H^{(k)}$  denote the  $k$ -subdivision of  $H$ . Consider two vertices  $u, v \in V(H) \subseteq V(G)$  that are adjacent in  $G$ . Observe that  $\text{dist}_{H^{(k)}}(u, v) = (k+1)\text{dist}_H(u, v)$ , and thus if  $u, v$  are adjacent in  $H$  then  $\text{dist}_{H^{(k)}}(u, v) \leq k+1$ . Assume now that  $u, v$  are non-adjacent in  $H$ . Since  $u, v$  are adjacent in  $G$ ,  $G$  has girth at least 6, and  $H$  is a spanning subgraph of  $G$ , it follows that in this case  $\text{dist}_H(u, v) \geq 5$ , and thus  $\text{dist}_{H^{(k)}}(u, v) \geq 5(k+1)$ . Therefore, by using the same decoder as the  $(k+1, 5(k+1)-1)$ -distance sketch for  $\mathcal{F}$ , and using the random sketch  $\text{sk}$  defined for  $G$ , we obtain an  $\text{adj}_H^E$ -sketch for  $H$ . This gives an  $\text{adj}^E$ -sketch for  $\mathcal{G}$  of size  $f(k+1)$ . ◀

In our proof of Theorem A.16 we have used Corollary A.6, which is based on the result of [50], stating that every graph of large minimum degree contains a bipartite subgraph of girth at least 6 and large minimum degree. The following stronger statement was conjectured by Thomassen [64].

► **Conjecture A.17** ([64]). *For every integer  $k$ , every graph of sufficiently large minimum degree contains a bipartite subgraph of girth at least  $k$  and large minimum degree.*


If Conjecture A.17 is true, it readily follows from our proof that the constant 5 in Theorem A.16 can be replaced by an arbitrarily large constant.



# Communication Complexity of Collision

Mika Göös 

EPFL, Lausanne, Switzerland

Siddhartha Jain 

EPFL, Lausanne, Switzerland

---

## Abstract

The *Collision problem* is to decide whether a given list of numbers  $(x_1, \dots, x_n) \in [n]^n$  is 1-to-1 or 2-to-1 when promised one of them is the case. We show an  $n^{\Omega(1)}$  randomised communication lower bound for the natural two-party version of Collision where Alice holds the first half of the bits of each  $x_i$  and Bob holds the second half. As an application, we also show a similar lower bound for a weak bit-pigeonhole search problem, which answers a question of Itsykson and Riazanov (CCC 2021).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity

**Keywords and phrases** Collision, Communication complexity, Lifting

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.19

**Category** RANDOM

**Related Version** Full Version: <https://eccc.weizmann.ac.il/report/2022/096/>

**Acknowledgements** We thank anonymous RANDOM reviewers for their helpful comments.

## 1 Introduction

### Collision problem

The *Collision problem*  $\text{COL}_N: [N]^N \rightarrow \{0, 1, *\}$  is the following partial (promise) function. The input is a list of numbers  $z = (z_1, \dots, z_N) \in [N]^N$  where  $N$  is even. The goal is to distinguish between the following two cases, when promised that  $z$  satisfies one of them.

- $\text{COL}_N(z) = 0$  iff  $z$  is 1-to-1, that is, every number in the list  $z$  appears in the list once.
- $\text{COL}_N(z) = 1$  iff  $z$  is 2-to-1, that is, every number in the list  $z$  appears in the list twice.

The Collision problem has been studied exhaustively in quantum query complexity [10, 1, 5, 14, 20, 6, 2, 3, 12]. It was initially introduced to model the task of breaking collision resistant hash functions, a central problem in cryptanalysis. A robust variant of Collision is complete for  $\text{NISZK}$  [8], and consequently it has been featured in black-box oracle separations [21, 9]. The problem has also been used in reductions to show hardness of other problems such as set-equality [23] and various problems in property testing [11]. Upper bounds for Collision has been used to design quantum algorithms for triangle finding [22] and approximate counting [4].

In this paper, we consider a natural bipartite communication version of this problem, where we split the binary encoding of each number between two parties, Alice and Bob. Specifically, for  $N = 2^n$  where  $n$  is even, we will define a bipartite function

$$\text{BiCOL}_N: (\{0, 1\}^{n/2})^N \times (\{0, 1\}^{n/2})^N \rightarrow \{0, 1, *\}.$$

Here Alice gets as input a list of half-numbers  $x = (x_1, \dots, x_N) \in (\{0, 1\}^{n/2})^N$ , Bob gets a list of half-numbers  $y = (y_1, \dots, y_N) \in (\{0, 1\}^{n/2})^N$ , and we view their concatenation  $z := x \cdot y$ , defined by  $z_i := x_i y_i$ , as an input to  $\text{COL}_N$ . Their goal is to compute  $\text{BiCOL}_N(x, y) := \text{COL}_N(x \cdot y)$ .



© Mika Göös and Siddhartha Jain;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 19; pp. 19:1–19:9



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

### Upper bounds

We first observe that  $\text{BiCOL}_N$  admits a deterministic protocol that communicates at most  $O(\sqrt{N} \log N)$  bits. Indeed, if  $x \cdot y$  is 1–1, then since Alice’s half-numbers are  $n/2$  bits long, there are  $\sqrt{N}$  distinct half-numbers, each appearing  $\sqrt{N}$  many times in  $x$ . We may assume this is true also if  $x \cdot y$  is 2–1 (as otherwise it is easy to tell that we are in case 2–1). Consider the set of indices  $I := \{i \in [N] : x_i = 0^{n/2}\}$ ,  $|I| = \sqrt{N}$ . Then  $x \cdot y$  restricted to indices  $I$  is 1–1 (resp. 2–1) if the original unrestricted input is 1–1 (resp. 2–1). Hence Alice can send the indices  $I$  to Bob, who can determine the value of the function.

If we are allowed randomness, we can do slightly better: there is a randomised protocol of cost  $O(N^{1/4} \log N)$ . In this protocol, Alice samples a subset  $I' \subseteq I$  of size  $|I'| = \Theta(N^{1/4})$  uniformly at random and sends it to Bob, who checks for a collision in his part of the input. If the original input was 2–1, then by the birthday paradox, Bob will observe a collision with high probability.

### Lower bound

As our main result, we prove a small polynomial lower bound for  $\text{BiCOL}_N$ , which shows that the above randomised protocol cannot be improved too dramatically.

► **Theorem 1.**  $\text{BiCOL}_N$  has randomised (and even quantum) communication complexity  $\Omega(N^{1/12})$ .

We conjecture that the  $O(N^{1/4} \log N)$ -bit protocol for  $\text{BiCOL}_N$  is essentially optimal (up to logarithmic factors) for randomised protocols. It is an interesting open problem to close this gap.

## 1.1 Application

### Bit-pigeonhole principle

We also show a lower bound for a search problem associated with the *pigeonhole principle*. We define  $\text{PHP}_N^M$  where  $M > N$  as the following search problem: On input  $z = (z_1, \dots, z_M) \in [N]^M$  the goal is to output a collision, that is, a pair of distinct indices  $i, j \in [M]$  such that  $z_i = z_j$ . We note that  $\text{PHP}_N^M$  is a *total* search problem (not a promise problem); it always has a solution since we require  $M > N$ . As before, we can turn  $\text{PHP}_N^M$  naturally into a bipartite communication search problem  $\text{BiPHP}_N^M$  where  $N = 2^n$  so that

- Alice holds  $x = (x_1, \dots, x_M) \in (\{0, 1\}^{n/2})^M$ ;
- Bob holds  $y = (y_1, \dots, y_M) \in (\{0, 1\}^{n/2})^M$ ; and
- the goal is find a collision, that is, distinct  $i, j \in [M]$  such that  $x_i y_i = x_j y_j$ .

### Lower bounds

Itsykson and Riazanov [17] proved that  $\text{BiPHP}_N^{N+1}$  requires  $\Omega(\sqrt{N})$  bits of randomised communication. Their proof was via a randomised reduction from set-disjointness. A corollary of their result is that any proof system that can be efficiently simulated by randomised protocols (most notably, tree-like  $\text{Res}(\oplus)$  [18]) requires exponential size to refute bit-pigeonhole formulas featuring  $N + 1$  pigeons and  $N$  holes. They asked whether a similar communication lower bound could be proved for the *weak* pigeonhole principle with  $M = 2N$  pigeons and  $N$  holes. We answer their question in the affirmative in the following theorem.



► **Theorem 2.**  $\text{BiPHP}_N^{2N}$  has randomised (and even quantum) communication complexity  $\Omega(N^{1/12})$ .

Previously, Hrubeš and Pudlák [15] showed a small polynomial lower bound for  $\text{BiPHP}_N^M$  for every  $M > N$  against deterministic (and even dag-like) protocols. By contrast, Theorem 2 is the first randomised lower bound in the  $M = 2N$  regime.

## 1.2 Techniques

Our proof of Theorem 1 proceeds as follows. A popular method to prove communication lower bounds is to start with a partial boolean function  $f: \{0, 1\}^n \rightarrow \{0, 1, *\}$  that is hard to compute for decision trees and then apply a *lifting theorem* (we use one due to Sherstov [26]) to conclude that the function  $f \circ g$  obtained by composing  $f$  with a small gadget  $g: \Sigma \times \Sigma \rightarrow \{0, 1\}$  is hard for communication protocols. Here  $f \circ g: \Sigma^n \times \Sigma^n \rightarrow \{0, 1, *\}$  is the communication problem where Alice holds  $x \in \Sigma^n$ , Bob holds  $y \in \Sigma^n$ , and their goal is to output

$$(f \circ g)(x, y) := f(g(x_1, y_1), \dots, g(x_n, y_n)).$$

A straightforward application of lifting often produces communication problems that are “artificial” since they are of the composed form. In particular, at first blush, it seems that the  $\text{BiCOL}_N$  problem cannot be written in the form  $f \circ g$  for any  $f$  and any  $g$  for which a lifting theorem holds. To address this issue, our main technical innovation is to show how the composed function  $\text{COL}_N \circ g$ , where  $g$  is a sufficiently “regular” gadget, can indeed be *reduced* to the natural problem  $\text{BiCOL}_N$ . In this reduction, the input length will blow up polynomially,  $N' = N^{\Theta(1)}$ , which is the main reason why we only get a small polynomial lower bound. Our new reduction generalises a previous reduction from [17, §6], which was tailored for the 2-bit XOR gadget.

To prove Theorem 2 we give a randomised *decision-to-search* reduction from  $\text{BiCOL}_N$  to  $\text{BiPHP}_N^{2N}$ . That is, we show that if there is an efficient randomised protocol for solving the *total* search problem  $\text{BiPHP}_N^{2N}$ , then there is an efficient randomised protocol for solving the *promise* problem  $\text{BiCOL}_N$ . Given this reduction, Theorem 2 then follows from Theorem 1. Similar style of randomised reductions have been considered in prior works [25, 16, 13, 17], although they have always reduced from set-disjointness.

## 2 Reductions and regular functions

We assume some familiarity with communication complexity; see, e.g., the textbooks [19, 24]. In particular, it is often useful to view a bipartite function  $f: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$  as a  $2^n$ -by- $2^n$  boolean matrix. We now give several definitions for the purposes of the proof of our main result.

► **Definition 3** (Rectangular reduction). For bipartite functions  $f, g$  with domains  $\{0, 1\}^n \times \{0, 1\}^n$  and  $\{0, 1\}^m \times \{0, 1\}^m$ , we write  $f \leq g$  if there is a rectangular reduction from  $f$  to  $g$ , that is, there exist  $a: \{0, 1\}^n \rightarrow \{0, 1\}^m$  and  $b: \{0, 1\}^n \rightarrow \{0, 1\}^m$  such that  $f(x, y) = g(a(x), b(y))$  for all  $x, y$ .

Next, using basic language from group theory, we define a new class of highly symmetric boolean functions that we call *regular*. (We borrow the term *regular* from group theory where group actions satisfying the property in Definition 4 below are called *regular*.)

Let  $\Pi_n$  denote the symmetric group on  $[n]$ , that is, the set of all permutations  $[n] \rightarrow [n]$ . Let  $S \subseteq \Pi_n \times \Pi_n$  be any group. We let  $S$  act on the set  $[n] \times [n]$  by permuting the rows and columns, that is, an element  $s = (s^A, s^B) \in S$  acts on  $(x, y) \in [n] \times [n]$  by  $s \cdot (x, y) := (s^A(x), s^B(y))$ . For  $(x, y) \in [n] \times [n]$ , we define its *orbit* by  $S \cdot (x, y) := \{s \cdot (x, y) : s \in S\}$ .

► **Definition 4** (Regular function). A bipartite function  $f: \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$  is regular if there is a group  $S \subseteq \Pi_{2^k} \times \Pi_{2^k}$  acting on the domain of  $f$  such that the orbit of any  $(x, y) \in f^{-1}(b)$ , where  $b \in \{0, 1\}$ , equals  $f^{-1}(b)$ , and, moreover, for every pair of inputs  $(x_1, y_1), (x_2, y_2) \in f^{-1}(b)$  there is a unique  $s \in S$  such that  $s \cdot (x_1, y_1) = s \cdot (x_2, y_2)$ .

It follows from the definition that  $|S| = |f^{-1}(b)| = 2^{2k-1}$  for both  $b \in \{0, 1\}$ . A simple example of a regular function is the 2-bit XOR function together with the 2-element group consisting of the identity map and the map  $(x, y) \mapsto (\neg x, \neg y)$ . However, the XOR function does not satisfy a fully general lifting theorem. This is why we consider the following more complicated gadget, called a *versatile* gadget, which has been shown to satisfy various lifting theorems [26, 13, 7].

► **Definition 5.**  $\text{VER}: \mathbb{Z}_4 \times \mathbb{Z}_4 \rightarrow \{0, 1\}$  is defined by  $\text{VER}(x, y) := 1$  iff  $x + y \pmod{4} \in \{2, 3\}$ .

► **Lemma 6.**  $\text{VER}$  is regular.

**Proof.** Consider the group  $S \subseteq \Pi_4 \times \Pi_4$  generated by the elements  $(x, y) \mapsto (x + 1, y - 1)$  and  $(x, y) \mapsto (1 - x, -y)$  where we use modulo 4 arithmetic. By explicit computations, we see that (here we list each element as a function of  $(x, y)$ )

$$S = \left\{ \begin{array}{llll} (x, y), & (x + 1, y - 1), & (x + 2, y - 2), & (x + 3, y - 3), \\ (1 - x, -y), & (2 - x, 3 - y), & (3 - x, 2 - y), & (-x, 1 - y) \end{array} \right\}.$$

It is straightforward to check that  $S$  gives rise to orbits  $\text{VER}^{-1}(0)$  and  $\text{VER}^{-1}(1)$ ; see Figure 1. Moreover, since  $|S| = 8 = |\text{VER}^{-1}(b)|$  for  $b \in \{0, 1\}$ , the uniqueness property holds, too. ◀

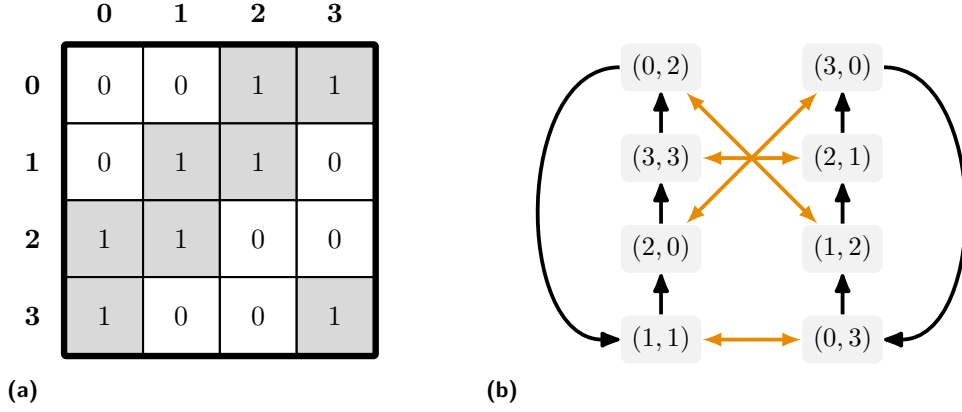
Previously, [13] showed that  $\text{VER}$  is *random self-reducible*, that is, it admits a randomised reduction that maps any fixed input  $(x, y) \in \text{VER}^{-1}(b)$  into a uniform random input in  $\text{VER}^{-1}(b)$ . It is easy to see that if a function is regular, then it is also random self-reducible (the random self-reduction is to apply a random symmetry from  $S$ ). The converse, however, is unclear to us: If  $f$  is random self-reducible and balanced (meaning  $|f^{-1}(0)| = |f^{-1}(1)|$ ), is it necessarily regular?

### 3 Lower bound for bipartite collision

In this section we prove Theorem 1. We start with a standard application of a lifting theorem to establish a lower bound for the (somewhat artificial) composed function  $\text{COL}_N \circ \text{VER}$ . Here we think of  $\text{COL}_N$  as a boolean function  $(\{0, 1\}^n)^N \rightarrow \{0, 1\}$  where  $N = 2^n$ .

► **Lemma 7.**  $\text{COL}_N \circ \text{VER}$  has randomised (and even quantum) communication complexity  $\Omega(N^{1/3})$ .

**Proof.** Aaronson and Shi [5] (building on [1]) showed that  $\deg_{1/3}(\text{COL}_N) \geq \Omega(N^{1/3})$  where  $\deg_{1/3}(f)$  for a partial boolean function  $f$  is the least degree of a multivariate polynomial  $p(x)$  such that  $p(x) = f(x) \pm 1/3$  for all  $x$  such that  $f(x) \in \{0, 1\}$  and  $|p(x)| \leq 4/3$  for all  $x$  with  $f(x) = *$ . Sherstov [26, §12] proved that for any partial boolean function  $f$ , we have that the randomised (and even quantum) communication complexity of  $f \circ \text{VER}$  is at least  $\Omega(\deg_{1/3}(f))$ . Combining these two results proves the lemma. ◀



**Figure 1** (a) The bipartite function  $VER: \mathbb{Z}_4 \times \mathbb{Z}_4 \rightarrow \{0, 1\}$ . (b) The group relative to which  $VER$  is regular is generated by two elements whose actions on  $VER^{-1}(1)$  are illustrated here. The first generator is  $(x, y) \mapsto (x+1, y-1)$  (black arrows) and the second is  $(x, y) \mapsto (1-x, -y)$  (orange arrows).

The challenging part of the proof is to find a reduction from  $COL_N \circ g$  to  $BICOL_{N'}$  where  $g$  is a regular gadget and  $N'$  is polynomially larger than  $N$ . Choosing  $g := VER$  in the following theorem and combining it with Lemma 7 completes the proof of Theorem 1. Note that the input length becomes  $N' := N^4$  so that we obtain the lower bound  $\Omega(N^{1/3}) = \Omega(N'^{1/12})$ , as claimed.

► **Theorem 8.** *Let  $g: \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$  be a regular gadget. For every  $N = 2^n$ ,*

$$COL_N \circ g \leq BICOL_{N^{2k}}.$$

**Proof.** Consider the bipartite function  $COL_N \circ g$ . Alice's input is an  $N$ -tuple  $(a^{(1)}, \dots, a^{(N)})$  where  $a^{(j)} \in (\{0, 1\}^k)^n$  for each  $j \in [N]$ . Bob's input  $(b^{(1)}, \dots, b^{(N)})$  has a similar form. These bipartite inputs encode, via the gadgets, the input  $(z^{(1)}, \dots, z^{(N)})$  to  $COL_N$  such that  $z^{(j)} := g^n(a^{(j)}, b^{(j)}) := (g(a_1^{(j)}, b_1^{(j)}), \dots, g(a_n^{(j)}, b_n^{(j)})) \in \{0, 1\}^n$  where  $a_i^{(j)}, b_i^{(j)} \in \{0, 1\}^k$ .

Let  $S \subseteq \Pi_{2k} \times \Pi_{2k}$  be the symmetry group relative to which  $g$  is regular. Recall that  $|S| = 2^{2k-1}$  and each  $s \in S$  has the form  $s = (s^A, s^B)$  with  $s^A, s^B \in \Pi_{2k}$ . We fix an arbitrary ordering of the elements of  $S$  and write  $S(i)$  for the  $i$ -th element in this ordering. Thus  $S = \{S(1), \dots, S(2^{2k-1})\}$ .

We first describe how the reduction expands each individual input  $(a, b) := (a^{(j)}, b^{(j)})$  to  $g^n$  into an ordered list of inputs to  $g^n$ . In more detail, the reduction

- takes an input  $(a, b) = (a_1, \dots, a_n, b_1, \dots, b_n) \in (\{0, 1\}^k)^{2n}$  to  $g^n$ , and
  - returns  $UNFOLD(a, b) \in (\{0, 1\}^{2kn})^{N^{2k-1}}$ , an ordered list of  $N^{2k-1}$  many inputs to  $g^n$ .
- For any  $n$ -tuple of indices  $I = (i_1, \dots, i_n) \in [|S|]^n$ , we define the  $I$ -th pair in  $UNFOLD(a, b)$  by

$$UNFOLD(a, b)_I := (\underbrace{s_1^A(a_1)s_2^A(a_2)\dots s_n^A(a_n)}_{\text{Alice's half}}, \underbrace{s_1^B(b_1)s_2^B(b_2)\dots s_n^B(b_n)}_{\text{Bob's half}}) \text{ where } s_j := S(i_j).$$

Besides each pair in the list  $UNFOLD(a, b)$  being an input to  $g^n$ , we will also soon interpret them as pairs of half-numbers that are part of the input to  $BICOL_{N^{2k}}$ . Below, we write  $SETUNFOLD(a, b) \subseteq \{0, 1\}^{2kn}$  for the set of elements in the list  $UNFOLD(a, b)$ , that is, ignoring the ordering and multiplicity of elements.

## 19:6 Communication Complexity of Collision

▷ **Claim 9.** We have the following properties.

- (i)  $\text{SETUNFOLD}(a, b) = (g^n)^{-1}(z) = g^{-1}(z_1) \times \cdots \times g^{-1}(z_n)$  where  $z_i := g(a_i, b_i)$ .
- (ii) All pairs in  $\text{UNFOLD}(a, b)$  are distinct.
- (iii) Suppose  $g^n(a, b) \neq g^n(a', b')$ . Then  $\text{SETUNFOLD}(a, b) \cap \text{SETUNFOLD}(a', b') = \emptyset$ .
- (iv) Suppose  $g^n(a, b) = g^n(a', b')$ . Then  $\text{SETUNFOLD}(a, b) = \text{SETUNFOLD}(a', b')$ .

**Proof.** Item (i): Up to reordering of bits, the set equals  $(S \cdot (a_1, b_1)) \times (S \cdot (a_2, b_2)) \times \cdots \times (S \cdot (a_n, b_n))$ . By regularity, the orbit  $S \cdot (a_i, b_i)$  is equal to  $g^{-1}(z_i)$  for any  $i$ . Item (ii): The uniqueness property of the regular group action ensures that we do not get any repeated elements. Item (iii): If  $z := g^n(a, b) \neq g^n(a', b') =: z'$  then there is some  $i$  such that  $z_i \neq z'_i$ . The  $i$ -th component of every pair in  $\text{UNFOLD}(a, b)$  lies in  $g^{-1}(z_i)$  while the  $i$ -th component of every pair in  $\text{UNFOLD}(a', b')$  lies in  $g^{-1}(z'_i)$ . The claim follows since these preimage sets are disjoint. Item (iv): If  $g^n(a, b) = g^n(a', b')$ , then i shows  $\text{UNFOLD}$  produces the same set for both  $(a, b)$  and  $(a', b')$ . ◁

Our final reduction from  $\text{COL}_N \circ g$  maps Alice's  $(a^{(1)}, \dots, a^{(N)})$  and Bob's  $(b^{(1)}, \dots, b^{(N)})$  (which together encode the input  $z = (z^{(1)}, \dots, z^{(N)})$  to  $\text{COL}_N$ ) to an input to  $\text{BiCOL}_{N^{2k}}$  given by

$$\text{UNFOLD}(a^{(1)}, b^{(1)}), \dots, \text{UNFOLD}(a^{(N)}, b^{(N)}).$$

Note that the reduction is rectangular: Alice can compute her part of the input, and Bob his.

It remains to check that the reduction treats 1–1 and 2–1 inputs correctly. If the input  $z$  to  $\text{COL}_N$  is 1–1, then the reduction produces a 1–1 input by ii and iii. If the input  $z$  to  $\text{COL}_N$  is 2–1 then for every index  $i$  there is exactly one more index  $j$  such that  $z^{(i)} := g^n(a^{(i)}, b^{(i)}) = g^n(a^{(j)}, b^{(j)}) =: z^{(j)}$ . Hence, by iv the lists  $\text{UNFOLD}(a^{(i)}, b^{(i)})$  and  $\text{UNFOLD}(a^{(j)}, b^{(j)})$  have every element colliding with each other. This produces a 2–1 input. ◀

### 4 Lower bound for bipartite pigeonhole

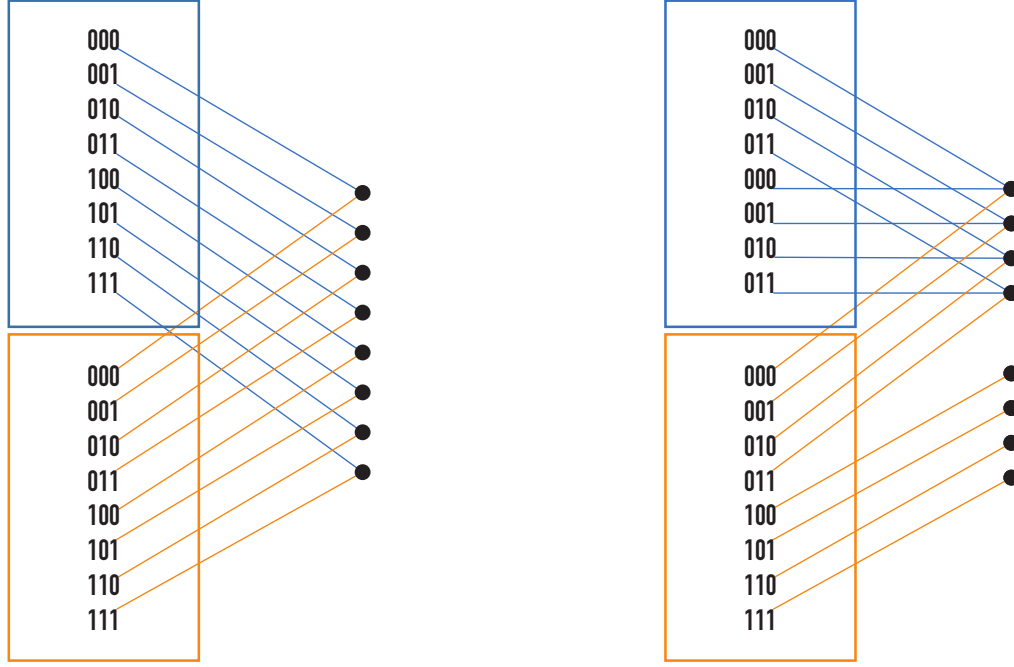
In this section we prove Theorem 2. We do it by describing a reduction from the decision problem  $\text{BiCOL}_N$  to the search problem  $\text{BiPHP}_N^{2N}$ .

► **Theorem 10.** *If there is a randomised protocol for  $\text{BiPHP}_N^{2N}$  of communication cost  $d$ , then there is a randomised protocol for  $\text{BiCOL}_N$  of cost  $O(d)$ .*

**Proof.** The proof idea is to start with an input to  $\text{BiCOL}_N$  and then append it with more numbers to construct an input to  $\text{BiPHP}_N^{2N}$ . Adding more numbers will create some new collisions in the input list, but our reduction will remember which collisions were “planted” during the reduction. We then randomly shuffle the input list so as to make the planted collisions indistinguishable from collisions (if any) coming from the original input to  $\text{BiCOL}_N$ . We now explain this in more detail.

Let  $(x, y)$  be an input to  $\text{BiCOL}_N$ . That is, Alice holds  $x = (x_1, \dots, x_N) \in (\{0, 1\}^{n/2})^N$  and Bob holds  $y = (y_1, \dots, y_N) \in (\{0, 1\}^{n/2})^N$ . In the reduction, we first append Alice's input by the *planted* half-numbers  $(a_1, \dots, a_N) \in (\{0, 1\}^{n/2})^N$  and Bob's input by the *planted* half-numbers  $(b_1, \dots, b_N) \in (\{0, 1\}^{n/2})^N$  where the concatenated strings  $a_i b_i$ ,  $i \in [N]$ , range lexicographically over all binary numbers in  $\{0, 1\}^n$ .

Next, Alice and Bob use public randomness to sample a permutation  $\pi: [2N] \rightarrow [2N]$  uniformly at random, which they then use to permute their lists of length  $2N$ . While doing so, they remember which positions in the permuted list occupy planted numbers (namely,



■ **Figure 2** Illustration of collisions in 1–1 and 2–1 inputs. The original input  $(x, y)$  is drawn at the top, and the planted numbers  $(a, b)$  are drawn at the bottom.

those in positions  $\pi(\{N + 1, \dots, 2N\})$ . Call the resulting list  $(x', y')$ . We now let Alice and Bob run the hypothesised protocol  $\mathcal{P}$  for  $\text{BiPHP}_N^{2N}$  on input  $(x', y')$  to find some collision  $x'_i y'_i = x'_j y'_j$  where  $i \neq j$ . (We assume for simplicity that  $\mathcal{P}$  finds a collision with probability 1. The following analysis can be adapted even when  $\mathcal{P}$  errs with bounded probability.)

We have two cases depending on whether  $(x, y)$  was 1–1 or 2–1 (see Figure 2):

- If  $(x, y)$  was 1–1 then  $(x', y')$  is 2–1. Moreover, each collision in  $(x', y')$  involves a planted number. In particular, the collision  $\{i, j\}$  found by the protocol always features at least one planted number.
- If  $(x, y)$  was 2–1 then  $(x', y')$  is an input where  $N/2$  many numbers appear thrice, and  $N/2$  numbers appear once. We claim that the collision  $\{i, j\}$  found by  $\mathcal{P}$  will not feature a planted number with probability at least  $1/3$  (over the random choice of  $\pi$ ). Indeed, let  $k \notin \{i, j\}$  be the third position such that  $x'_i y'_i = x'_j y'_j = x'_k y'_k$ . Then conditioned on  $\pi$  having produced the input  $(x', y')$ , each position in  $\{i, j, k\}$  is equally likely to occupy a planted number. Thus, with probability  $1/3$ , the planted number lies in position  $k$  and not in  $\{i, j\}$ .

Our protocol for  $\text{BiCOL}_N$  guesses that  $(x, y)$  is 2–1 if the collision  $\{i, j\}$  returned by  $\mathcal{P}$  does not involve a planted number. We can further reduce the error probability down to  $(2/3)^t$  by repeating the randomised reduction and  $\mathcal{P}$  some  $t = O(1)$  times and seeing if any one of these runs finds a collision without a planted number. ◀

## References

- 1 Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of the 34th Symposium on Theory of Computing (STOC)*, pages 635–642. ACM, 2002. doi:10.1145/509907.509999.
- 2 Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. *Quantum Information and Computation*, 12(1-2):21–28, 2012. doi:10.26421/QIC12.1-2-3.
- 3 Scott Aaronson. The collision lower bound after 12 years, 2013. QStart talk. URL: <https://scottaaronson.blog/?p=1458>.
- 4 Scott Aaronson, Robin Kothari, William Kretschmer, and Justin Thaler. Quantum lower bounds for approximate counting via laurent polynomials. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 7:1–7:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.7.
- 5 Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, July 2004. doi:10.1145/1008731.1008735.
- 6 Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: collision and element distinctness with small range. *Theory Comput.*, 1:37–46, 2005. doi:10.4086/toc.2005.v001a003.
- 7 Anurag Anshu, Shalev Ben-David, and Srijita Kundu. On query-to-communication lifting for adversary bounds. In *Proceedings of the 36th Computational Complexity Conference (CCC)*, volume 200, pages 30:1–30:39. Schloss Dagstuhl, 2021. doi:10.4230/LIPICs.CCC.2021.30.
- 8 Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991. doi:10.1137/0220068.
- 9 Adam Boulund, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the power of statistical zero knowledge. *SIAM Journal on Computing*, 49(4):FOCS17–1–FOCS17–58, 2019. doi:10.1137/17m1161749.
- 10 Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Proceedings of the 3rd Latin American Symposium on Theoretical Informatics (LATIN)*, pages 163–169. Springer, 1998.
- 11 Sergey Bravyi, Aram Harrow, and Avinandan Hassidim. Quantum algorithms for testing properties of distributions. *IEEE Transactions on Information Theory*, 57(6):3971–3981, 2011. doi:10.1109/TIT.2011.2134250.
- 12 Mark Bun and Justin Thaler. Dual polynomials for collision and element distinctness. *Theory Comput.*, 12(1):1–34, 2016. doi:10.4086/toc.2016.v012a016.
- 13 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.
- 14 Lov K. Grover and Terry Rudolph. How significant are the known collision and element distinctness quantum algorithms? *Quantum Inf. Comput.*, 4(3):201–206, 2004. doi:10.26421/QIC4.3-5.
- 15 Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 121–131, 2017. doi:10.1109/FOCS.2017.20.
- 16 Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 233–248. ACM, 2012. doi:10.1145/2213977.2214000.
- 17 Dmitry Itsykson and Artur Riazanov. Proof complexity of natural formulas via communication arguments. In *Proceedings of 36th Computational Complexity Conference (CCC)*, volume 200, pages 3:1–3:34. Schloss Dagstuhl, 2021. doi:10.4230/LIPICs.CCC.2021.3.
- 18 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Annals of Pure and Applied Logic*, 171(1):1–31, 2020. doi:10.1016/j.apal.2019.102722.

- 19 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. doi:10.1017/CB09780511574948.
- 20 Samuel Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1(2):29–36, 2005. doi:10.4086/toc.2005.v001a002.
- 21 Shachar Lovett and Jiapeng Zhang. On the impossibility of entropy reversal, and its application to zero-knowledge proofs. In *Proceedings of the 15th Theory of Cryptography Conference (TCC)*, pages 31–55. Springer, 2017. doi:10.1007/978-3-319-70500-2\_2.
- 22 Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, January 2007. doi:10.1137/050643684.
- 23 Gatis Midrijānis. A polynomial quantum query lower bound for the set equality problem. In *Proceedings of the 31st International Conference on Automata, Languages and Programming (ICALP)*, volume 3142, pages 996–1005. Springer, 2004. doi:10.1007/978-3-540-27836-8\_83.
- 24 Anup Rao and Amir Yehudayoff. *Communication Complexity: And Applications*. Cambridge University Press, 2020. doi:10.1017/9781108671644.
- 25 Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, July 1992. doi:10.1145/146637.146684.
- 26 Alexander Sherstov. The pattern matrix method. *SIAM Journal on Computing*, 40(6):1969–2000, 2011. doi:10.1137/080733644.





# Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness

Venkatesan Guruswami ✉

University of California Berkeley, CA, USA

Xin Lyu ✉

University of California Berkeley, CA, USA

Xiuhan Wang ✉

Tsinghua University, Beijing, China

---

## Abstract

In the range avoidance problem, the input is a multi-output Boolean circuit with more outputs than inputs, and the goal is to find a string outside its range (which is guaranteed to exist). We show that well-known explicit construction questions such as finding binary linear codes achieving the Gilbert-Varshamov bound or list-decoding capacity, and constructing rigid matrices, reduce to the range avoidance problem of log-depth circuits, and by a further recent reduction [Ren, Santhanam, and Wang, FOCS 2022] to  $\text{NC}_4^0$  circuits where each output depends on at most 4 input bits.

On the algorithmic side, we show that range avoidance for  $\text{NC}_2^0$  circuits can be solved in polynomial time. We identify a general condition relating to correlation with low-degree parities that implies that any almost pairwise independent set has some string that avoids the range of every circuit in the class. We apply this to  $\text{NC}^0$  circuits, and to small width CNF/DNF and general De Morgan formulae (via a connection to approximate-degree), yielding non-trivial small hitting sets for range avoidance in these cases.

**2012 ACM Subject Classification** Theory of computation → Pseudorandomness and derandomization

**Keywords and phrases** Pseudorandomness, Explicit constructions, Low-depth circuits, Boolean function analysis, Hitting sets

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.20

**Category** RANDOM

**Related Version** Full Version: <https://eccc.weizmann.ac.il/report/2022/102>

**Funding** Venkatesan Guruswami: Research supported in part by NSF CCF-2210823 and a Simons Investigator Award.

**Acknowledgements** We thank anonymous reviewers for helpful comments and suggestions.

## 1 Introduction

We study a basic computational problem in circuit analysis called the *range avoidance* problem (which we call **AVOID** henceforth): given the description of a multi-output Boolean circuit  $C$  mapping  $n$  input bits to  $m := m(n) > n$  output bits<sup>1</sup>, find a  $y \in \{0, 1\}^m$  that is outside the range of  $C$  (i.e.,  $C(x) \neq y$  for every  $x \in \{0, 1\}^n$ ). This is a total search problem that has been the subject of a few recent works [11, 13, 21], which highlight its significance and connections to central themes in computational complexity including circuit complexity, proof complexity, and pseudorandomness.

---

<sup>1</sup> The function  $m(n)$  is called the *stretch* of the circuit.



To gain some intuition about the problem, note that AVOID can be trivially solved by a Monte Carlo algorithm: a random guess would solve AVOID with probability  $1 - 2^{n-m} \geq \frac{1}{2}$ . There is also a straightforward  $\text{ZPP}^{\text{NP}}$  algorithm for AVOID: the algorithm just repeatedly samples a string  $y \in \{0, 1\}^m$  and tests if  $y \in \text{Range}(C)$  by calling the NP oracle. Remarkably, the work by Korten [13] showed that if we can deterministically solve AVOID, then we can obtain explicit constructions of many important objects in CS theory and mathematics, including Ramsey graphs, two-source extractors, rigid matrices, Boolean functions hard against polynomial-size circuits, etc. These reductions put AVOID in a central position among several notoriously hard explicit construction questions that have resisted attack for decades.

In this work, we study AVOID problem for low-depth Boolean circuits (in particular,  $\text{NC}^0$  and  $\text{NC}^1$  circuits). For every constant  $k \geq 1$ , we say a circuit  $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is an  $\text{NC}_k^0$ -AVOID instance, if each output bit of  $C$  depends on at most  $k$  input bits. Similarly, we say  $C$  is an  $\text{NC}_k^1$  instance, if each output bit of  $C$  can be computed by a  $(k \log n)$ -depth Boolean circuit of fan-in two. A recent work by Ren, Santhanam and Wang [21] demonstrates some attractive motivations to study AVOID problem for these weak circuit models. In particular, they showed the following.

► **Theorem 1** (Theorem 5.8 of [21]). *Suppose there is an FP (resp.  $\text{FP}^{\text{NP}}$ )<sup>2</sup> algorithm for  $\text{NC}_4^0$ -AVOID. Then the following statements are true.*

- *For every  $k \geq 1$ , there is an FP (resp.  $\text{FP}^{\text{NP}}$ ) algorithm for  $\text{NC}_k^1$ -AVOID.*
- *For every  $\varepsilon > 0$ , there is a family of functions in E (resp.  $\text{E}^{\text{NP}}$ ) that does not have Boolean circuits of depth  $n^{1-\varepsilon}$ .*

Item (1) shows that  $\text{NC}_4^0$ -AVOID is as hard as  $\text{NC}^1$ -AVOID. Item (2) shows that finding explicit Boolean functions hard against low-depth circuits can be reduced to  $\text{NC}_4^0$ -AVOID. Together, these connections demonstrate that studying AVOID for weak circuit classes is already challenging and fruitful. This suggests two new research directions to approach AVOID from above and below: (i) we can show the “usefulness” of AVOID for “weak” circuit classes by reducing further explicit construction problems to it, and (ii) starting from weak circuit classes such as  $\text{NC}_2^0$ , we can try to design algorithms for AVOID of increasingly powerful models. Ultimately, we aim for an AVOID algorithm for a circuit class expressive enough to capture some elusive explicit construction questions.

## 1.1 Our Results

In this work, we make progress on both directions mentioned above. On the one hand, we reduce a sample of famous explicit construction problems to  $\text{NC}^1$ -AVOID. This improves the previous results by Korten [13], who only showed reductions to AVOID of general polynomial-size circuits. Reducing the explicit construction problems to  $\text{NC}^1$ -AVOID makes them potentially more tractable.

On the other hand, towards solving AVOID of low-depth circuits *unconditionally*, we offer two approaches to design deterministic algorithms for AVOID of low-depth circuits. We give a simple deterministic algorithm for  $\text{NC}_2^0$ -AVOID, and a novel approach to construct *hitting sets* for AVOID instances. This is to say, for a class of circuits  $\mathcal{C} \subseteq \{C : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$  that satisfy certain conditions, we can deterministically construct a set  $S \subseteq \{0, 1\}^m$  of size  $|S| = \text{poly}(m)$ , such that for every  $C \in \mathcal{C}$ , we have  $S \not\subseteq \text{Range}(C)$ . Note that a hitting set

<sup>2</sup> Recall that  $\text{FP}, \text{FP}^{\text{NP}}$  are function classes analogue of the decision problem classes  $\text{P}, \text{P}^{\text{NP}}$ .

construction implies an  $\text{FP}^{\text{NP}}$  algorithm to solve  $\text{AVOID}$  of  $\mathcal{C}$ . It is incomparable to an  $\text{FP}$  algorithm, because the hitting set is *oblivious* to the actual circuit, and the same hitting set can work for a broad class of “weak” circuits.

In the following, we elaborate on our contributions and their implications.

### 1.1.1 Reductions to $\text{NC}^1\text{-AVOID}$

As our first set of results, we reduce a sample of famous and central explicit construction questions to  $\text{NC}_k^1\text{-AVOID}$  for constant  $k$ . In particular, we consider the following explicit construction tasks.

- Rigid matrices. A matrix  $M \in \mathbb{F}_2^{n \times n}$  is called  $(\varepsilon, \delta)$ -rigid, if one cannot reduce the rank of  $M$  to  $\varepsilon n$  by alternating at most  $\frac{\delta n^2}{\log n}$  entries in  $M$ . The motivation to study explicit constructions of rigid matrices is due to its connection to circuit lower bounds [26].
- Binary linear codes which meet the Gilbert-Varshamov bound (the best known rate vs. distance trade-off for binary codes which is achieved by random linear codes). This is an outstanding challenge that has been open for much of coding theory’s history. Recently there has been impressive progress in the low-rate regime [23], but the general question remains a tantalizing challenge at the intersection of coding theory and pseudorandomness.
- Binary linear codes that achieve list-decoding capacity. While there are explicit codes over large alphabets that achieve list-decoding capacity (i.e., are decodable up to the information-theoretically largest fraction of worst-case errors with small lists) [7], the best known binary codes fall well short of achieving capacity [8].

We reduce these explicit construction questions to  $\text{AVOID}$ . We first define explicit construction problems in the complexity-theoretic language: let  $\Pi \in \{\text{LINEAR CODE}, \text{LIST-DECODABLE CODE}, \text{RIGID MATRIX}\}$  be a property of algebraic objects. Define the  $\Pi$ -construction problem: given as input  $1^n$ , output an object of size  $n$  that satisfies the property  $\Pi$ .

► **Theorem 2 (Informal).** *Suppose that for each  $k \geq 1$ , there is an  $\text{FP}$  (resp.  $\text{FP}^{\text{NP}}$ ) algorithm for  $\text{NC}_k^1\text{-AVOID}$ . Then, there is an  $\text{FP}$  (resp.  $\text{FP}^{\text{NP}}$ ) algorithm for  $\Pi\text{-Construction}$  for  $\Pi \in \{\text{LINEAR CODE}, \text{LIST-DECODABLE CODE}, \text{RIGID MATRIX}\}$ .*

Furthermore, by Theorem 1, the same conclusion holds if we assume the existence of an  $\text{FP}$  (resp.  $\text{FP}^{\text{NP}}$ ) algorithm for  $\text{NC}_4^0\text{-AVOID}$ .

Our reductions for linear codes are new, and the reduction for rigid matrices improves a similar result in [13], in the sense that we reduce the question to  $\text{AVOID}$  on *logarithmic-depth* circuits. Our technique is general enough that it can be applied to many other construction problems to give reductions to  $\text{AVOID}$  of low-depth circuits<sup>3</sup>. For brevity, we only present three representative examples in this paper.

**Proof idea.** All of the three reductions follow the same framework. To illustrate the idea, we briefly discuss the reduction for rigid matrices. We follow the idea of Korten [13]. That is, we carefully construct a circuit  $C : \{0, 1\}^{n^2-1} \rightarrow \{0, 1\}^{n^2}$ , whose outputs, when interpreted as matrices in  $\mathbb{F}_2^{n \times n}$ , contain all “non-rigid” matrices. To design the circuit, note that if a

<sup>3</sup> However, we note that the reduction for two-source extractors in [13] might be an exception. Still, by combining [13] with our technique, one can reduce two-source extractor construction to  $\text{NC}^2\text{-AVOID}$ . i.e., each output can be computed by a Boolean circuit of depth  $O(\log^2 n)$ .

matrix  $M \in \mathbb{F}_2^{n \times n}$  is not rigid, then there is a way to compress the matrix. Namely, we can write  $M = L \cdot R + S$  where  $L, R$  are  $n \times \varepsilon n$  and  $\varepsilon n \times n$  matrices, and  $S$  is a sparse matrix with only  $\frac{\delta n^2}{\log n}$  entries being 1. Note that for  $\varepsilon, \delta \in (0, 1)$  sufficiently small, we can encode  $L, R, S$  with  $2\varepsilon n^2 + 2 \log n \cdot \frac{\delta n^2}{\log n} < n^2$  bits and recover  $M$  in polynomial time. In more detail, the encoding just stores  $L, R$  explicitly, and stores a list of  $\frac{\delta n^2}{\log n}$  pairs  $(x, y) \in [n]^2$ , specifying the non-zero entries of  $S$ .

Given this encoding, the reduction to AVOID is simple: we design a circuit  $C$  as follows. The input to  $C$  is a tuple  $(L, R, S)$ , where  $L, R$  are  $n \times \varepsilon n$  and  $\varepsilon n \times n$  matrix, respectively.  $S$  is a list of  $\frac{n^2}{\log n}$  pairs describing a sparse matrix  $S$ . Given the tuple, the circuit  $C$  computes the matrix  $L \cdot R + S$ . It is easy to see that the range of  $C$  includes every non-rigid matrix. Hence, we can construct a rigid matrix by finding a matrix outside the range of  $C$ . However, it is not clear from the reduction whether  $C$  can be implemented in logarithmic depth.

In fact, computing  $L \cdot R$  can be done by a logarithmic circuit easily. If the matrix  $S$  is presented in its natural form as a square matrix, adding  $S$  to  $L \cdot R$  is also easy. Therefore, the main bottleneck in this reduction is to recover the sparse matrix  $S$  from its short description  $S$ . Note that using a short encoding of  $S$  is essential for the reduction, as we need to ensure that the input length is strictly smaller than  $n^2$ . Still, there is some room for manoeuvre: it is not necessary to encode  $S$  in an information-theoretically optimal way, and we *can* afford a certain amount of redundancy, as long as the overall number of bits to encode  $L, R, S$  is bounded by  $n^2 - 1$ .

**Succincter comes into play.** We achieve the improvement by utilizing techniques from succinct data structures (see, e.g., [17, 31]). Succinct data structures allow storage of a data set using an amount of memory that is close to the information-theoretic lower bound, but they still allow for retrieving information efficiently. In particular, there is a classic data structure [17], which can store an  $n$ -bit string of Hamming weight  $k$  using  $\log \binom{n}{k} + O(n/\log^2 n)$  bits. Moreover, one can recover any bit of the string by querying at most  $O(\log n)$  bits in the memory. This data structure perfectly fits our purpose: we can encode the sparse matrix  $S$  by the memory configuration<sup>4</sup> of the data structure storing  $S$ , which is denoted by  $S'$  in the following. Then, we can recover each entry of  $S$  by querying  $O(\log n)$  bits in  $S'$ . By a simple construction (Lemma 26), this implies that each entry of  $S$  can be computed by a logarithmic-depth circuit given  $S'$ .

Therefore, given  $L, R$  and  $S'$ , there is a logarithmic-depth circuit  $C'$  that computes  $L \cdot R + S$ . The number of bits to describe  $L, R, S'$  is bounded by

$$2\varepsilon n^2 + \log \left( \frac{n^2}{\log n} \right) + O(n^2/\log^2 n) < (1 - \Omega(1))n^2 + O(n^2/\log^2 n) < n^2.$$

Hence,  $C'$  is a valid  $\text{NC}^1$ -AVOID instance, and any matrix outside the range of  $C'$  is  $(\varepsilon n, \frac{\delta n^2}{\log n})$ -rigid.

This completes the proof sketch for the rigid matrix reduction. Reductions for linear codes follow the same approach. Namely, every generator matrix  $M$  that fails to generate a desired code can be compressed, where the compression of  $M$  consists of a structured algebraic part  $A$  and a low-Hamming weight binary string  $B$ . The structured part  $A$  has an efficient encoding/decoding scheme, and the combination of  $A$  and  $B$  to recover  $M$  is

<sup>4</sup> In our application, we do not care about the complexity of preparing the data structure, as the AVOID problem asks one to avoid every output in the range of the circuit.

also efficiently computable. Using a naive encoding scheme for  $B$  results in an inefficient (in terms of circuit depth) decoding procedure. Replacing the naive encoding scheme with the succinct data structure gives the desired efficient reduction.

### 1.1.2 Unconditional Algorithms for AVOID of Weak Circuits

On the positive side, we show an algorithm for  $\text{NC}_2^0\text{-AVOID}$ , as well a hitting set construction for solving AVOID of low-depth circuits and large stretch.

**A polynomial time algorithm for  $\text{NC}_2^0\text{-AVOID}$ .** When the given circuit  $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is in  $\text{NC}_2^0$  (i.e., each output bit depends on only two input bits), we can solve AVOID of  $C$  by a simple deterministic polynomial-time algorithm.

► **Theorem 3.** *There is a polynomial time algorithm which, given an  $\text{NC}_2^0$  circuit  $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m > n$ , outputs a string  $y \in \{0, 1\}^m$  that is not in the range of  $C$ .*

The idea behind Theorem 3 is simple. Let  $C_1(x)$  be the first output bit of  $C$ . We observe that there is always a way to fix  $C_1$  to a constant, so that we can reduce the problem to solving  $\text{NC}_2^0\text{-AVOID}$  for a smaller circuit  $C' : \{0, 1\}^{n-1} \rightarrow \{0, 1\}^{m-1}$ . To illustrate, suppose that  $C_1(x)$  is an AND of two variables (say,  $x_1$  and  $x_2$ ). Then, by setting  $C_1$  to 1, we have effectively restricted that  $x_1$  and  $x_2$  must be 1. Hence, we can replace every appearance of  $x_1, x_2$  with constant 1 in  $C$ , and get a new  $\text{NC}_2^0\text{-AVOID}$  instance  $C' : \{0, 1\}^{n-2} \rightarrow \{0, 1\}^{m-1}$ . Suppose  $y \in \{0, 1\}^{m-1}$  is not in the range of  $C'$ . Then we claim that  $1 \circ y$  (where  $\circ$  denotes string concatenation) is not in the range of  $C$ . In fact, for  $C_1(x)$  evaluating to 1, one has to set both  $x_1$  and  $x_2$  as 1. But then there is no way to find an input  $x$  where  $C(x)_{2\dots m} = C'(x) = y$ .

The argument above illustrates one step of the reduction. To design an algorithm for  $\text{NC}_2^0\text{-AVOID}$ , we can recursively apply the reduction, until at one point where we are left with a circuit  $C'' : \{0, 1\}^0 \rightarrow \{0, 1\}^{m-n}$ . At this point,  $C''$  always outputs a fixed string, while the number of possible outputs is  $2^{m-n} > 1$ , which allows us to solve AVOID for  $C''$  trivially. Finally, we can backtrack to recover a string  $y \in \{0, 1\}^m$ , which solves AVOID for the original circuit  $C$ .

Since the result in [21] (see also Theorem 1) gives a strong evidence suggesting that solving  $\text{NC}_4^0\text{-AVOID}$  unconditionally is hard and would imply surprisingly strong circuit lower bounds, the strategy above probably fails to give an algorithm for  $\text{NC}_4^0\text{-AVOID}$ . Still, finding out the complexity of  $\text{NC}_3^0\text{-AVOID}$  remains an interesting question.

**Approaching AVOID via hitting sets.** We also introduce a novel technique for solving AVOID in  $\text{FP}^{\text{NP}}$ . Informally, we show that there is an  $\text{FP}^{\text{NP}}$  algorithm for simple circuits if the stretch  $m(n)$  is large enough. Here is the list of our results.

► **Theorem 4 (Informal).** *Let  $m = m(n), s = s(n)$  be two non-decreasing functions and  $k, w \geq 1$  be two constants. Suppose  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  is a multi-output function. There is an  $\text{FP}^{\text{NP}}$  algorithm for  $\text{AVOID}(C)$  if one of the following statements hold:*

- *Each output bit  $C_i(x)$  depends on only  $k$  input bits and  $m \geq 2^{4k+1}n^{k-1} + n$ ;*
- *Each output bit  $C_i(x)$  is a width- $w$  size- $s$  CNF or DNF of input bits and  $m \geq 32s^2n^w$ ;*
- *Each output bit  $C_i(x)$  is a size- $s$  De Morgan formula of input bits and  $m \geq n^{\omega(\sqrt{s})}$ ;*
- *Each output bit  $C_i(x)$  is a size- $s$  DNF or CNF of input bits and  $m \geq 2^{\omega(n^{1/2} \cdot \log(s))}$ .*

Formally, our result is stronger than  $\text{FP}^{\text{NP}}$  algorithm. Our construction is a hitting set which is independent of the circuit  $C$ . That is, we can output a set of polynomial size which always contains a solution for  $\text{AVOID}(C)$ , without looking at the input circuit  $C$ . We formally list our results here.

► **Theorem 5.** *Let  $m = m(n)$ ,  $s = s(n)$  be two non-decreasing functions and  $k, w \geq 1$  be two constants. Suppose  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  is a multi-output function. The following statements hold.*

- *If each output bit  $C_i(x)$  depends on only  $k$  input bits and  $m \geq 2^{4k+1}n^{k-1} + n$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $2^{O(k)}m^2$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*
- *If each output bit  $C_i(x)$  is a width- $w$  size- $s$  CNF or DNF of input bits and  $m \geq 32s^2n^w$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $O(s^2 \log^2 m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*
- *If each output bit  $C_i(x)$  is a size- $s$  De Morgan formula of input bits and  $m \geq n^{\omega(\sqrt{s})}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*
- *If each output bit  $C_i(x)$  is a size- $s$  DNF or CNF of input bits and  $m \geq 2^{\omega(n^{1/2} \cdot \log(s))}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*

*In all cases, the set  $S$  is independent of the circuit  $C$ . Namely, only knowing  $m, n, s, k, w$  suffices to construct the set  $S$ .*

Perhaps surprisingly, we construct the hitting set by exploiting an interesting connection to pseudorandomness of distributions. In particular, we carry out a two-step plan as follows.

- For a class of simple circuits  $\mathcal{C} \subseteq \{C : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ , we show that if the stretch  $m$  is sufficiently large, then under any input distribution  $\mathbf{x}$  over  $\{0, 1\}^n$ , the output distribution  $C(\mathbf{x})$  cannot be pairwise independent over  $\{0, 1\}^m$ .
- On the other hand, we *can* sample a pairwise independent string of length  $m$ , with only  $2 \log m$  truly random bits.

Putting two items together, we conclude that the support of a low-entropy pairwise independent distribution  $\mathcal{D}$  over  $\{0, 1\}^m$  constitutes a hitting set for  $\text{AVOID}$  of  $\mathcal{C}$ . Indeed, if the support of  $\mathcal{D}$  is contained in  $\text{Range}(C)$  for some  $C \in \mathcal{C}$ , then we know that under a proper input distribution  $\mathbf{x}$  over  $\{0, 1\}^n$ ,  $C(\mathbf{x})$  can sample  $\mathcal{D}$  perfectly, which leads to a contradiction to Item (1).

Here, Item (2) is standard [1]. We achieve Item (1) by generalizing a technique by Mossel, Shpilka and Trevisan [15], where the authors showed that it is impossible for  $\text{NC}_3^0$  circuits to expand  $n$  *uniformly random* bits into a  $(4n + 1)$ -bit string that fools every linear test (i.e., the output fails to be a low-biased distribution). We generalize the [15] result by considering an arbitrary distribution (instead of uniform distribution) over inputs.

We briefly describe the high-level proof strategy below. We start with a simplicity measure of Boolean functions, parameterized by an integer  $d \geq 1$  and a real  $\delta \in (0, 1)$ . A function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is called  $(d, \delta)$ -simple, if under any distribution  $\mathbf{x}$  over  $\{0, 1\}^n$ , there is a parity test over a set  $S \subseteq [n]$  of size  $|S| \leq d$ , such that

$$\left| \Pr_{\mathbf{x}}[f(\mathbf{x}) = \bigoplus_{q \in S} x_q] - \frac{1}{2} \right| \geq \delta.$$

The following theorem shows our general template to construct hitting sets based on simplicity of functions.

► **Theorem 6.** *Suppose  $m > n \geq 2$ . Let  $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be a circuit and  $\varepsilon > 0$  be a parameter. Suppose each output bit  $C_i$  is a  $(d, \varepsilon)$ -simple function of input bits and  $m > \frac{2}{\varepsilon^2}n^d$ . Then, for every distribution  $\mathbf{x}$  over the input space  $\{0, 1\}^n$ , the output distribution  $C(\mathbf{x})$  is not pairwise independent.*



We prove Theorem 6 following the technique of [15]. Let  $\mathbf{x}$  be sampled from an arbitrary but fixed distribution. Since there are  $m \geq \frac{2}{\varepsilon^2} n^d$  outputs and each output is correlated with a parity test on at most  $d$  inputs, by pigeonhole principle, there are at least  $\frac{2}{\varepsilon^2}$  output bits that are  $\varepsilon$ -correlated with the same parity test. Then we follow [15] and carry out a second-moment argument, which shows that there is a pair of indices  $i, j \in [m]$  among the  $\frac{2}{\varepsilon^2}$  outputs, such that  $C_i(\mathbf{x})$  and  $C_j(\mathbf{x})$  have a correlation lower-bounded by  $\frac{3}{8}\varepsilon^2$ , meaning that  $C(\mathbf{x})$  does not sample a pairwise independent distribution.

Note that the argument above also shows a lower bound of the correlation between two output bits. This allows us to use an *almost* pairwise independent distribution in the final construction, which makes the size of our hitting set even smaller. See Section 4 for the details.

Instantiating Theorem 6 with some canonical circuit classes, we deduce the results listed in Theorem 5.

- The results for  $\text{NC}_k^0$  circuits and constant-width DNF/CNFs are proved by ad-hoc but straightforward arguments. We remark that [15] has shown that every  $\text{NC}_k^0$  function is either an  $\mathbb{F}_2$  polynomial of degree  $\lceil k/2 \rceil$  or correlated with a parity test on at most  $\lceil k/2 \rceil$  inputs under the *uniform* distribution of inputs. We managed to prove a correlation lower bound under arbitrary distributions, but we need to use parity tests on at most  $(k-1)$  inputs, which in turn determines that our construction only works for  $\text{NC}_k^0$ -AVOID with stretch at least  $\Omega(n^{k-1})$ . Still this is non-trivial in the sense that prior to our work, even an algorithm for  $\text{NC}_k^0$ -AVOID with stretch  $o(n^k)$  appears to not have been known.
- The results for unbounded-width CNF/DNFs and small-size De Morgan formulae are proved by relating the simplicity of functions to their (large-error) *approximate degree*, a central notion in complexity theory that has been studied extensively (see, e.g., [12, 19, 4]). Specifically, to show the simplicity of a function, it suffices (and, in some sense, is necessary) to find a low-degree polynomial over reals that point-wise approximates the function within a slightly non-trivial error (e.g. within error  $\frac{1}{2} - \frac{1}{n}$ )<sup>5</sup>. This connection allows us to translate known approximate degree upper bounds for CNF/DNF [12] and small-size De Morgan formulae [20] to the simplicity of corresponding function classes.

**Discussions.** We find the connection to pseudorandomness quite interesting. In some sense, following Razborov and Rudich’s natural proof [18], our argument establishes a separation result for weak circuits (with large stretches) by studying a natural property about *distributions*<sup>6</sup> over hypercubes. Namely, we consider the property of being a pairwise independent distribution. By standard pseudorandomness constructions [1], there is a low-entropy distribution that attains this property easily, while our results rule out the possibility of sampling such distributions by weak circuit classes that only receive a short random seed, even if the random seed can come from an arbitrary distribution.

We leave it as an intriguing question to further explore the potential of this framework. Namely, can we identify more (pseudorandom) property of distributions, where there exists a low-entropy (and hopefully polynomial-time constructible) distribution with this property, but every weak circuit from a class  $\mathcal{C}$  fails to sample a distribution with this property, even if its input distribution can be carefully tailored?

<sup>5</sup> Note that the polynomial  $p(x) \equiv \frac{1}{2}$  trivially  $\frac{1}{2}$ -approximates every Boolean function.

<sup>6</sup> This is in contrast with the typical notion of natural proofs, where natural properties of languages/-Boolean functions are considered.

Note that the existence of such a “pseudorandom” property usually implies an efficient statistical test to distinguish the output of  $\mathcal{C}$ -circuits from uniform (in our example, this is a linear test on two output bits of  $\mathcal{C}$ ). Thus, under the cryptography assumption that  $\text{NC}^1$  circuits can compute PRG of polynomial stretch, it seems difficult to push this technique to  $\text{NC}^1$ . Still, we note there is a gap between our results and the best-known lower bounds and pseudorandomness results: for example, we know strong lower bounds and good PRGs against  $\text{AC}^0$  (see e.g. [10, 25, 14]). Moreover, when the input distribution is uniform, we have very good sampling lower bounds against  $\text{AC}^0$  circuits of quasi-polynomial stretches [27, 28]. If, quantitatively, solving  $\mathcal{C}$ -AVOID is as hard as proving lower bounds/constructing PRGs for  $\mathcal{C}$ , then these results suggest that one should be able to solve  $\text{AC}^0$ -AVOID of (large) quasi-polynomial stretches. However, our result can only give a hitting set construction for  $\text{AC}^0$  circuits of sub-exponential stretch<sup>7</sup>. We leave it as an interesting open question to close the gap between the known pseudorandomness results and our hitting sets. Namely, can we give better hitting sets for  $\text{AC}^0$  circuits of smaller stretch, or is there any formal evidence suggesting that AVOID of low-end models (e.g.,  $\text{AC}^0$ ) is strictly harder than designing PRG for  $\text{AC}^0$ ?

**Comparison with previous works.** Attempting to solve AVOID of weak circuits with large stretch, Ren, Santhanam and Wang [21] presented an algorithmic framework in  $\text{FP}^{\text{NP}}$ , which is based on Williams’ algorithmic method [29] and rectangular PCPs [3]. Our framework is not directly comparable to theirs. A polynomial-size hitting set construction appears to be stronger than an  $\text{FP}^{\text{NP}}$  algorithm, as a hitting set implies an  $\text{FP}^{\text{NP}}$  algorithm in a straightforward way. But our assumption (the existence of a proper “natural property” of distributions) is incomparable to the assumption in [21].

We note that [21] also showed an  $\text{FP}^{\text{NP}}$  algorithm for De Morgan formula-AVOID with stretch  $m \geq 2^{\omega(\sqrt{s} \log(s))}$  as an application of their technique. To devise the algorithm, they also used the approximate degree upper bounds [20] as a key technical ingredient. For this application, our result compares favorably with theirs. First, our hitting set construction is considerably simpler and can also handle a somewhat smaller stretch: the algorithm in [21] needs a “constructive version” of the approximate degree upper bounds, which roughly says that one can deterministically find a degree- $(\sqrt{s} \log s)$  polynomial approximating a given size- $s$  De Morgan formula. The  $\log(s)$  overload in turn determines that their algorithm can only handle stretches larger than  $n^{\omega(\sqrt{s} \log s)}$ . In contrast, our solution only needs the *existence* of a low-degree approximate polynomial, enabling us to construct hitting sets for stretch  $n^{O(\sqrt{s})}$ . Second, the framework in [21] cannot obtain a non-trivial algorithm from large-error ( $\varepsilon = 1 - n^{-\Omega(1)}$ ) approximate degree. In particular, their framework does not naturally apply to polynomial-size DNF/CNFs as our result does.

## 1.2 Conclusion & Open Questions

In this work, we study the range avoidance problem for low depth circuits. We reduce some explicit construction challenges to the range avoidance problem of  $\text{NC}_4^0$  circuits. On the algorithmic side, we give a polynomial time algorithm for  $\text{NC}_2^0$ -range avoidance. We also introduce a hitting set construction for the range avoidance problem of weak circuit classes with large stretch.

---

<sup>7</sup> We explicitly give a construction for depth-2 circuits (e.g., DNFs) with stretch  $2^{n^{1/2}}$ . It is easy to see that we can extend our results to depth- $d$   $\text{AC}^0$  circuit of stretch roughly  $2^{n^{1-\Omega(1/d)}}$  by the known approximate degree upper bound for  $\text{AC}^0$ .



As suggested by [21],  $\text{NC}_4^0\text{-AVOID}$  might be hard to solve. For  $\text{NC}_3^0$ , our hitting set construction works when the stretch is at least  $C \cdot n^2$  for a large constant  $C$ . For smaller stretch, the complexity of  $\text{NC}_3^0$  is less clear. It is natural to ask:

**Open Question 1.** Is there a deterministic polynomial time algorithm for  $\text{NC}_3^0\text{-AVOID}$  with stretch  $n^{1+o(1)}$ , even when an NP oracle is available?

As we have mentioned, our hitting set construction suggests a new approach to solve AVOID for weak computational models. It naturally raises the following questions.

**Open Question 2.** For some weak computational models (e.g.,  $\text{AC}^0$ ), is there a distribution that can be efficiently sampled using a short seed but cannot be sampled by these models? They are some known sampling lower bounds for  $\text{AC}^0$  when the input distribution is uniform [27, 28]. Do techniques in those works help in proving a sampling lower bound under arbitrary distributions?

**Open Question 3.** For a class of circuits  $\mathcal{C} \subseteq \{\{0,1\}^n \rightarrow \{0,1\}^m\}$ , it is easy to see (via the probabilistic method) that there *exists* a hitting set  $H$  for AVOID of  $\mathcal{C}$  with size  $|H| \leq \text{poly}(\log |\mathcal{C}|)$ . Note that such a hitting set constitutes a “universal” solution to explicit construction problems. Namely, for every explicit construction problem  $\Pi$  that is reducible to  $\mathcal{C}\text{-AVOID}$ , there is a string  $x \in H$  that has the property  $\Pi$ . It would be interesting to identify the construction of the hitting set  $H$  itself as an explicit construction problem, and study its complexity and/or algorithms via various kinds of approaches (including the pseudorandomness approach considered in this work).

### 1.3 Organization

The rest of the paper is organized as follows. In Section 2, we put some preliminaries, including the problems we study and some mathematical tools used in our proofs. In Section 3, we show how to reduce some explicit construction problems to  $\text{NC}_4^0\text{-AVOID}$ . In Section 4, we show a general framework to construct hitting sets for AVOID by correlations with low-degree parities. Finally we show some applications of this method.

## 2 Preliminaries

In this section, we state necessary background knowledge and set up some useful pieces of notation.

**Range Avoidance.** We first define the AVOID problem, which is the primary subject of this work.

► **Definition 7** (AVOID [13, 21]). *AVOID is the following problem: given a Boolean circuit  $C : \{0,1\}^n \rightarrow \{0,1\}^m$  where  $m > n$ , find an  $m$ -bit string outside the range of  $C$ . If the input circuit is guaranteed to be in some circuit class  $\mathcal{C}$ , we also call the problem  $\mathcal{C}\text{-AVOID}$ .*

► **Definition 8** (NC circuits). *For each  $k \geq 1$ , we define  $\text{NC}_k^0$  and  $\text{NC}_k^1$  as follows.  $\text{NC}_k^0$  contains all functions that depend on at most  $k$  input bits. For every  $n \geq 1$ ,  $\text{NC}_k^1$  contains all  $n$ -bit functions that are computable by  $(k \log n)$ -depth Boolean circuits of fan-in two.*

We will be mainly interested in  $\text{NC}_k^0\text{-AVOID}$  and  $\text{NC}_k^1\text{-AVOID}$  for constant  $k$ 's. When we say an explicit construction problem reduces to  $\text{NC}^1\text{-AVOID}$ , we mean there exists  $k \geq 1$  such that the problem reduces to  $\text{NC}_k^1\text{-AVOID}$ .

**Explicit Constructions.** We study the following explicit construction problems, starting with rigid matrices.

► **Definition 9** (rigid matrix [26]). Let  $q \geq 1$  be a prime power and  $r, s \geq 1$  be two integers. We say an  $n \times n$  matrix  $M$  over  $\mathbb{F}_q$  is  $(r, s)$ -rigid, if for any matrix  $S \in \mathbb{F}_q^{n \times n}$  with at most  $s$  non-zero entries, the rank of  $M + S$  is at least  $r$ .

An explicit construction of  $(\Omega(n^2), n^{1+\epsilon})$ -rigid matrices would imply a lower bound against linear-size, logarithmic-depth arithmetic circuits [26]. By probabilistic method, a random matrix is  $(\Omega(n^2), \Omega(n^2/\log n))$ -rigid with high probability. This motivates us to formulate the following problem.

► **Definition 10** (RIGID).  $(\epsilon, \delta, q)$ -RIGID is the following problem: given input  $1^n$ , output an  $n \times n$  matrix over  $\mathbb{F}_q$  that is  $(\epsilon n, \frac{\delta n^2}{\log n})$ -rigid.

The next object we consider is linear codes with good rates and distances.

► **Definition 11** (linear code [9]). Let  $r, p \in (0, 1)$ ,  $n \in \mathbb{N}$  and  $k = r \cdot n$ . We say an  $k \times n$  matrix  $G$  of full row rank over  $\mathbb{F}_2$  is a generator matrix of a  $(r, p)$ -linear code, if every two distinct codewords generated by  $G$  have Hamming distance at least  $pn$ , or equivalently, the Hamming weight of any nonzero codeword is at least  $pn$ .

By probabilistic method, for every  $r, p \in (0, 1)$  such that  $r < 1 - h(p)$ , there is a family of linear codes with rate  $r$  and distance  $pn$  (the inequality  $r < 1 - h(p)$  is called Gilbert-Varshamov bound in literature). However, despite an extensive line of efforts, an explicit construction meeting this bound remains widely open. We formulate the linear code construction in the complexity-theoretic language as follows.

► **Definition 12** (LINEARCODE).  $(r, p)$ -LINEARCODE is the following problem: given input  $1^n$ , output a matrix  $G \in \mathbb{F}_2^{rn \times n}$  such that  $G$  is a generator matrix of a  $(r, p)$ -linear code.

Finally, we study linear codes with good list-decoding capacity.

► **Definition 13** (list-decodable code [6, 30]). Let  $r, p \in (0, 1)$ ,  $n \in \mathbb{N}$  and  $k = r \cdot n$ . We say an  $rn \times n$  matrix  $G$  over  $\mathbb{F}_2$  is a generator matrix of a  $(p, L)$ -list decodable code if for every  $z \in \mathbb{F}_2^n$ , the number of codewords  $c \in \text{Im}(G)$  within Hamming distance  $pn$  from  $z$  is at most  $L$ , i.e.

$$|\{s \in \mathbb{F}_2^{rn} : \text{wt}(sG - z) \leq pn\}| \leq L$$

where  $\text{wt}(s)$  denotes the number of ones in the string  $s$ .

The probabilistic method shows the existence of  $(r, p, L)$ -list decodable codes, provided that  $r < 1 - h(p) - \frac{2}{\log_2 L}$ . Again, finding an explicit family of linear codes approaching this limit remains an outstanding challenge.

► **Definition 14** (LISTDECODABLE).  $(r, p, L)$ -LISTDECODABLE is the following problem: given input  $1^n$ , output a matrix  $G \in \mathbb{F}_2^{rn \times n}$  such that  $G$  is a generator matrix of a  $(p, L)$ -list decodable code.

## 2.1 Boolean Functions

In this subsection, we list some useful notations about Boolean functions. To represent a Boolean variable, we sometimes use  $\mathbb{F}_2$  as the domain and sometimes use  $\{-1, 1\}$  as the domain. When the domain is  $\mathbb{F}_2$ , we use 1 to represent **True** and 0 to represent **False**. When the domain is  $\{-1, 1\}$ , we use  $-1$  to represent **True** and 1 to represent **False**.

► **Definition 15** (parity functions). For every set  $S \subseteq [n]$ , define the parity function  $\chi_S : \{-1, 1\}^n \rightarrow \{-1, 1\}$  by

$$\chi_S(x) = \prod_{i \in S} x_i.$$

► **Definition 16** (equality functions). For every set  $S \subseteq [n]$  and  $z \in \{-1, 1\}^n$ , we define the equality function  $\text{EQ}_{S,z}(x)$  to be 1 if  $x_i = z_i$  holds for every  $i \in S$  and 0 otherwise.

It can be easily verified that parity functions and equality functions have the following relation:

► **Fact 17.** For every set  $S \subseteq [n]$  and  $z \in \{-1, 1\}^n$ , we have

$$\text{EQ}_{S,z}(x) = \frac{1}{2^{|S|}} \sum_{T \subseteq S} \left( \prod_{i \in T} z_i \right) \chi_T(x).$$

► **Definition 18** (inner product). For two Boolean functions  $f, g : \{-1, 1\}^n \rightarrow \mathbb{R}$ , we define their inner product by:

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{x \in \{-1, 1\}^n} f(x)g(x).$$

For a given distribution  $\varphi : \{-1, 1\}^n \rightarrow [0, 1]$  where  $\varphi(x) := \Pr_{\mathbf{x} \sim \varphi}[\mathbf{x} = x]$ , we define their inner product over  $\varphi$  by:

$$\langle f, g \rangle_\varphi = \sum_{x \in \{-1, 1\}^n} f(x)g(x)\varphi(x).$$

► **Definition 19** (correlation). We say two Boolean functions  $f, g : \{-1, 1\}^n \rightarrow \{-1, 1\}$  are  $\varepsilon$ -correlated if

$$|\langle f, g \rangle| \geq \varepsilon.$$

For a given distribution  $\varphi : \{-1, 1\}^n \rightarrow [0, 1]$ , we say they are  $\varepsilon$ -correlated under  $\varphi$  if

$$|\langle f, g \rangle_\varphi| \geq \varepsilon.$$

## 2.2 Miscellaneous

► **Definition 20** (binary entropy). The binary entropy function  $h : [0, 1] \rightarrow \mathbb{R}$  is defined as

$$h(p) := -p \log_2 p - (1 - p) \log_2 (1 - p).$$

For a distribution  $\mathcal{D}$ , we use  $\mathbf{x} \sim \mathcal{D}$  to denote that a random variable  $\mathbf{x}$  is drawn from  $\mathcal{D}$ . We then define  $\varepsilon$ -biased distribution and  $\varepsilon$ -almost pairwise independent distribution here.

► **Definition 21** ( $\varepsilon$ -biased distribution). A distribution  $\mathcal{D}$  on  $\{-1, 1\}^n$  is  $\varepsilon$ -biased if for every nonempty  $T \subseteq [n]$ , it holds that

$$-\varepsilon \leq \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[ \prod_{i \in T} x_i \right] \leq \varepsilon.$$

► **Definition 22** ( $\varepsilon$ -almost pairwise independent distribution). *A distribution  $\mathcal{D}$  on  $\{-1, 1\}^n$  is  $\varepsilon$ -almost pairwise independent if for every two distinct indices  $i, j \in [n]$  and vector  $\vec{v} \in \{-1, 1\}^2$ , it holds that*

$$\left| \Pr_{\mathbf{x} \sim \mathcal{D}}[(x_i, x_j) = \vec{v}] - \frac{1}{4} \right| < \varepsilon.$$

We have the following standard constructions of  $\varepsilon$ -biased distribution and  $\varepsilon$ -almost pairwise independent distribution.

► **Theorem 23** ([16, 2]). *For every  $\varepsilon \in (0, 1)$  and  $n \in \mathbb{N}$ , there is an explicit (polynomial-time computable)  $\varepsilon$ -biased distribution with support size  $O(n^2/\varepsilon^2)$ .*

► **Theorem 24** ([2]). *For every  $\varepsilon \in (0, 1)$  and  $n \in \mathbb{N}$ , there is an explicit (polynomial-time computable)  $\varepsilon$ -almost pairwise independent distribution with support size  $O(\log^2 n/\varepsilon^2)$ .*

### 3 Explicit Constructions Reduce to $\text{NC}_4^0$ -AVOID

In this section, we reduce several central explicit construction problems in coding theory and complexity theory to solving AVOID for logarithmic depth circuits.

#### 3.1 Technical Ingredients

We need the following technical tools from literature.

► **Lemma 25** ([17], Theorem 1). *Given an array of  $n$  elements from an alphabet  $\Sigma$ , and let  $f_\sigma > 0$  be the number of occurrences of letter  $\sigma$  in the array. There is a data structure storing the array with at most*

$$O(|\Sigma| \log n) + \sum_{\sigma \in \Sigma} f_\sigma \log_2 \frac{n}{f_\sigma} + O(n/\log^2 n)$$

*bits of memory. Moreover, there is an algorithm that, upon receiving an index  $i \in [n]$ , queries at most  $O(\log n)$  bits in the data structure and returns the  $i$ -th entry of the array.*

Note that the  $\sum_{\sigma \in \Sigma} f_\sigma \log_2 \frac{n}{f_\sigma}$  term is the entropy of the array, i.e. the information-theoretical lower bound to store the array.

The query process of the data structure can be modeled as a depth- $O(\log n)$  decision tree. The following lemma converts it to an  $\text{NC}^1$  circuit to suit our purpose.

► **Lemma 26.** *Every function that can be computed by a depth- $d$  decision tree can be computed by a depth- $2d$  circuit.*

**Proof.** Prove by induction. When  $d = 1$ , the output then only depends on a single bit and can be trivially computed by a circuit of depth 2.

Suppose the statement holds for  $d$ . Let  $T$  be a depth- $(d+1)$  decision tree. Suppose the root of  $T$  queries  $x_i$  and proceeds to the left or right subtree, depending on whether  $x_i$  is 0 or 1. By our assumption, the two subtrees can be computed by circuits of depth  $2d$ . Let the circuits be  $C_0, C_1$ . Then we construct a circuit  $C$  as

$$C(x) := (x_i \wedge C_1(x)) \vee (\neg x_i \wedge C_0(x)).$$

It is easy to verify that  $C(x)$  computes  $T(x)$  correctly and has depth  $2(d+1)$ , which completes the proof. ◀

The following lemma asserts that the summation of integers is in  $\text{NC}^1$ .

► **Lemma 27** ([22]). *Iteratively adding (i.e., summing up)  $n$   $n$ -bit integers can be done in  $\text{NC}^1$ .*

### 3.2 Rigid Matrices

In this subsection, we reduce constructing rigid matrices to  $\text{NC}^1\text{-AVOID}$ .

► **Theorem 28.** *For any fixed  $\varepsilon, \delta$  such that  $\varepsilon + \delta < \frac{1}{2}$  and any prime power  $q$ ,  $(\varepsilon, \delta, q)\text{-RIGID}$  reduces in polynomial time to  $\text{NC}^1\text{-AVOID}$ .*

**Proof.** WLOG we can assume  $n$  is sufficiently large since for small values of  $n$  we can solve the problem by brute force and reduce to a trivial instance. Let  $M$  be an  $n \times n$  matrix over  $\mathbb{F}_q$  that is not  $(\varepsilon n, \frac{\delta n^2}{\log n})$  rigid. That is,  $M$  can be written as  $X + S$  where  $X$  has rank at most  $\varepsilon n$  and  $S$  has at most  $\frac{\delta n^2}{\log n}$  non-zero entries.  $X$  can be equivalently expressed as the product of an  $n \times \varepsilon n$  matrix  $L$  and  $\varepsilon n \times n$  matrix  $R$ .  $S$  can be encoded by the data structure in Lemma 25. Given this observation, we construct a circuit as follows. We interpret the input bits as the encoding of  $L, R$  and the data structure encoding  $S$ . For the output bit representing  $M_{ij}$ , we first compute  $\sum_{k \in [\varepsilon n]} L_{ik} R_{kj}$ , which can be done by a circuit of  $O(\log n)$ -depth. Then we compute  $S_{ij}$  by making a query to the data structure, which can also be done by a circuit of  $O(\log n)$ -depth by Lemma 26. Finally, we compute  $M_{ij}$  by XOR-ing these two results.

To encode  $L$  and  $R$ , we need  $2\varepsilon n^2 \log q$  bits. One way to encode  $S$  is by storing the indices of the non-zero entries and their values, which requires  $2 \log n + \log q$  bits per non-zero entry. Thus the optimal encoding requires at most  $(2 \log n + \log q) \frac{\delta n^2}{\log n}$  bits, and our data structure needs

$$(2 \log n + \log q) \frac{\delta n^2}{\log n} + 2q \log n + O(n^2 / \log^2 n) < 2\delta n^2 \log q$$

number of bits. Note that the number of output bits is  $n^2 \log q$ . As  $\varepsilon + \delta < \frac{1}{2}$ , the number of input bits is less than the number of output bits. Thus the resulting instance is a valid  $\text{NC}^1\text{-AVOID}$  instance, and any string outside the range of the circuit must be  $(\varepsilon n, \frac{\delta n^2}{\log n})$ -rigid. ◀

By similar techniques, we can reduce constructing binary codes that approach the Gilbert-Varshamov bound and constructing binary codes that achieve list-decoding capacity to  $\text{NC}^1\text{-AVOID}$ .

► **Theorem 29.** *For any  $r, p \in (0, 1)$  such that  $r < 1 - h(p)$ ,  $(r, p)\text{-LINEARCODE}$  reduces in polynomial time to  $\text{NC}^1\text{-AVOID}$ .*

► **Theorem 30.** *For any fixed  $r, p, L$  such that  $r < 1 - h(p) - \frac{2}{\lceil \log_2 L \rceil}$ ,  $(r, p, L)\text{-LISTDECODABLE}$  reduces in polynomial time to  $\text{NC}^1\text{-AVOID}$ .*

We omit the proof here. Readers can refer to our full version for a concrete proof.

### 3.3 Reduction to $\text{NC}_4^0\text{-AVOID}$

In this subsection, we use the reduction by Ren, Santhanam and Wang [21] (i.e., Theorem 1) to further reduce these explicit construction problems to  $\text{NC}_4^0\text{-AVOID}$ . This result shows that solving even  $\text{NC}_4^0\text{-AVOID}$  would have unexpected consequences in pseudorandomness and complexity theory.

Specifically, combining Theorem 1 with our reductions (Theorem 28, 29 and 30), we get the following corollary.

► **Corollary 31.** *Suppose there is a polynomial-time deterministic algorithm for  $\text{NC}_4^0\text{-AVOID}$ . Then the following are true.*

- *For every  $\varepsilon, \delta$  such that  $\varepsilon + \delta < \frac{1}{2}$ , there is a family of  $(\varepsilon n, \frac{\delta n^2}{\log n})$ -rigid matrices that are computable in deterministic polynomial time.*
- *For every rate  $r \in (0, 1)$  and  $p < 1 - h(r)$ , there is a family of  $(r, p)$ -linear code, whose generator matrices are computable in deterministic polynomial time.*
- *For every rate  $r \in (0, 1)$  and parameters  $p \in (0, 1), L \geq 1$  such that  $r < 1 - h(p) - \frac{2}{\lceil \log_2 L \rceil}$ , there is a family of  $(r, p, L)$ -list-decodable code, whose generator matrices are computable in deterministic polynomial time.*

From an algorithmic perspective, Corollary 31 provides a potential approach to attack these notoriously hard explicit construction problems. From a pessimistic viewpoint, Corollary 31 gives further evidence supporting the hardness of solving  $\text{NC}_4^0$  unconditionally.

## 4 A Hitting Set Construction for AVOID

In this section, we use  $\{-1, 1\}$  to represent **True** and **False**, respectively. By Boolean function we mean functions of the form  $f: \{-1, 1\}^n \rightarrow \{-1, 1\}$ .

### 4.1 The General Template

We first show the general framework to construct hitting sets for AVOID instances of weak circuits. We start with a definition of “simple functions”.

► **Definition 32.** *Let  $f: \{-1, 1\}^n \rightarrow \{-1, 1\}$  be a Boolean function. Let  $d \in \mathbb{N}$  and  $\delta > 0$ . We say  $f$  is a  $(d, \delta)$ -simple function, if for any distribution  $\varphi$  over  $\{-1, 1\}^n$ , there is a set  $S \subseteq [n]$  of size at most  $d$  such that the correlation between  $\chi_S$  and  $f$  under  $\varphi$  is at least  $\delta$ . That is,*

$$|\langle f, \chi_S \rangle_\varphi| := \left| \sum_x \varphi(x) f(x) \chi_S(x) \right| \geq \delta.$$

*Suppose  $\mathcal{F} \subseteq \{f: \{-1, 1\}^n \rightarrow \{-1, 1\}\}$  is a collection of functions. We say  $\mathcal{F}$  is a  $(d, \delta)$ -simple collection, if each function in  $\mathcal{F}$  is  $(d, \delta)$ -simple.*

For intuition, it is easy to see that every  $k$ -bit function  $f: \{-1, 1\}^k \rightarrow \{-1, 1\}$  is  $(k, 2^{-k})$ -simple.

**The meta-construction of hitting set.** The following theorem shows our “meta-construction” of hitting set: roughly, for every AVOID-instance  $C: \{-1, 1\}^n \rightarrow \{-1, 1\}^m$ , if the stretch  $m = m(n)$  is sufficiently large (relative to the “simplicity” of the function class), then the support of an almost pairwise independent distribution would be a hitting set for  $\text{AVOID}(C)$ .

► **Theorem 33.** *Suppose  $m > n \geq 2$ . Let  $C: \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  be a circuit and  $\varepsilon > 0$  be a parameter. Suppose each output bit  $C_i$  is a  $(d, \varepsilon)$ -simple function of input bits and  $m > \frac{2}{\varepsilon^2} n^d$ . Let  $\mathcal{D}$  be any  $\frac{3}{8}\varepsilon^2$ -almost pairwise independent distribution over  $\{0, 1\}^m$ . Then, the support of  $\mathcal{D}$  is a hitting set for  $\text{AVOID}(C)$ . That is,  $\text{supp}(\mathcal{D}) \not\subseteq \text{Range}(C)$ .*

Using a standard construction of  $\varepsilon$ -almost pairwise independent distributions (Theorem 24), the support of  $\mathcal{D}$  has size bounded by  $O(\log^2 m / \varepsilon^4)$ . Therefore, by Theorem 33 we can construct a hitting set of size  $O(\log^2 m / \varepsilon^4)$  for  $\text{AVOID}(C)$ . Remarkably, the hitting set is *oblivious* to the circuit  $C$ : one can construct it without actually looking into  $C$ .

**Proof of Theorem 33.** Suppose by contradiction that there exists a pairwise independent distribution  $\mathcal{D}$  such that  $\text{supp}(\mathcal{D}) \subseteq \text{Range}(C)$ . Then, every string  $y \in \text{supp}(\mathcal{D})$  is an output of  $C$ . This implies that under a proper input distribution  $\varphi$  over  $\{-1, 1\}^n$ , the output distribution of  $\{C(x) : x \sim \varphi\}$  is exactly  $\mathcal{D}$ , which is a  $\frac{3}{8}\varepsilon^2$ -almost pairwise independent distribution. In the following, we show that  $C$  *cannot* sample a  $\frac{3}{8}\varepsilon^2$ -almost pairwise independent distribution under *any* input distribution. This would lead to a contradiction and complete the proof.

Let  $\varphi$  be a distribution supported on  $\{-1, 1\}^n$ . Given  $\varphi$ , every output of  $C$  is correlated with  $\chi_S$  for some  $|S| \leq d$  by Definition 32. By pigeonhole principle, there must be  $\frac{m}{2 \cdot n^d} > \frac{1}{\varepsilon^2} =: t$  outputs  $C_1, C_2, \dots, C_t$  that are correlated with the same set  $S$ . By negating the output if necessary, we can assume WLOG that

$$\Pr_{\mathbf{x} \sim \varphi} [C_i(\mathbf{x}) = \chi_S] \geq \frac{1}{2} + \varepsilon, \quad \forall i \in [t].$$

Define

$$Z(\mathbf{x}) = |\#\{i \in [t] : C_i(\mathbf{x}) = 0\} - \#\{i \in [t] : C_i(\mathbf{x}) = 1\}|.$$

We note that

$$\mathbb{E}_{\mathbf{x} \sim \varphi} [Z(\mathbf{x})] \geq \mathbb{E}_{\mathbf{x} \sim \varphi} [|\#\{i \in [t] : C_i(\mathbf{x}) = \chi_S(\mathbf{x})\} - |\#\{i \in [t] : C_i(\mathbf{x}) \neq \chi_S(\mathbf{x})\}|] \geq 2\varepsilon t.$$

Define  $Z_{i,j}(\mathbf{x})$  to be 1 if  $C_i(\mathbf{x}) = C_j(\mathbf{x})$  and  $-1$  otherwise. Then clearly  $Z_{i,i}(\mathbf{x}) = 1$ . Note that  $Z(\mathbf{x})^2 = \sum_{i,j} Z_{i,j}(\mathbf{x})$ , then

$$\mathbb{E}_{\mathbf{x} \sim \varphi} \left[ \sum_{i,j} Z_{i,j}(\mathbf{x}) \right] = \mathbb{E}_{\mathbf{x} \sim \varphi} [Z(\mathbf{x})^2] \geq \mathbb{E}_{\mathbf{x} \sim \varphi} [Z(\mathbf{x})]^2 \geq 4\varepsilon^2 t^2 = 4t.$$

It then follows that

$$\mathbb{E}_{\mathbf{x} \sim \varphi} \left[ \sum_{i \neq j} Z_{i,j}(\mathbf{x}) \right] \geq 3t.$$

Hence, there must be some  $i \neq j$  such that  $\mathbb{E}_{\mathbf{x} \sim \varphi} [Z_{i,j}(\mathbf{x})] \geq \frac{3t}{t(t-1)} > \frac{3\varepsilon^2}{2}$ , meaning that

$$\Pr_{\mathbf{x} \sim \varphi} [C(\mathbf{x})_i = C(\mathbf{x})_j] - \Pr_{\mathbf{x} \sim \varphi} [C(\mathbf{x})_i \neq C(\mathbf{x})_j] = \mathbb{E}_{\mathbf{x} \sim \varphi} [C(\mathbf{x})_i \cdot C(\mathbf{x})_j] > \frac{3\varepsilon^2}{2}.$$

By averaging principle, either  $\Pr_{\mathbf{x} \sim \varphi} [(C(\mathbf{x})_i, C(\mathbf{x})_j) = (1, 1)]$  or  $\Pr_{\mathbf{x} \sim \varphi} [(C(\mathbf{x})_i, C(\mathbf{x})_j) = (-1, -1)]$  is greater than  $\frac{1}{4} + \frac{3\varepsilon^2}{8}$ . This contradicts to the fact that  $C(\varphi)$  samples a  $\frac{3\varepsilon^2}{8}$ -almost pairwise independent distribution.  $\blacktriangleleft$

In the following, we show that for many natural circuit classes ( $\text{NC}_k^0$  for constant  $k$ , constant-width CNF/DNFs, small-size De Morgan formulae, etc.), functions computable in these classes are  $(d, \delta)$ -simple with interesting parameters. Consequently, it allows us to apply Theorem 33 to construct hitting set for AVOID problem of those circuit classes (for large enough stretch).

► **Theorem 5.** *Let  $m = m(n), s = s(n)$  be two non-decreasing functions and  $k, w \geq 1$  be two constants. Suppose  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  is a multi-output function. The following statements hold.*



- If each output bit  $C_i(x)$  depends on only  $k$  input bits and  $m \geq 2^{4k+1}n^{k-1} + n$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $2^{O(k)}m^2$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .
- If each output bit  $C_i(x)$  is a width- $w$  size- $s$  CNF or DNF of input bits and  $m \geq 32s^2n^w$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $O(s^2 \log^2 m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .
- If each output bit  $C_i(x)$  is a size- $s$  De Morgan formula of input bits and  $m \geq n^{\omega(\sqrt{s})}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .
- If each output bit  $C_i(x)$  is a size- $s$  DNF or CNF of input bits and  $m \geq n^{\omega(\sqrt{n \log s})}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .

In all cases, the set  $S$  is independent of the circuit  $C$ . Namely, only knowing  $m, n, s, k, w$  suffices to construct the set  $S$ .

We defer the proof to Appendix A.

---

## References

- 1 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.
- 2 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost  $k$ -wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992. doi:10.1002/rsa.3240030308.
- 3 Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular pcps or: Hard claims have complex proofs. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 858–869. IEEE, 2020. doi:10.1109/FOCS46700.2020.00084.
- 4 Mark Bun and Justin Thaler. The large-error approximate degree of  $\text{ac}^0$ . *Theory Comput.*, 17:1–46, 2021. URL: <https://theoryofcomputing.org/articles/v017a007/>.
- 5 Ronald de Wolf. A note on quantum algorithms and the minimal degree of epsilon-error polynomials for symmetric functions. *CoRR*, 2008. doi:10.48550/ARXIV.0802.1816.
- 6 Peter Elias. List decoding for noisy channels, 1957.
- 7 Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inf. Theory*, 54(1):135–150, 2008.
- 8 Venkatesan Guruswami and Atri Rudra. Better binary list decodable codes via multilevel concatenation. *IEEE Trans. Inf. Theory*, 55(1):19–26, 2009.
- 9 R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.
- 10 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014. doi:10.1137/120897432.
- 11 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.44.
- 12 Adam R. Klivans and Rocco A. Servedio. Learning DNF in time  $2^{\tilde{O}(n^{1/3})}$ . *J. Comput. Syst. Sci.*, 68(2):303–318, 2004. doi:10.1016/j.jcss.2003.07.007.



- 13 Oliver Korten. The hardest explicit construction. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 433–444. IEEE, 2021. doi:10.1109/FOCS52979.2021.00051.
- 14 Xin Lyu. Improved pseudorandom generators for  $AC^0$  circuits. *Electron. Colloquium Comput. Complex.*, TR22-021, 2022. arXiv:TR22-021.
- 15 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in  $nc^0$ . *Random Struct. Algorithms*, 29(1):56–81, 2006. doi:10.1002/rsa.20112.
- 16 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. doi:10.1137/0222053.
- 17 Mihai Patrascu. Succincter. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 305–313. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.83.
- 18 Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.
- 19 Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of  $ac^0$ . *SIAM J. Comput.*, 39(5):1833–1855, 2010. doi:10.1137/080744037.
- 20 Ben Reichardt. Reflections for quantum query algorithms. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 560–569. SIAM, 2011. doi:10.1137/1.9781611973082.44.
- 21 Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. *Electron. Colloquium Comput. Complex.*, 2022. URL: <https://eccc.weizmann.ac.il/report/2022/048>.
- 22 John E. Savage. *The complexity of computing*. Wiley New York, 1976.
- 23 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 238–251, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055408.
- 24 Avishay Tal. Formula lower bounds via the quantum method. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1256–1268. ACM, 2017. doi:10.1145/3055399.3055472.
- 25 Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of  $AC^0$ . In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 242–247. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.32.
- 26 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In Jozef Gruska, editor, *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977. doi:10.1007/3-540-08353-7\_135.
- 27 Emanuele Viola. The complexity of distributions. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 202–211. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.27.
- 28 Emanuele Viola. Sampling lower bounds: Boolean average-case and permutations. *SIAM J. Comput.*, 49(1):119–137, 2020. doi:10.1137/18M1198405.
- 29 Ryan Williams. Nonuniform acc circuit lower bounds. *J. ACM*, 61(1), January 2014. doi:10.1145/2559903.
- 30 John M Wozencraft. List decoding. *Quarterly Progress Report*, 48:90–95, 1958.
- 31 Huacheng Yu. Nearly optimal static las vegas succinct dictionary. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1389–1401. ACM, 2020. doi:10.1145/3357713.3384274.

## A

 Proof of Theorem 5

### A.1 Applications

**$\text{NC}_k^0$  circuits.** Our first application is a hitting set for  $\text{NC}_k^0\text{-AVOID}$  with stretch  $m(n) \geq \omega(n^{k-1})$ .

► **Lemma 34.** *For every  $k \geq 2$  and every  $k$ -bit Boolean function  $f : \{-1, 1\}^k \rightarrow \{-1, 1\}$  that is not  $\chi_{[k]}$  or  $-\chi_{[k]}$ , the following is true. For any distribution  $\varphi : \{-1, 1\}^k \rightarrow [0, 1]$ , there is some  $S \subsetneq [k]$  and some  $z$  and such that*

$$\left| \sum_{x \in \{-1, 1\}^k} \varphi(x) f(x) \text{EQ}_{S, z}(x) \right| \geq 2^{-2k}.$$

**Proof.** Let  $k \geq 2$  and  $\delta = 2^{-2k}$ . Suppose there is a function  $f : \{-1, 1\}^k \rightarrow \{-1, 1\}$  and a distribution  $\varphi$  violating the statement of lemma. We derive a contradiction in the following.

We say that two inputs  $x, y$  are *adjacent* if they only differ at one coordinate. Suppose there are two adjacent  $x, y$  such that they differ at the  $i$ -th coordinate and  $|\varphi(x) - \varphi(y)| \geq \delta$ . We construct  $S = [k] \setminus \{i\}$  and observe that

$$\left| \sum_{z \in \{-1, 1\}^k} \varphi(z) f(z) \text{EQ}_{S, x}(z) \right| = |\varphi(x) f(x) + \varphi(y) f(y)| \geq |\varphi(x) - \varphi(y)| \geq \delta.$$

Since we assumed that  $f$  and  $\varphi$  violate the lemma statement, we have

**Observation 1:**  $|\varphi(x) - \varphi(y)| < \delta$  holds for every adjacent  $x, y \in \{-1, 1\}^k$ .

Next, since  $f$  is not  $\chi_{[k]}$  or  $-\chi_{[k]}$ , there must be adjacent inputs  $x, y$  such that  $f(x) \neq f(y)$ . Suppose they differ at the  $i$ -th coordinate. Let  $S = [k] \setminus \{i\}$ . Similarly, we have

$$\left| \sum_{z \in \{-1, 1\}^k} \varphi(z) f(z) \text{EQ}_{S, x}(z) \right| = |\varphi(x) f(x) + \varphi(y) f(y)| = \varphi(x) + \varphi(y).$$

Having assumed that  $f$  and  $\varphi$  violate the lemma statement, we have

**Observation 2:** There are adjacent  $x, y \in \{-1, 1\}^k$  such that  $\varphi(x) \leq \varphi(y) \leq \delta$ .

Finally, for each  $z \in \{-1, 1\}^k$ , let  $\text{dis}(x, z)$  denote the Hamming distance between  $x$  and  $z$ . By two observations above, we have

$$\sum_{z \in \{-1, 1\}^k} \varphi(z) \leq \sum_{z \in \{-1, 1\}^k} (\text{dis}(x, z) + 1) \cdot \delta \leq k 2^k \delta \leq k 2^{-k} < 1.$$

This contradicts to the fact that  $\varphi$  is a distribution. ◀

► **Lemma 35.** *For every  $k \geq 2$ , every  $k$ -bit Boolean function  $f : \{-1, 1\}^k \rightarrow \{-1, 1\}$  that is not  $\chi_{[k]}$  or  $-\chi_{[k]}$  is  $(k - 1, 2^{-2k})$ -simple.*

**Proof.** By Lemma 34 and Fact 17, there exists some  $S \subsetneq [k]$  and  $z \in \{-1, 1\}^k$  such that

$$\left| \sum_{x \in \{-1, 1\}^k} \varphi(x) f(x) \text{EQ}_{S, z}(x) \right| = \left| \sum_{x \in \{-1, 1\}^k} \varphi(x) f(x) \sum_{T \subseteq S} \frac{1}{2^{|S|}} \left( \prod_{i \in T} z_i \right) \chi_T(x) \right| \geq 2^{-2k}.$$

As  $\prod_{i \in T} z_i$  can only be  $\pm 1$ , by averaging principle, there exists some  $T \subseteq S \subsetneq [k]$  such that

$$|\langle f, \chi_T \rangle_\varphi| = \left| \sum_{x \in \{-1,1\}^k} \varphi(x) f(x) \chi_T(x) \right| \geq 2^{-2k}. \quad \blacktriangleleft$$

► **Corollary 36.** *Let  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  be a  $\text{NC}_k^0$  circuit where  $m > 2^{4k+1}n^{k-1} + n$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $2^{O(k)}m^2$  that is computable in polynomial time and  $S \not\subseteq \text{Range}(C)$ .*

**Proof.** Let  $S$  be the support of a  $(\frac{3}{2} \cdot 2^{-4k})$ -biased distribution over  $\{-1, 1\}^m$ . By Theorem 23, it is of size  $2^{O(k)}m^2$  and can be computed in polynomial time. We consider two cases.

- Suppose there are at least  $n+1$  outputs that are parities of exactly  $k$  input bits. Without loss of generality assume they are  $C_1, \dots, C_{n+1}$ . In this case, we interpret each  $C_i$  as a function  $C_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  by identifying  $-1, 1$  with  $1, 0$  respectively. Then we know for each  $i \in [n+1]$ ,  $C_i$  is an affine function of input bits. Since there are only  $n$  input bits,  $C_1, \dots, C_{n+1}$  are linearly dependent. That is, there is  $\emptyset \neq J \subseteq [n+1]$  such that  $\prod_{i \in J} C_i(x)$  is a constant. On the other hand, as  $S$  is the support of a  $\frac{3}{2} \cdot 2^{-4k}$ -biased distribution, there must be  $y^1, y^2 \in S$  such that

$$\prod_{i \in J} y_i^1 \neq \prod_{i \in J} y_i^2.$$

Therefore, at least one of  $y^1, y^2$  is not in the range of  $C$ .

- It remains to consider the case that there are at least  $m-n$  outputs that are not in the form  $\pm \chi_S$  where  $|S| = k$ . In this case, the correctness follows directly by combining Theorem 33 and Lemma 35. ◀

**Constant-width CNF/DNFs.** Next, we apply our construction to constant-width CNF and DNFs.

► **Lemma 37.** *For every function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  that can be computed by a width- $w$  size- $s$  CNF/DNF, and any distribution  $\varphi : \{-1, 1\}^n \rightarrow [0, 1]$ , there exists a set  $S \subseteq [n]$  and some  $z$  such that  $|S| \leq w$  and*

$$\left| \sum_{x \in \{-1,1\}^n} \varphi(x) f(x) \text{EQ}_{S,z}(x) \right| \geq \frac{1}{4s}.$$

**Proof.** WLOG we assume  $f$  is a DNF. Suppose  $f = \bigvee_{i=1}^s C_i$ , where each  $C_i$  is a logical AND over at most  $w$  literals (i.e., variables or their negations). We can first assume  $\Pr_{\mathbf{x} \sim \varphi}[f(\mathbf{x}) = \text{True}] \in (\frac{1}{4}, \frac{3}{4})$ , or otherwise we can set  $S = \emptyset$  and  $z = 0^n$  such that

$$\left| \sum_{x \in \{-1,1\}^n} \varphi(x) f(x) \text{EQ}_{S,z}(x) \right| = \left| \Pr_{\mathbf{x} \sim \varphi}[f(\mathbf{x}) = \text{False}] - \Pr_{\mathbf{x} \sim \varphi}[f(\mathbf{x}) = \text{True}] \right| > \frac{1}{2} > \frac{1}{4s}.$$

By averaging principle, there exists  $i \in [s]$  such that  $\Pr_{\mathbf{x} \sim \varphi}[C_i(\mathbf{x}) = \text{True}] \geq \frac{1}{4s}$ . Let  $S$  be the variables in  $C_i$  and  $z$  be an arbitrary assignment satisfying  $C_i$ , then we have  $|S| \leq w$  and

$$\left| \sum_{x \in \{-1,1\}^n} \varphi(x) f(x) \text{EQ}_{S,z}(x) \right| \geq \frac{1}{4s}. \quad \blacktriangleleft$$

► **Lemma 38.** *Every function  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  that can be computed by a width- $w$  size- $s$  CNF/DNF is  $(w, \frac{1}{4s})$ -simple.*

**Proof.** By Lemma 37 and Fact 17, there exists some  $S \subseteq [n]$  and  $z \in \{-1, 1\}^n$  such that  $|S| \leq w$  such that

$$\left| \sum_{x \in \{-1, 1\}^n} \varphi(x) f(x) \text{EQ}_{S,z}(x) \right| = \left| \sum_{x \in \{-1, 1\}^n} \varphi(x) f(x) \sum_{T \subseteq S} \frac{1}{2^{|S|}} \left( \prod_{i \in T} z_i \right) \chi_T(x) \right| \geq \frac{1}{4s}.$$

As  $\prod_{i \in T} z_i$  can only be  $\pm 1$ , by averaging principle, there exists some  $T \subseteq S$  such that  $|T| \leq |S| \leq w$  and

$$|\langle f, \chi_T \rangle_\varphi| = \left| \sum_{x \in \{-1, 1\}^n} \varphi(x) f(x) \chi_T(x) \right| \geq \frac{1}{4s}. \quad \blacktriangleleft$$

► **Corollary 39.** *Let  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  be a circuit where  $m > 32s^2n^w$  and each output can be computed by a width- $w$  size- $s$  CNF/DNF, then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $O(s^2 \log^2 m)$  that is computable in polynomial time and  $S \not\subseteq \text{Range}(C)$ .*

**Proof.** Let  $S$  be the support of a  $(\frac{3}{8} \cdot \frac{1}{16s^2})$ -almost pairwise independent distribution. By Theorem 24, it is of size  $O(s^2 \log^2 m)$  and can be computed in polynomial time. The correctness directly follows from Theorem 33 and Lemma 38.  $\blacktriangleleft$

## A.2 Simplicity of Functions from Approximate Degree

In this section, we derive the simplicity of functions (as per Definition 32) by connecting it with (large-error) approximate degree of Boolean functions, a well-studied complexity measure of Boolean functions in literature (see, e.g., [12, 19, 20, 4]).

► **Definition 40.** *Let  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  be a Boolean function. For any  $\varepsilon \in [0, 1)$ , the  $\varepsilon$ -approximate degree of  $f$ , denoted by  $\deg_\varepsilon(f)$ , is the least  $d \in \mathbb{N}$  such that there is a degree- $d$  real polynomial  $p : \mathbb{R}^n \rightarrow \mathbb{R}$  satisfying  $|p(x) - f(x)| \leq \varepsilon$  for every  $x \in \{-1, 1\}^n$ .*

In the literature, when  $\varepsilon$  is not specified, it is typically set as  $\varepsilon = \frac{1}{3}$ . However, for us it is also beneficial to study case that  $\varepsilon$  is very close to 1 (Note that the zero polynomial trivially 1-approximates every Boolean function).

We show that upper bounds for  $\varepsilon$ -approximate degree translate to simplicity of functions.

► **Theorem 41.** *Suppose  $n \geq 10$ . Let  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  be a Boolean function. Let  $\varepsilon \in (0, 1)$ ,  $d \in \mathbb{N}$  be such that  $\deg_{1-\varepsilon}(f) \leq d$ . Then  $f$  is  $(d, \frac{\varepsilon}{3n^{d/2}})$ -simple.*

**Proof.** Let  $p(x) = \sum_{S \subseteq [n], |S| \leq d} \hat{p}(S) \cdot \chi_S(x)$  be a degree- $d$  real polynomial such that  $|f(x) - p(x)| \leq 1 - \varepsilon$  holds for every  $x \in \{-1, 1\}^n$ . By Parseval's identity, we have

$$\sum_{S \subseteq [n]} \hat{p}(S)^2 = \mathbb{E}_{\mathbf{x} \sim \{-1, 1\}^n} [p(\mathbf{x})^2] \leq (1 + 1 - \varepsilon)^2 \leq 4.$$

By Cauchy-Schwarz inequality, we have

$$4 \cdot 2n^d \geq \left( \sum_{S \subseteq [n]} \hat{p}(S)^2 \right) \left( \sum_{S \subseteq [n], |S| \leq d} 1 \right) \geq \left( \sum_{S \subseteq [n], |S| \leq d} |\hat{p}(S)| \right)^2.$$

Therefore,

$$\sum_{S \subseteq [n], |S| \leq d} |\hat{p}(S)| \leq 3n^{d/2}.$$

Note that from  $f(x) \in \{-1, 1\}$  and  $|p(x) - f(x)| \leq 1 - \varepsilon$  we have  $f(x)p(x) \geq \varepsilon$ . Hence, for every distribution  $\varphi$  over  $\{-1, 1\}^n$ , we have

$$\sum_{x \in \{-1, 1\}^n} \varphi(x) \cdot f(x) \cdot p(x) \geq \sum_{x \in \{-1, 1\}^n} \varphi(x) \cdot \varepsilon = \varepsilon.$$

Write  $p(x) = \sum_{S \subseteq [n], |S| \leq d} \hat{p}(S) \cdot \chi_S(x)$ . By averaging principle, there is  $S \subseteq [n], |S| \leq d$  such that

$$\sum_{x \in \{-1, 1\}^n} \varphi(x) \cdot f(x) \cdot \chi_S(x) \geq \frac{\varepsilon}{3n^{d/2}}.$$

Since this argument holds for every distribution  $\varphi$ , we conclude that  $f$  is  $(d, \frac{\varepsilon}{3n^{d/2}})$ -simple. ◀

**Approximate degree of natural circuit classes.** We have the following known upper bounds on approximate degree.

- For  $f$  being a size- $s$  De Morgan formula, following a long line of efforts [20, 24], we now know that  $\deg_{1/3}(f) = O(\sqrt{s})$ . Consequently,  $f$  is  $(O(\sqrt{s}), n^{-O(\sqrt{s})})$ -simple.
- For  $f$  being a CNF/DNF of unbounded width and size  $s$ , it is known that  $\deg_{1-\frac{1}{s}}(f) = O(\sqrt{n \log s})$  [12, 5]. Consequently,  $f$  is  $(\sqrt{n \log s}, n^{-O(n^{1/2} \log s)})$ -simple.

Combining these approximate degree upper bounds with Theorem 33 and 41, as well as the construction from Theorem 24, the following corollary is immediate.

► **Corollary 42.** *Let  $m = m(n), s = s(n)$  be two non-decreasing functions. Suppose  $C : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$  is a multi-output function. The following statements hold.*

- *If each output bit  $C_i(x)$  is a size- $s$  De Morgan formula of input bits and  $m \geq n^{\omega(\sqrt{s})}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*
- *If each output bit  $C_i(x)$  is a size- $s$  DNF or CNF of input bits and  $m \geq n^{\omega(\sqrt{n \log s})}$ , then there is a set  $S \subseteq \{-1, 1\}^m$  of size  $\text{poly}(m)$  that is computable in polynomial time and satisfies  $S \not\subseteq \text{Range}(C)$ .*

*In both cases, the set  $S$  is independent of the circuit  $C$ .*



# Learning Generalized Depth Three Arithmetic Circuits in the Non-Degenerate Case

Vishwas Bhargava ✉🏠

Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Ankit Garg ✉🏠

Microsoft Research, Bangalore, India

Neeraj Kayal ✉🏠

Microsoft Research, Bangalore, India

Chandan Saha ✉🏠

Indian Institute of Science, Bangalore, India

---

## Abstract

Consider a homogeneous degree  $d$  polynomial  $f = T_1 + \dots + T_s$ ,  $T_i = g_i(\ell_{i,1}, \dots, \ell_{i,m})$  where  $g_i$ 's are homogeneous  $m$ -variate degree  $d$  polynomials and  $\ell_{i,j}$ 's are linear polynomials in  $n$  variables. We design a (randomized) learning algorithm that given black-box access to  $f$ , computes black-boxes for the  $T_i$ 's. The running time of the algorithm is  $\text{poly}(n, m, d, s)$  and the algorithm works under some *non-degeneracy* conditions on the linear forms and the  $g_i$ 's, and some additional technical assumptions  $n \geq (md)^2$ ,  $s \leq n^{d/4}$ . The non-degeneracy conditions on  $\ell_{i,j}$ 's constitute non-membership in a variety, and hence are satisfied when the coefficients of  $\ell_{i,j}$ 's are chosen uniformly and randomly from a large enough set. The conditions on  $g_i$ 's are satisfied for random polynomials and also for natural polynomials common in the study of arithmetic complexity like determinant, permanent, elementary symmetric polynomial, iterated matrix multiplication. A particularly appealing algorithmic corollary is the following: Given black-box access to an  $f = \text{Det}_r(L^{(1)}) + \dots + \text{Det}_r(L^{(s)})$ , where  $L^{(k)} = (\ell_{i,j}^{(k)})_{i,j}$  with  $\ell_{i,j}^{(k)}$  being linear forms in  $n$  variables chosen randomly, there is an algorithm which in time  $\text{poly}(n, r)$  outputs matrices  $(M^{(k)})_k$  of linear forms s.t. there exists a permutation  $\pi : [s] \rightarrow [s]$  with  $\text{Det}_r(M^{(k)}) = \text{Det}_r(L^{(\pi(k))})$ .

Our work follows the works [22, 7] which use lower bound methods in arithmetic complexity to design average case learning algorithms. It also vastly generalizes the result in [22] about learning depth three circuits, which is a special case where each  $g_i$  is just a monomial. At the core of our algorithm is the partial derivative method which can be used to prove lower bounds for generalized depth three circuits. To apply the general framework in [22, 7], we need to establish that the non-degeneracy conditions arising out of applying the framework with the partial derivative method are satisfied in the random case. We develop simple but general and powerful tools to establish this, which might be useful in designing average case learning algorithms for other arithmetic circuit models.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Algebraic complexity theory; Computing methodologies  $\rightarrow$  Algebraic algorithms

**Keywords and phrases** Arithmetic Circuits, Average-case Learning, Depth 3 Arithmetic Circuits, Learning Algorithms, Learning Circuits, Circuit Reconstruction

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.21

**Category** RANDOM

**Related Version** Full Version: <https://eccc.weizmann.ac.il/report/2021/155/>

**Funding** Vishwas Bhargava: Research supported in part by the Simons Collaboration on Algorithms and Geometry and NSF grant CCF-1909683.

**Acknowledgements** The authors would like to thank the anonymous referees for useful comments that improved the presentation of the results.



© Vishwas Bhargava, Ankit Garg, Neeraj Kayal, and Chandan Saha; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 21; pp. 21:1–21:22



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Arithmetic circuits are a natural model for computing polynomials using basic arithmetic operations like addition and multiplication. The problem of learning arithmetic circuits a.k.a. reconstruction is an important and well studied problem. It can be defined for various arithmetic circuit models. Unsurprisingly, there is enough evidence to point out that the problem is likely to be hard in the worst case for most arithmetic circuit models [14, 5, 23, 31].<sup>1</sup> Hence, it is imperative to explore algorithms for learning arithmetic circuits that are efficient and work in the average case. One classic example of a stark contrast between the worst case and average case complexities is the tensor decomposition problem. Let us focus on  $n \times n \times n$  tensors for simplicity. In the language of arithmetic complexity, tensor decomposition corresponds to learning depth three set-multilinear circuits. We have three sets of variables  $\mathbf{y} = \{y_1, \dots, y_n\}$ ,  $\mathbf{z} = \{z_1, \dots, z_n\}$ ,  $\mathbf{w} = \{w_1, \dots, w_n\}$ . Then the problem is to decompose a set-multilinear polynomial  $f(\mathbf{y}, \mathbf{z}, \mathbf{w}) = \sum_{j,k,\ell} T_{j,k,\ell} y_j z_k w_\ell$  as

$$\sum_{i=1}^s \ell_{i1}(\mathbf{y}) \ell_{i2}(\mathbf{z}) \ell_{i3}(\mathbf{w})$$

for the smallest possible  $s$  (here  $\ell_{ij}$ 's are linear forms). This is NP-hard in the worst case [14]. However, it is possible to design efficient algorithms for tensor decomposition which work well under some mild assumptions. One such algorithm is due to Jennrich [13, 27] and states that given  $f(\mathbf{y}, \mathbf{z}, \mathbf{w})$  we can find the above decomposition in polynomial time if  $s \leq n$  and  $(\ell_{1a}, \dots, \ell_{sa})$  are linearly independent for all  $a \in [3]$ . Note that the algorithm works under a bound<sup>2</sup> on  $s$  and also a mild assumption on the linear forms (which is satisfied when the linear forms are chosen randomly). Our algorithms will also work under such *non-degeneracy conditions*. Kayal and Saha [22] designed algorithms for learning depth three arithmetic circuits in the non-degenerate case. That is, they design an algorithm for decomposing

$$f(\mathbf{x}) = \sum_{i=1}^s \prod_{j=1}^d \ell_{ij}(\mathbf{x})$$

assuming a bound on  $s$  and certain non-degeneracy conditions on the  $\ell_{ij}$ 's. Note that the above model is different from tensor decomposition or set-multilinear circuits since there is no partitioning of variables into disjoint sets and the linear forms can depend on all the variables. We prove a far-reaching generalization of the result of [22].

### 1.1 The model and our results

We study the model of generalized depth three circuits. A circuit in this class computing a degree  $d$  polynomial  $f(\mathbf{x})$  is an expression of the following kind,

$$f(\mathbf{x}) = g_1(\ell_{11}, \dots, \ell_{1m}) + \dots + g_s(\ell_{s1}, \dots, \ell_{sm}),$$

where  $g_i$ 's are  $m$ -variate degree  $d$  homogeneous polynomials and  $\ell_{ij}$ 's are linear forms in the variables  $\mathbf{x} = (x_1, \dots, x_n)$ . Our main result is an algorithm for learning decompositions of the above kind assuming certain non-degeneracy conditions.

<sup>1</sup> Despite this, there has been much success in designing worst case reconstruction algorithms. This includes reconstruction algorithms for the models of sparse polynomials [25], read-once algebraic branching programs (ROABPs) [1, 24] and for models with bounded top fan-in [23, 17, 11, 32, 2, 3].

<sup>2</sup> This bound usually corresponds to the best known lower bounds we can prove for the corresponding model.



► **Theorem 1** (Learning generalized depth three circuits in the *non-degenerate* case). *There is a randomized algorithm that given black-box access to an  $n$ -variate degree  $d$  polynomial  $f = T_1 + \dots + T_s$ , where  $T_i = g_i(\ell_{i1}, \dots, \ell_{im})$  for a homogeneous  $m$ -variate polynomial  $g_i$ , outputs black-boxes for the individual summands  $T_i$ 's. The running time of the algorithm is  $\text{poly}(n, m, d, s)$ . The algorithm works under certain non-degeneracy conditions and also under some additional technical assumptions such as  $n \geq (md)^2$ ,  $s \leq n^{d/4}$ ,  $|\mathbb{F}| \geq \text{poly}(n^d, s)$  and  $\text{char}(\mathbb{F}) = 0$  or  $\text{char}(\mathbb{F}) > d$ .*

The non-degeneracy conditions are mentioned explicitly in Section 2.1. These non-degeneracy conditions are satisfied when the coefficients of the linear forms are chosen uniformly and independently at random from a large enough set and when the  $g_i$ 's are either random or one of the well-known polynomials in arithmetic complexity such as determinant, permanent, elementary symmetric polynomial etc. Let us mention one such appealing corollary which follows from Theorem 1 and the algorithms for equivalence-testing of the determinant [19, 6].

► **Corollary 2** (Learning sums of random projections of determinants). *Suppose  $n, r, \mathbb{F}, s$  be such that  $n \geq r^6$ ,  $s \leq n^{r/4}$ ,  $|\mathbb{F}| \geq \text{poly}(n^r, s)$  and  $\text{char}(\mathbb{F}) = 0$  or  $\text{char}(\mathbb{F}) > r$ . There is a randomized  $\text{poly}(n, r)$  time algorithm that given black-box access to an  $f = \text{Det}_r(L^{(1)}) + \dots + \text{Det}_r(L^{(s)})$ , where  $L^{(k)} = (\ell_{i,j}^{(k)})_{i,j}$  with  $\ell_{i,j}^{(k)}$ 's being linear forms in  $n$  variables whose coefficients are chosen independently and uniformly at random from an arbitrary set  $S \subset \mathbb{F}$  of size  $|S| \geq \text{poly}(n^r, s)$ , it outputs matrices of linear forms  $(M^{(k)})_k$  s.t. there exists a permutation  $\pi : [s] \rightarrow [s]$  with  $\text{Det}_r(M^{(k)}) = \text{Det}_r(L^{(\pi(k))})$ .*

## Remarks

1. Once we have the black-boxes for the  $T_i$ 's as in Theorem 1, it is not hard to output black-boxes for  $\tilde{g}_i$ 's and also  $\tilde{\ell}_{i1}, \dots, \tilde{\ell}_{im}$  s.t.  $T_i = \tilde{g}_i(\tilde{\ell}_{i1}, \dots, \tilde{\ell}_{im})$ . This is done by finding an invertible linear transformation on  $g_i$  that restrict it to its “essential variables”, see [18, Thm 4.1]. Note that we cannot hope to exactly recover the  $g_i$ 's since there is some redundancy. One can always apply a linear transformation to the input variables of  $g_i$ 's to obtain different decompositions.
2. We get a similar result (as in Corollary 2) with  $g_i$ 's being the elementary symmetric polynomial, permanent, iterated matrix multiplication, monomials etc. Note that it could be a mixture of these. It might seem strange that we are able to handle permanent, but note that we are only dealing with black-boxes and hence the complexity of the permanent does not come into play. It is already known how to do equivalence-testing of the permanent efficiently [19] which is similar in spirit to the  $s = 1$  case.
3. The field size and the size of the set  $S$  in Theorem 1 and Corollary 2 depends exponentially on the degree. This does not affect the runtime since one can do arithmetic in exponentially large fields in polynomial time. It is possible to get a polynomial dependence on the degree. We have not elaborated on this to preserve simplicity of analysis but we provide a sketch of an argument to reduce the field size in Appendix C.

## 1.2 Techniques and proof overview

We follow the meta framework of [22, 7] for designing learning algorithms for arithmetic circuits in the non-degenerate case via lower bounds. We note that while the meta framework is quite general, still a lot of technical work is needed to carry it out for a particular circuit class if one has lower bounds for that class. The same holds for this paper. We will not go into the full generality of the framework and refer the reader to the exposition in [7]. Instead, we will explain the details for our special case.

Let us first see how one would prove a lower bound (on the number of summands  $s$ ) for the model of generalized depth three circuits. Consider the set of all partial differential operators of order  $k$  i.e.  $\mathcal{L} = \partial_{\mathbf{x}}^{\mathbf{k}}$ . These are linear maps from  $\mathbb{F}[\mathbf{x}]_d$  to  $\mathbb{F}[\mathbf{x}]_{d-k}$ , where  $\mathbb{F}[\mathbf{x}]_t$  denote the ring of homogeneous degree  $t$  polynomials in  $\mathbb{F}[\mathbf{x}]$ . Note that

$$\dim(\langle \mathcal{L} \circ T_i \rangle) \leq \binom{m+k-1}{k},$$

if  $T_i$  is of the form  $g_i(\ell_{i1}, \dots, \ell_{im})$ . This is easy to verify if  $T_i$  were equal to  $g_i(x_1, \dots, x_m)$ . Then one can use the fact that the dimension of the partial derivative space doesn't change upon an invertible linear transformation of the variables. Also note that

$$\langle \mathcal{L} \circ f \rangle \subseteq \langle \mathcal{L} \circ T_1 \rangle + \dots + \langle \mathcal{L} \circ T_s \rangle \quad (1)$$

$$\dim(\langle \mathcal{L} \circ f \rangle) \leq \sum_{i=1}^s \dim(\langle \mathcal{L} \circ T_i \rangle) \leq s \binom{m+k-1}{k}.$$

It is not too hard to design an  $f$  for which  $\dim(\langle \mathcal{L} \circ f \rangle) \approx \binom{n+k-1}{k}$  (when  $k \leq \lfloor d/2 \rfloor$ ) and for such an  $f$  we get a lower bound  $\approx (n/m)^k$ . We can choose  $k = \lfloor d/2 \rfloor$  to get the highest lower bound.

It is natural to wonder what is the connection to learning, if there is any at all. Consider Equation 1. One can hope that in the generic case, one would get

$$\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \dots \oplus \langle \mathcal{L} \circ T_s \rangle. \quad (2)$$

That is the inclusion becomes an equality and the sum becomes a direct sum. Furthermore, let us assume that it holds for  $\mathcal{L}' = \partial_{\mathbf{x}}^{\mathbf{k}+1}$  as well. That is,

$$\langle \mathcal{L}' \circ f \rangle = \langle \mathcal{L}' \circ T_1 \rangle \oplus \dots \oplus \langle \mathcal{L}' \circ T_s \rangle. \quad (3)$$

So we have  $U := \langle \mathcal{L} \circ f \rangle$ ,  $V := \langle \mathcal{L}' \circ f \rangle$  and the linear maps  $\partial_{\mathbf{x}}^{-1}$  from  $U$  to  $V$ . Let  $U_i := \langle \mathcal{L} \circ T_i \rangle$  and  $V_i := \langle \mathcal{L}' \circ T_i \rangle$ . Note that the linear maps  $\partial_{\mathbf{x}}^{-1}$  map  $U_i$  into  $V_i$ . So one is naturally led towards the following *vector decomposition problem*.

► **Problem 3 (Vector space decomposition).** *Given the triple  $(\mathcal{M}, U, V)$  consisting of vector spaces  $U$  and  $V$  and a set of linear maps  $\mathcal{M}$  from  $U$  to  $V$ , decompose  $U$  and  $V$  as*

$$U = U_1 \oplus \dots \oplus U_s \quad V = V_1 \oplus \dots \oplus V_s$$

*s.t.  $\langle \mathcal{M} \circ U_i \rangle \subseteq V_i$  for all  $i \in [s]$ .*

For our setting, one such decomposition is described in Equations (2) and (3). Once one has access to  $U_i$ 's (black-box access to a basis), it is not hard to obtain black-boxes for the  $T_i$ 's. So the only thing remains to prove is the uniqueness of vector space decomposition (in addition to (2) and (3) themselves). There are many efficient algorithms to solve the vector space decomposition problem. Please refer to Appendix A for specialized algorithms that work for our setting, and [7] for a thorough discussion on the general problem and related work. Let us now describe our approaches to prove Equations (2) and (3) and also the uniqueness of decomposition.

For proving uniqueness of decomposition, we employ the use of the notion of an adjoint algebra, following [7]. The adjoint algebra essentially captures “homomorphisms” of the triple  $(\mathcal{M}, U, V)$ . That is,

$$\text{Adj}(\mathcal{M}, U, V) = \{(D, E) : D : U \rightarrow U, E : V \rightarrow V \text{ linear maps and } LD = EL \forall L \in \mathcal{M}\}$$

Suppose the triple  $(\mathcal{M}, U, V)$  admits a vector space decomposition  $U = U_1 \oplus \cdots \oplus U_s$ ,  $V = V_1 \oplus \cdots \oplus V_s$ . Then the projection maps  $(\Pi_{U_i}, \Pi_{V_i})$  (which are identity on  $U_i, V_i$  respectively and map other vector spaces in the direct sum to 0) lie in the adjoint. We say that the adjoint algebra is *trivial* if it is spanned by these projectors. It is not hard to show that if the adjoint algebra is trivial, then the above vector space decomposition is unique (Lemma 34). Note that one can always combine blocks in an arbitrary way, but the decomposition is unique among all “finest” decompositions where one cannot decompose any block further. So we are left with proving the uniqueness of the decomposition in Equations (2) and (3). We prove that the adjoint algebra is trivial in this case (proof of Theorem 5) using a non-degeneracy condition on the  $g_i$ ’s (Item 3 in Section 2.1; also see Section 3.3).

So now let us see how to prove Equations (2) and (3). Showing the direct sum  $U_1 + \cdots + U_s = U_1 \oplus \cdots \oplus U_s$  (and the same for  $V_i$ ’s) is done in a similar way to [22], Schwartz-Zippel lemma yields the direct sum once one can show the existence of some set of linear forms satisfying the direct sum property. This is done using a design construction based on Nisan-Wigderson designs. This construction is inspired from [22] but more general. We differ significantly from previous works [22, 7] in our technique for showing that  $U = U_1 + \cdots + U_s$ . The previous works relied on intricate design constructions to exhibit linear forms which satisfy this property (followed by a use of Schwartz-Zippel lemma). For our setting, one can get away with the above design based approach, but this can become more cumbersome and challenging as the circuit models become more complicated. Hence, we devise a general way of proving statements of the form

$$\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle + \cdots + \langle \mathcal{L} \circ T_s \rangle$$

for  $f = T_1 + \cdots + T_s$ , which is conceptually more appealing. It is useful to have the linear maps  $\mathcal{L}$  from a subspace of the operators (so for our case think of  $\mathcal{L} = \langle \partial_{\mathbf{x}}^{\leq k} \rangle$ ). Since

$$\langle \mathcal{L} \circ f \rangle \subseteq \langle \mathcal{L} \circ T_1 \rangle + \cdots + \langle \mathcal{L} \circ T_s \rangle,$$

it suffices to prove that  $\langle \mathcal{L} \circ T_i \rangle \subseteq \langle \mathcal{L} \circ f \rangle$  for all  $i$ . Let us consider the operators annihilating a particular term  $T_i$ .

$$\mathcal{L}_i^{\text{null}} := \{L \in \mathcal{L} : L \circ T_i = 0\}$$

Now note that for any  $L \in \cap_{j \neq i} \mathcal{L}_j^{\text{null}}$ ,  $L \circ f = L \circ T_i$ . If the subspace of operators  $\cap_{j \neq i} \mathcal{L}_j^{\text{null}}$  was rich enough, at least to the extent relevant to  $T_i$ , then we would be done. We are able to show this by moving to the duals of the vector spaces  $\mathcal{L}_i^{\text{null}}$  (with respect to an appropriate bilinear form) and proving a direct sum property there (the proof of which turns out to be almost identical to the proof we have for the direct sum of the  $U_i$ ’s!). For more details, see Section 2.

**Comparison with previous works.** Our work closely follows the papers [22, 7] on learning arithmetic circuits in the non-degenerate case via lower bounds. However, there are substantial differences as well. Firstly, as explained above, we devise a general technique for proving statements of the kind  $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle + \cdots + \langle \mathcal{L} \circ T_s \rangle$ . Secondly, ours is the first paper that uses the full machinery of the learning from lower bounds framework in [22, 7]. In [22], the framework was present in a rudimentary form and that made the analysis more cumbersome. While the framework was fully laid out in [7], for their application of learning sums of powers of low degree polynomials, they eventually implement a somewhat ad hoc approach. Without this learning framework, it seems rather challenging to get such a general result as in Theorem 1.

### 1.3 Related work

[9] proved a lower bound for the more general model of generalized depth-four circuits (bounded bottom-fanin). [17] study the worst case learning algorithms for a model which is similar to our model in many ways, but their parameters are different (they also call their model generalized depth three circuits). There has been a lot of work on *worst case* reconstruction algorithms which includes reconstruction algorithms for the models of sparse polynomials [25], read-once algebraic branching programs (ROABPs) [1, 24] and for models with bounded top fan-in [23, 17, 11, 32, 2, 3].

In [10], a randomized polynomial-time proper learning algorithm was given for *non-degenerate*<sup>3</sup> multilinear formulas having fan-in two. A randomized polynomial-time proper learning algorithm for non-degenerate regular formulas having fan-in two was given in [12]. An efficient randomized reconstruction for non-degenerate homogeneous ABPs of width at most  $\frac{\sqrt{n}}{2}$  is presented in [20]. [22] designed algorithms for learning non-degenerate depth three circuits which is a special case of our model with the  $g_i$ 's being a monomial. [7], following [22], developed a meta framework for learning non-degenerate arithmetic circuits via lower bounds. They implemented it to learn sums of powers of low degree polynomials in the non-degenerate case.

As already mentioned, the problem of tensor decomposition is a special case for our model. Tensor decomposition is widely studied in the machine learning community as well (also known as CP decomposition), e.g. see the surveys [26, 4, 15]. Another kind of tensor decomposition, Tucker decomposition is also widely studied, see Section 4 in [26]. Tensor decomposition roughly corresponds to the  $m = 1$  case in our model<sup>4</sup> Tucker decomposition roughly corresponds to  $s = 1$  in our model.<sup>5</sup> Given the wide variety of applications of these two problems in machine learning, we hope that (noise-resilient versions of) our algorithms will handle much more challenging problems in machine learning.

### 1.4 Roadmap of the paper

In Section 2, we present our algorithm for learning non-degenerate generalized depth three circuits, the corresponding non-degeneracy conditions and the analysis of the algorithm assuming the non-degeneracy conditions. In Section 3, we prove that the non-degeneracy conditions are satisfied for random circuits. Section 4 contains the summary of the work and some of the open problems that arise from this work. Section A contains some basic facts about the vector space decomposition problem. Finally, Section B contains some facts about how to perform linear algebra given black boxes.

## 2 The learning algorithm and its analysis

In this section, we describe our algorithm for learning non-degenerate generalized depth three circuits and the analysis assuming the non-degeneracy conditions. Since we are aiming for  $\text{poly}(s)$  time-complexity, we can assume that we know  $s$ . For a field  $\mathbb{F}$  and  $d \in \mathbb{N}$ , let  $\mathbb{F}[\mathbf{x}]_d$  denote the ring of homogeneous degree  $d$  polynomials in  $\mathbb{F}[\mathbf{x}]$ . Consider a homogeneous

<sup>3</sup> The papers [10, 12] state the results for random formulas, but it is not difficult to state the non-degeneracy conditions by taking a closer look at the algorithms.

<sup>4</sup> Strictly speaking  $m = 1$  would be symmetric tensor decomposition and exactly modeling general tensor decomposition would require higher  $m$  but in spirit tensor decomposition is closer to the  $m = 1$  case than higher  $m$ .

<sup>5</sup> Again, ignoring some symmetry considerations here.

degree  $d$  polynomial  $f \in \mathbb{F}[\mathbf{x}]_d$  which is computed by a homogeneous generalized depth three circuit i.e.,  $f = T_1 + \dots + T_s$ , where  $T_i = g_i(\ell_{i1}, \dots, \ell_{im})$  for  $i \in [s]$ . Here  $\ell_{ij}$ 's are linear forms.

## 2.1 Non-degeneracy conditions

We impose the following non-degeneracy conditions on  $f$  (or more precisely the circuit computing it):

1. For each  $i \in [s]$ , the linear forms  $(\ell_{i1}, \dots, \ell_{im})$  are linearly independent. Also the vector spaces  $W_1^{(d-k)} := \mathbb{F}[\ell_{11}, \dots, \ell_{1m}]_{d-k}, \dots, W_s^{(d-k)} := \mathbb{F}[\ell_{s1}, \dots, \ell_{sm}]_{d-k}$  form a direct sum i.e.

$$W_1^{(d-k)} + \dots + W_s^{(d-k)} = W_1^{(d-k)} \oplus \dots \oplus W_s^{(d-k)}.$$

The same assumption for the vector spaces  $W_i^{(d-k-1)}$ 's.

2. We will use  $\partial =^k$  to denote the set of order- $k$  partial differential operators in the variables  $\mathbf{x}$ . Consider the vector spaces  $U := \langle \partial =^k f \rangle$ ,  $V := \langle \partial =^{(k+1)} f \rangle$ ,  $U_i := \langle \partial =^k T_i \rangle$ ,  $V_i = \langle \partial =^{(k+1)} T_i \rangle$ . We will assume that

$$U = U_1 \oplus \dots \oplus U_s$$

and

$$V = V_1 \oplus \dots \oplus V_s.$$

3. For the polynomials  $g_i \in \mathbb{F}[\mathbf{z}]_d$ ,  $\mathbf{z} = (z_1, \dots, z_m)$ , the triple  $(\partial_{\mathbf{z}} =^1, \langle \partial_{\mathbf{z}} =^k g_i \rangle, \langle \partial_{\mathbf{z}} =^{(k+1)} g_i \rangle)$  has a trivial adjoint algebra for all  $i \in [s]$  (see Definitions 30 and 32). That is, if  $D : \langle \partial_{\mathbf{z}} =^k g_i \rangle \rightarrow \langle \partial_{\mathbf{z}} =^k g_i \rangle$  and  $E : \langle \partial_{\mathbf{z}} =^{(k+1)} g_i \rangle \rightarrow \langle \partial_{\mathbf{z}} =^{(k+1)} g_i \rangle$  are linear maps s.t.  $\partial_{z_j} D(p) = E(\partial_{z_j} p)$  for all  $j \in [m]$  and all  $p \in \langle \partial_{\mathbf{z}} =^k g_i \rangle$ , then  $D, E$  are both identity maps (up to a scalar multiplication). Note that Corollary 39 implies that this condition is preserved if we apply an invertible linear transformation to the  $\mathbf{z}$  variables.

The algorithm is stated in Algorithm 1. We will need the following lemma in the proof of the main theorem.

► **Lemma 4.** *Let  $h \in \mathbb{F}[\mathbf{x}]_d$  be a homogeneous degree  $d$  polynomial and  $\ell_1, \dots, \ell_m \in \mathbb{F}[\mathbf{x}]_1$  be linearly independent linear forms. Then  $h \in \mathbb{F}[\ell_1, \dots, \ell_m]_d$  iff  $\sum_{j=1}^n \alpha_j \partial_{x_j} h(\mathbf{x}) = 0$  for all  $\alpha \in \mathbb{F}^n$  s.t.  $\ell_i(\alpha) = 0$  for all  $i \in [m]$ .*

**Proof.** Let  $\ell_i = \sum_{j=1}^n \ell_{ij} x_j$ . In one direction, suppose  $h \in \mathbb{F}[\ell_1, \dots, \ell_m]_d$  so that  $h = g(\ell_1, \dots, \ell_m)$  for  $g \in \mathbb{F}[\mathbf{z}]$ ,  $\mathbf{z} = (z_1, \dots, z_m)$ . Then

$$\begin{aligned} \sum_{j=1}^n \alpha_j \partial_{x_j} h(\mathbf{x}) &= \sum_{j=1}^n \alpha_j \sum_{i=1}^m \ell_{ij} \partial_{z_i} g(\mathbf{z})|_{\mathbf{z}=(\ell_1, \dots, \ell_m)} \\ &= \sum_{i=1}^m \ell_i(\alpha) \partial_{z_i} g(\mathbf{z})|_{\mathbf{z}=(\ell_1, \dots, \ell_m)} \\ &= 0 \end{aligned}$$

for all  $\alpha \in \mathbb{F}^n$  s.t.  $\ell_i(\alpha) = 0$  for all  $i \in [m]$ . In the other direction, suppose  $\sum_{j=1}^n \alpha_j \partial_{x_j} h(\mathbf{x}) = 0$  for all  $\alpha \in \mathbb{F}^n$  s.t.  $\ell_i(\alpha) = 0$  for all  $i \in [m]$ . Extend  $\ell_1, \dots, \ell_m$  to a full basis of  $\mathbb{F}[\mathbf{x}]_1$ ,  $\ell_1, \dots, \ell_n$  (in an arbitrary way). We can write  $h$  as  $g(\ell_1, \dots, \ell_n)$  for some  $g \in \mathbb{F}[\mathbf{w}]$ ,  $\mathbf{w} = (w_1, \dots, w_n)$ . Our goal now is to prove that  $\partial_{w_i} g(\mathbf{w}) = 0$  for all  $i \in \{m+1, \dots, n\}$ . Now

$$\begin{aligned} \sum_{j=1}^n \alpha_j \partial_{x_j} h(\mathbf{x}) &= \sum_{j=1}^n \alpha_j \sum_{i=1}^n \ell_{ij} \partial_{w_i} g(\mathbf{w})|_{\mathbf{w}=(\ell_1, \dots, \ell_n)} \\ &= \sum_{i=1}^n \ell_i(\alpha) \partial_{w_i} g(\mathbf{w})|_{\mathbf{w}=(\ell_1, \dots, \ell_n)}. \end{aligned}$$

For  $i \in \{m+1, \dots, n\}$ , we can choose an  $\alpha$  s.t.  $\ell_j(\alpha) = 0$  for all  $j \neq i$  and  $\ell_i(\alpha) \neq 0$ . Then from the assumption and the above calculation we get that  $\partial_{w_i} g(\mathbf{w})|_{\mathbf{w}=(\ell_1, \dots, \ell_n)} = 0$ . Since  $\ell_1, \dots, \ell_n$  are linearly independent, we get that  $\partial_{w_i} g(\mathbf{w}) = 0$  for all  $i \in \{m+1, \dots, n\}$ . ◀

■ **Algorithm 1** Learning generalized depth three circuits.

**Input:** black-box access to an  $f \in \mathbb{F}[\mathbf{x}]_d$  that is computed by a non-degenerate homogeneous generalized depth three circuit i.e.,  $f = T_1 + \dots + T_s$ , where  $T_i = g_i(\ell_{i1}, \dots, \ell_{im})$  for  $i \in [s]$ .

**Output:**  $s$  black-boxes  $\mathcal{B}_1, \dots, \mathcal{B}_s$  such that there exists a permutation  $\pi : [s] \rightarrow [s]$  s.t.  $\mathcal{B}_i$  provides black-box access to  $T_{\pi(i)}$ .

**Subroutines:**

1. Computing black-boxes for partial derivatives from the black-box for a polynomial. (Fact 29)
2. Vector space decomposition (Algorithm 2 and Corollary 41).

**Parameters:** The order of partial derivatives:  $k$ .

1. Compute black-boxes for a basis of the vector spaces  $U := \langle \partial =^k f \rangle$  and  $V := \langle \partial =^{(k+1)} f \rangle$  using Subroutine 1.
2. Using Subroutine 2, obtain a vector space decomposition  $U = U'_1 \oplus \dots \oplus U'_{s'}$  and  $V = V'_1 \oplus \dots \oplus V'_{s'}$  for the triple  $(\partial =^1, U, V)$ . If  $s' \neq s$ , then abort. Otherwise continue.
3. For each  $\alpha$  s.t.  $\sum_{i=1}^n \alpha_i = k$ , write the corresponding differential operator acting on  $f$ ,  $\partial_{\alpha} f$ , as  $u'_{\alpha 1} + \dots + u'_{\alpha s}$  with  $u'_{\alpha i} \in U'_i$  (note that there is a unique such representation). We only obtain black-boxes for the polynomials  $u'_{\alpha i}$ 's. This step can be carried out using Corollary 41.
4. The black-box  $\mathcal{B}_i$  on input  $\mathbf{x}$  will output  $\frac{(d-k)!}{d!} \sum_{\alpha} \binom{k}{\alpha_1, \dots, \alpha_n} \mathbf{x}^{\alpha} u'_{\alpha i}(\mathbf{x})$ .

The next theorem states the correctness of Algorithm 1 assuming the non-degeneracy conditions.

► **Theorem 5.** *Suppose the non-degeneracy conditions stated above are satisfied. Then Algorithm 1 never aborts. Suppose  $\mathcal{B}_1, \dots, \mathcal{B}_s$  be the black-boxes output by the algorithm. Then there exists a permutation  $\pi : [s] \rightarrow [s]$  s.t.  $\mathcal{B}_i$  is a black-box for  $T_{\pi(i)}$ .*

**Proof.** It suffices to prove uniqueness of decomposition for the triple  $(\partial =^1, U, V)$  (see Definition 33). Assuming uniqueness of decomposition,  $s' = s$  and there exists a permutation  $\pi : [s] \rightarrow [s]$  s.t.  $U'_i = U_{\pi(i)}$  and  $V'_i = V_{\pi(i)}$ . Since the  $U'_i$ 's form a direct sum, there is a unique way of writing each element  $u \in U$  as  $u = u'_1 + \dots + u'_s$  with  $u'_i \in U'_i$ . For  $u = \partial_{\alpha} f$ ,  $u = \partial_{\alpha} T_{\pi(1)} + \dots + \partial_{\alpha} T_{\pi(s)}$  is one such decomposition and hence the only one. Thus  $u'_{\alpha i} = \partial_{\alpha} T_{\pi(i)}$  in which case  $\mathcal{B}_i$  computes the black-box for  $T_{\pi(i)}$  by Lagrange's formula,

$$h(\mathbf{x}) = \frac{(d-k)!}{d!} \sum_{\alpha} \binom{k}{\alpha_1, \dots, \alpha_n} \mathbf{x}^{\alpha} \partial_{\alpha} h(\mathbf{x})$$

for a homogeneous degree  $d$  polynomial  $h$ .

To prove uniqueness of decomposition, it suffices to prove that the adjoint algebra for the triple  $(\partial=1, U, V)$  is trivial because of Lemma 34. Consider linear maps  $D : U \rightarrow U$  and  $E : V \rightarrow V$  s.t.  $\partial_{x_j} D(u) = E(\partial_{x_j} u)$  for all  $u \in U$ . Then we need to prove that  $D(U_i) \subseteq U_i$ ,  $E(V_i) \subseteq V_i$  for all  $i \in [s]$  and that  $(D, E)$  are scalar multiples of identity when restricted to  $(U_i, V_i)$  respectively. The latter follows from Item 3 in the non-degeneracy conditions, so we only need to prove the former. To prove the former, consider  $(D, E)$  in the adjoint algebra. Note that  $U_i \subseteq \mathbb{F}[\ell_{i1}, \dots, \ell_{im}]_{d-k}$  and  $V_i \subseteq \mathbb{F}[\ell_{i1}, \dots, \ell_{im}]_{d-k-1}$ . Hence if  $u \in U_i$ , then

$$\sum_{j=1}^n \alpha_j \partial_{x_j} u(\mathbf{x}) = 0$$

for all  $\alpha$  s.t.  $\ell_{i1}(\alpha) = \dots = \ell_{im}(\alpha) = 0$ , by Lemma 4. Because of the relation  $\partial_{x_j} D(u) = E(\partial_{x_j} u)$ , we get that

$$\sum_{j=1}^n \alpha_j \partial_{x_j} D(u)(\mathbf{x}) = 0$$

for all  $\alpha$  s.t.  $\ell_{i1}(\alpha) = \dots = \ell_{im}(\alpha) = 0$ . Hence by Lemma 4, we get that  $D(u) \in \mathbb{F}[\ell_{i1}, \dots, \ell_{im}]_{d-k}$ . Hence  $D(u) \in U \cap \mathbb{F}[\ell_{i1}, \dots, \ell_{im}]_{d-k} = U_i$  (because of the direct sum structure of the vector spaces  $\mathbb{F}[\ell_{i1}, \dots, \ell_{im}]_{d-k}$  in Item 1). This completes the proof that  $D(U_i) \subseteq U_i$ . Now the space  $V_i$  has a basis which consists of a subset of polynomials from  $\partial_{\beta} T_i$  as  $\beta$  varies over monomials of degree  $k+1$ . We can write  $\partial_{\beta} T_i$  as  $\partial_{x_j} \partial_{\alpha} T_i$  for some  $j \in [n]$  and some  $\alpha$  of degree  $k$ . Then

$$E(\partial_{\beta} T_i) = E(\partial_{x_j} \partial_{\alpha} T_i) = \partial_{x_j} D(\partial_{\alpha} T_i).$$

Since  $D(\partial_{\alpha} T_i) \in U_i$ , we get that  $E(\partial_{\beta} T_i) \in V_i$ . This completes the proof that  $E(V_i) \subseteq V_i$ . ◀

We will now proceed to proving Theorem 1.

**Proof of Theorem 1.** We will run Algorithm 1 on the given black-box with the parameter  $k$  being set to  $\lceil \frac{2 \log s}{\log n} \rceil$ . Notice that, by Fact 29, the time complexity of subroutine 1 is  $\text{poly}(d^k, n) = \text{poly}(s, n)$ . See Remark 6. Since Theorem 5 guarantees the correctness of our output, we just have to verify its running time. Note that the time complexity of remaining steps is  $\text{poly}(n^k, s) = \text{poly}(n, s)$ , which concludes the proof. ◀

### 3 Non-degeneracy of random circuits

In this section we will show that if  $n > (md)^2$ ,  $s \leq n^{d/4}$  and  $k = \lceil \frac{2 \log s}{\log n} \rceil$ , then a *random*  $(n, d, s, \{g_i\}_{i \in [s]})$  homogeneous generalized depth three circuit is non-degenerate with high probability. To better understand the regime of parameters, we record a few relations among the parameters  $n, d, s$  and  $k$  in the following easy to verify remark.

► **Remark 6.** If  $s \leq n^{d/4}$ ,  $md \leq \sqrt{n}$  and  $k = \lceil \frac{2 \log s}{\log n} \rceil$  then  $k \leq d/2$  and  $\binom{m+k-1}{k} \leq ns$ .

We will proceed by showing that each of our non-degeneracy conditions is satisfied for random circuits, and then the result will hold directly by the union bound. We also show that  $(n, d, s, \{g_i\}_{i \in [s]})$  homogeneous generalized depth three circuits are non-degenerate if the  $g_i$ 's belong to special polynomial families like  $\text{Det}_d, \text{Perm}_d, \text{IMM}_{r,d}, \text{Sym}_{r,d}$  and only  $\ell_{i,j}$ 's are chosen randomly. This is because non-degeneracy condition 1 and 2 just depend<sup>6</sup> on  $\ell_{i,j}$ 's

<sup>6</sup> Strictly speaking, non-degeneracy condition 2 does depend on  $g_i$ 's, but we show that it holds if we just pick  $\ell_{i,j}$ 's randomly. See Lemma 16



and non-degeneracy condition 3 depends on the  $g_i$ 's, and we can show that aforementioned polynomial families satisfy these mild technical conditions required to show non-degeneracy condition 3.

### 3.1 Non-degeneracy of random circuits: Condition 1

Let's begin by restating our first non-degeneracy condition for a generalized depth three circuit  $\sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$ .

**Non-degeneracy condition 1.** The vector spaces  $W_1^{(d-k)} := \mathbb{F}[\ell_{11}, \dots, \ell_{1m}]_{d-k}, \dots, W_s^{(d-k)} := \mathbb{F}[\ell_{s1}, \dots, \ell_{sm}]_{d-k}$  form a direct sum i.e.

$$W_1^{(d-k)} + \dots + W_s^{(d-k)} = W_1^{(d-k)} \oplus \dots \oplus W_s^{(d-k)}.$$

And, the same assumption for the vector spaces  $W_i^{(d-k-1)}, \dots, W_s^{(d-k-1)}$ .

We will show that if  $m \leq \frac{\sqrt{n}}{t}$  and  $s \leq n^{t/2}$  then a random choice of  $\{\ell_{i,j}\}_{(i,j) \in ([s],[m])}$  satisfies the equality  $\sum_{i=1}^s W_i^{(t)} = \oplus W_i^{(t)}$ . To show this we will need the notion of *combinatorial designs*.

► **Definition 7** (Nisan-Wigderson designs [29]). A family of sets  $\mathcal{A} = \{A_1, \dots, A_s\}$  is said to be an  $(n, m, d)$  design if  $A_i \subseteq [n]$  with  $|A_i| = m$  for all  $i \in [s]$ . And, for  $i \neq j$ ,  $|A_i \cap A_j| < d$ .

We will be using a standard construction of such designs based on the Reed-Solomon codes.

► **Lemma 8** (Explicit Design). Let  $m \leq \sqrt{n}$ . There exists an  $(n, m, d)$ -design  $\{A_1, \dots, A_s\}$  for  $s \leq m^d$ .

► **Lemma 9.** Let  $S \subseteq \mathbb{F}$  be a finite set. If  $m \leq \frac{\sqrt{n}}{t}$  and  $s \leq n^{t/2}$  then for a random choice of  $\{\ell_{i,j}\}_{(i,j) \in ([s],[m])}$  linear forms over  $S$ ,

$$\sum_{i=1}^s W_i^{(t)} = \oplus_{i \in [s]} W_i^{(t)}$$

with probability at least  $1 - \frac{s \cdot \binom{m+t-1}{t} \cdot t}{|S|}$ .

For proof of the above lemma see the full version.

As a direct consequence of Lemma 9 for  $t = d - k - 1$  and  $t = d - k$  we get that the non-degeneracy condition 1 holds with high probability.

► **Corollary 10.** If  $(md)^2 \leq n$ ,  $k = \lceil 2 \frac{\log s}{\log n} \rceil$ ,  $s \leq n^{d/4}$  and  $|S| \geq \text{poly}(n^d)$  then for a random choice of  $\{\ell_{i,j}\}_{(i,j) \in ([s],[m])}$  linear forms over a set  $S$ ,  $\sum_{i=1}^s W_i^{(d-k)} = \oplus_{i \in [s]} W_i^{(d-k)}$  and  $\sum_{i=1}^s W_i^{(d-k-1)} = \oplus_{i \in [s]} W_i^{(d-k-1)}$  with probability  $1 - o(1)$ .

### 3.2 Non-degeneracy of random circuits: Condition 2

Our next non-degeneracy condition for  $f(\mathbf{x}) = \sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$  requires that the vector spaces  $U := \langle \partial = {}^k f \rangle$  and  $V := \langle \partial = {}^{(k+1)} f \rangle$  have a direct sum structure. That is,

$$U = U_1 \oplus \dots \oplus U_s \text{ and } V = V_1 \oplus \dots \oplus V_s, \quad (4)$$



where  $U_i := \langle \partial^{=k} T_i \rangle$  and  $V_i := \langle \partial^{=(k+1)} T_i \rangle$  where  $T_i := g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$ . Note that as  $U_i \subseteq W_i^{(d-k)}$ , the direct sum structure of  $W_i^{(d-k)}$  directly gives  $U_1 + U_2 + \dots + U_s = U_1 \oplus U_2 \oplus \dots \oplus U_s$ . Indeed for the regime of parameters we are interested in,  $W_i^{(d-k)}$  do have a direct sum structure for random  $\ell_{i,j}$ 's by Lemma 9. Thus in order to show non-degeneracy condition 2 for random circuits, it suffices to show

$$U = U_1 + U_2 + \dots + U_s. \quad (5)$$

Clearly,  $U \subseteq U_1 + U_2 + \dots + U_s$ . To show the other direction, it suffices to show that  $U \supseteq U_i$  for all  $i \in [s]$ . We show this via a novel technique of studying the space of partial derivative operators (i.e.  $\langle \partial^{=k} \rangle$ ) themselves, as opposed to the space when they are applied to a polynomial (i.e.  $\langle \partial^{=k} f \rangle$ ). Interestingly, our proof is very general and works for action of any general linear operators on a space! Thus, we state and prove it in full generality and later instantiate the setting needed for our work.

We start by elaborating on our abstract setting. Let  $f = T_1 + T_2 + \dots + T_s$  where  $T_i \in \mathcal{C}_i$  and  $\mathcal{L}$  is a vector space of linear operators from  $\mathbb{F}[\mathbf{x}]$  to  $W$ . Here,  $\mathcal{C}_i$  is a circuit class consisting of polynomials in  $\mathbb{F}[x]$ . Also, let  $\mathcal{B} : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{F}$  be a non-degenerate bilinear form, that is for any non-zero  $u \in \mathcal{L}$  there exists a  $v \in \mathcal{L}$  s.t.  $\mathcal{B}(u, v) \neq 0$ . Furthermore, let  $\mathcal{L}_i^\perp := \{L \in \mathcal{L} \mid Lh = 0, \forall h \in \mathcal{C}_i\}$ . Using the bilinear product  $\mathcal{B}$  and any subspace  $U$  of  $\mathcal{L}$ , we define  $U^{\perp \mathcal{B}}$  as the linear operators (in  $\mathcal{L}$ ) s.t. for all  $u \in U$  the bilinear product is 0. Formally,  $U^{\perp \mathcal{B}} := \{L \in \mathcal{L} \mid \forall u \in U, \mathcal{B}(L, u) = 0\}$ .

Our next lemma shows that under a direct sum structure of  $\sum_{i \in [s]} (\mathcal{L}_i^\perp)^{\perp \mathcal{B}}$ ,  $\mathcal{L}(f) = \sum_{i \in [s]} \mathcal{L}(T_i)$ .

► **Lemma 11.** *Let  $\mathcal{L}, \mathcal{B}, f(\mathbf{x})$ , and  $T_i$ 's be as defined above. If  $\sum_{i \in [s]} (\mathcal{L}_i^\perp)^{\perp \mathcal{B}} = \oplus_{i \in [s]} (\mathcal{L}_i^\perp)^{\perp \mathcal{B}}$  then  $\mathcal{L}(f) = \sum_i \mathcal{L}(T_i)$ .*

For proof of the above lemma see the full version.

For our application of showing  $U \supset U_i$ , we set  $\mathcal{L} = \langle \partial^{=k} \rangle$ ,  $\mathcal{C}_i$  is the class of polynomials  $g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$  where  $g_i$  is an  $m$ -variate degree  $d$  polynomial and  $\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m}$  are random  $n$ -variate linear forms. Also,  $\mathcal{L}_i^\perp = \mathcal{D}_i^\perp := \{D \in \langle \partial^{=k} \rangle \mid Dh = 0, \forall h \in W_i^{(d)}\}$ . Note that in order to apply Lemma 11 in our setting, we have to come up with a non-degenerate bilinear map  $\mathcal{B}$ , s.t.  $\sum_i (\mathcal{D}_i^\perp)^{\perp \mathcal{B}} = \oplus_i (\mathcal{D}_i^\perp)^{\perp \mathcal{B}}$ . Let's first note that if  $(\mathcal{D}_i^\perp)^{\perp \mathcal{B}}$  does satisfy the direct sum property then we are done! Indeed, on setting,  $\mathcal{L} = \langle \partial^{=k} \rangle$  and  $\mathcal{L}_i^\perp = \mathcal{D}_i^\perp$  to  $\mathcal{L}(f) = \sum_i \mathcal{L}(T_i)$  gives  $\langle \partial^{=k}(f) \rangle = \sum_i \langle \partial^{=k} \rangle(T_i)$ , thus implying (5).

In the rest of the section, we will focus on coming up with a bilinear form and showing that it is indeed non-degenerate. And later, via another application of Lemma 9, show the direct sum structure of  $(\mathcal{D}_i^\perp)^{\perp \mathcal{B}}$ . For two polynomials  $f$  and  $g$ , define

$$\mathcal{B}(f, g) := f\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \dots, \frac{\partial}{\partial x_n}\right) \cdot g.$$

This inner product among two polynomials is known as apolar inner product, and is a fundamental notion with a lot of applications, see [30] and the references therein. It is easy to see that  $\mathcal{B}(\ell_1, \ell_2) = v_{\ell_1} \cdot v_{\ell_2}$ ; where  $\ell_1(\mathbf{x}), \ell_2(\mathbf{x})$  are two linear forms,  $v_{\ell_1}, v_{\ell_2}$  are canonical vectors associated with them and  $v_{\ell_1} \cdot v_{\ell_2}$  is the standard dot product among vectors. The non-degenerate bilinear map needed for our purpose acts on  $\mathcal{L} \times \mathcal{L}$  instead of polynomials as defined above. But in our case  $\mathcal{L} = \langle \partial^{=k} \rangle$  is nothing but polynomials of degree  $k$  with  $\frac{\partial}{\partial x_i}$  as variables, thus the definition of  $\mathcal{B}$  extends naturally.

In order to show that our bilinear map is non-degenerate, it will be convenient to work with an orthogonal basis of  $\ell_{i,j}$ 's. We will therefore need the following lemma.

► **Lemma 12.** *When  $\text{char}(\mathbb{F}) \neq 2$ , there exists an orthogonal basis of any finite dimensional vector space over  $\mathbb{F}$  with respect to any non-degenerate bilinear (dot) product.*

For proof of the above lemma see the full version.

Let  $V$  be the space of some linear forms in  $\mathbb{F}[\mathbf{x}]$ , then by the above lemma one can assume that there exist an orthogonal basis of  $V$  as long as  $\text{char}(\mathbb{F}) \neq 2$ . Now, we will state an observation on what is  $\mathcal{B}(f, \cdot)$  when  $f$  and  $g$  are expressed as polynomials in an orthogonal basis.

► **Observation 13.** *If  $\ell_1(\mathbf{x}), \dots, \ell_n(\mathbf{x})$  is an orthogonal basis of  $\mathbb{F}[\mathbf{x}]_1$  then if  $g = \sum_{\alpha} c_{\alpha} \ell^{\alpha}$  and  $f = \sum_{\alpha} d_{\alpha} \ell^{\alpha}$  are degree  $d$  polynomials. Then*

$$\mathcal{B}(f, g) = \sum_{\alpha} c_{\alpha} \cdot d_{\alpha} \alpha!.$$

Here  $\alpha! = \prod_{i \in [n]} \alpha_i!$  and  $\alpha$  as an index varies over exponent vector of  $n$ -variate monomials of degree exactly  $d$ .

The above observation follows directly by observing it when  $g$  is a monomial and extending by linearity. Now, if  $\text{char}(\mathbb{F}) > d$  or 0, then using this observation we directly get that  $\mathcal{B}(f, g)$  is non-degenerate.

► **Lemma 14.** *The bi-linear map  $\mathcal{B}(f, g)$  over  $\mathbb{F}[\mathbf{x}]_d$  is non-degenerate if  $\text{char}(\mathbb{F}) > d$  or 0. That is for all non-zero  $g \in \mathbb{F}[\mathbf{x}]_d$  there exist  $f \in \mathbb{F}[\mathbf{x}]_d$  s.t.  $\mathcal{B}(f, g) \neq 0$ .*

**Proof.** Let  $\ell_1(\mathbf{x}), \dots, \ell_n(\mathbf{x})$  be an orthogonal basis of  $\mathbb{F}[\mathbf{x}]_1$ . Now, for any  $g = \sum_{\alpha} c_{\alpha} \ell^{\alpha} \neq 0$ , let  $\alpha_o$  be an exponent vector s.t.  $c_{\alpha_o} \neq 0$ . On choosing  $f = \ell^{\alpha_o}$  we get  $\mathcal{B}(f, g) = c_{\alpha_o} \alpha_o! \neq 0$  if  $\text{char}(\mathbb{F}) > d$  or 0. ◀

### 3.2.1 Direct sum structure of $(\mathcal{D}_i^{\perp})^{\perp \mathcal{B}}$

The only thing left to show non-degeneracy condition 2 is a direct sum structure on derivative operators  $(\mathcal{D}_i^{\perp})^{\perp \mathcal{B}}$ . We will first study the space  $\mathcal{D}_i^{\perp}$ , as it will help us show the required direct sum structure. Let's assume (WLOG) that  $\ell_{i,1}, \dots, \ell_{i,m}, q_{i,1}, \dots, q_{i,n-m}$  is an orthogonal basis of  $\mathbb{F}^n$  w.r.t.  $\mathcal{B}$ . That is, for  $i \neq j$   $\langle \ell_{i,1}, \ell_{1,j} \rangle = 0$ ,  $\langle \ell_{1,i}, q_{1,j} \rangle = 0$  and  $\langle \ell_{1,i}, \ell_{1,i} \rangle \neq 0$ . Notice that,

$$\mathcal{D}_i^{\perp} \supseteq q_{i,1} \cdot \langle \partial = (k-1) \rangle + q_{i,2} \cdot \langle \partial = (k-1) \rangle + \dots + q_{i,n-m} \cdot \langle \partial = (k-1) \rangle. \quad (6)$$

▷ **Claim 15.** Let  $\text{char}(\mathbb{F}) > d$  or  $\text{char}(\mathbb{F}) = 0$ , then  $W_i^{(k)} := \mathbb{F}[\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m}]_k \supseteq (\mathcal{D}_i^{\perp})^{\perp \mathcal{B}}$ .

**Proof.** For brevity we will denote the space  $(q_{i,1} \cdot \langle \partial = (k-1) \rangle + q_{i,2} \cdot \langle \partial = (k-1) \rangle + \dots + q_{i,n-m} \cdot \langle \partial = (k-1) \rangle)$  by  $Q$ . We have that  $\mathcal{D}_i^{\perp} \supseteq Q$ , thus  $(\mathcal{D}_i^{\perp})^{\perp \mathcal{B}} \subseteq Q^{\perp \mathcal{B}}$ . The proof concludes by showing that  $Q^{\perp \mathcal{B}} = W_i^{(k)}$ . Clearly,  $Q^{\perp \mathcal{B}} \supseteq W_i^{(k)}$ . For the other direction, let  $p \in Q^{\perp \mathcal{B}}$  s.t.  $p \notin W_i^{(k)}$ . We can write,  $p = w + q$  where  $w \in W_i^{(k)}$  and  $q \in Q$  s.t.  $q \neq 0$ . Now, since  $p \in Q^{\perp \mathcal{B}}$ ,  $\mathcal{B}(p, q') = 0 \forall q' \in Q$ . Notice that for any  $q' \in Q$ ,  $\mathcal{B}(w, q') = 0$ . Thus,  $\mathcal{B}(p, q') = \mathcal{B}(q + w, q') = \mathcal{B}(q, q')$ . Now, just like in the proof of Lemma 14 we can choose  $q' \in Q$  s.t.  $\mathcal{B}(q, q') \neq 0$ . That is, pick  $q'$  to be any monomial in  $\mathbb{F}[\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m}, q_{i,1}, \dots, q_{i,n-m}]_d$  with non-zero coefficient in  $q$  and by observation 13 we get that  $\mathcal{B}(q, q') \neq 0$ . This implies  $p \in Q$  and  $\mathcal{B}(p, q') \neq 0$   $p(\bar{\partial}) \cdot p(\bar{x}) \neq 0$ , thus contradicting  $p \in Q^{\perp \mathcal{B}}$ . ◀

► **Lemma 16.** For homogeneous degree  $d$  polynomials  $\{g_i\}_{i \in [s]}$ , let  $f = \sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m}) \in \mathbb{F}[\mathbf{x}]_d$  and  $U := \langle \partial^=^k f \rangle$ ,  $V := \langle \partial^=^{(k+1)} f \rangle$ ,  $U_i := \langle \partial^=^k T_i \rangle$ ,  $V_i := \langle \partial^=^{(k+1)} T_i \rangle$ . If  $\text{char}(\mathbb{F}) > d$  or  $\text{char}(\mathbb{F}) = 0, (md)^2 \leq n, k = \lceil 2 \frac{\log s}{\log n} \rceil$  and  $s \leq n^{d/4}$  then for a random choice of  $\{\ell_{i,j}\}_{(i,j) \in ([s],[m])}$  linear forms over a set  $S \subset \mathbb{F}$  such that  $|S| \geq \text{poly}(n^d)$ ,  $U = U_1 \oplus \dots \oplus U_s$  and  $V = V_1 \oplus \dots \oplus V_s$  with probability at least  $1 - o(1)$ .

**Proof.** By Lemma 9 we get that for a finite set  $S \subseteq \mathbb{F}$ , if  $m \leq \frac{\sqrt{n}}{d}$  and  $s \leq n^{d/4}$  then for a random choice of  $\{\ell_{i,j}\}_{(i,j) \in ([s],[m])}$  linear forms over  $S$ ,  $\sum_{i=1}^s W_i^{(k)} = \oplus W_i^{(k)}$  with probability at least  $1 - o(1)$ . Now, since  $W_i^{(k)}$  has a direct sum structure, the same will hold for their respective subspaces. Thus,  $\sum_{i \in [s]} (\mathcal{D}_i^\perp)^{\perp_S} = \oplus_{i \in [s]} (\mathcal{D}_i^\perp)^{\perp_S}$ . Now, using Lemma 11 with  $\mathcal{L} = \langle \partial^=^k \rangle$  and  $\mathcal{L}_i^\perp = \mathcal{D}_i^\perp := \{D \in \langle \partial^=^k \rangle \mid Dh = 0, \forall h \in W_i^{(d)}\}$  implies  $U = U_1 + U_2 + \dots + U_s$ . And, as  $U_i \subseteq W_i^{(d-k)}$ , the direct sum structure of  $W_i^{(d-k)}$  directly gives  $U_1 + U_2 + \dots + U_s = U_1 \oplus U_2 \oplus \dots \oplus U_s$ . Notice that,  $W_i^{(d-k)}$  have direct sum structure by corollary 10 as  $m \leq \frac{\sqrt{n}}{d}$  and  $s \leq n^{d/4}$ . The proof for  $V = V_1 \oplus \dots \oplus V_s$  is identical. ◀

### 3.3 Non-degeneracy condition 3: Adjoint algebra is trivial

We will start by restating non-degeneracy condition 3.

**Non-degeneracy condition 3.** For a generalized depth 3 circuit  $f = \sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$  where  $g_i \in \mathbb{F}[\mathbf{z}]_d$ ,  $\mathbf{z} = (z_1, \dots, z_m)$ , the triple  $(\partial_{\mathbf{z}}^=^1, \langle \partial_{\mathbf{z}}^=^k g_i \rangle, \langle \partial_{\mathbf{z}}^=^{(k+1)} g_i \rangle)$  has a trivial adjoint algebra for all  $i \in [s]$ . That is, if  $D : \langle \partial_{\mathbf{z}}^=^k g_i \rangle \rightarrow \langle \partial_{\mathbf{z}}^=^k g_i \rangle$  and  $E : \langle \partial_{\mathbf{z}}^=^{(k+1)} g_i \rangle \rightarrow \langle \partial_{\mathbf{z}}^=^{(k+1)} g_i \rangle$  are linear maps s.t.  $\partial_{z_j} D(p) = E(\partial_{z_j} p)$  for all  $p \in \langle \partial_{\mathbf{z}}^=^k g_i \rangle$ , then  $D, E$  are both identity maps (up to a scalar multiple).

We will show this for random  $g_i$ 's as well as various interesting polynomial families like monomials, determinant, permanent, elementary symmetric polynomial and iterated matrix multiplication. This is done by observing that under mild technical conditions on  $g$ , the triple  $(\partial_{\mathbf{z}}^=^1, \langle \partial_{\mathbf{z}}^=^k g \rangle, \langle \partial_{\mathbf{z}}^=^{(k+1)} g \rangle)$  has a trivial adjoint algebra. Define  $\{\partial_{\mathbf{z}}^=^k g\} := \{\partial_m^=^k g \mid \forall m \text{ degree } k \text{ monomials in } \mathbb{F}[\mathbf{z}]\}$ . And, let  $\text{var}(f)$  denote the set of variables  $f$  depends on. We start by stating our technical condition:

► **Technical condition 17.** Let  $g \in \mathbb{F}[\mathbf{z}]_d$ , we need  $\text{var}(p) \neq \text{var}(p')$  for any non-zero (and distinct)  $p, p' \in \{\partial_{\mathbf{z}}^=^k g\}$ . And all non-zero elements of  $\{\partial_{\mathbf{z}}^=^{k+1} g\}$  to be  $\mathbb{F}$ -linearly independent.

► **Lemma 18.** Let  $g \in \mathbb{F}[\mathbf{z}]_d$  that satisfies condition 17 and  $D : \langle \partial_{\mathbf{z}}^=^k g \rangle \rightarrow \langle \partial_{\mathbf{z}}^=^k g \rangle$  and  $E : \langle \partial_{\mathbf{z}}^=^{(k+1)} g \rangle \rightarrow \langle \partial_{\mathbf{z}}^=^{(k+1)} g \rangle$  be two linear maps. If  $\forall j \in [m]$  and all  $p \in \langle \partial_{\mathbf{z}}^=^k g \rangle$ ,  $\partial_{z_j} D(p) = E(\partial_{z_j} p)$ , then  $D(p) = c_p \cdot p$  for all  $p \in \{\partial_{\mathbf{z}}^=^k g\}$ , where  $c_p \in \mathbb{F}$  which could depend on  $p$ .

For proof of the above lemma see the full version.

Note that, the above lemma *doesn't* imply that the adjoint algebra is trivial, as  $c_{p_i}$  could possibly depend on  $g_i$ . To show that the adjoint algebra is trivial, we need to prove that  $c_{p_i}$  is the same constant for all  $p_i$ 's. In order to do that we will need the following notion of graph associated with a polynomial.

► **Definition 19.** For a polynomial  $g \in \mathbb{F}[\mathbf{z}]_d$ , let  $G_g^k$  be the graph whose vertices are degree  $k$  multilinear monomials  $m$  s.t.  $\partial_m^=^k g \neq 0$  and the edge set consist of pairs of monomials  $(m_1, m_2)$  with  $\Delta(m_1, m_2) = 2$  and  $\partial_{\text{lcm}(m_1, m_2)}^=^{k+1} g \neq 0$ . Here  $\Delta(m_1, m_2)$  is the hamming distance among the exponent vectors of  $m_1$  and  $m_2$ .

► **Lemma 20.** *Let  $g \in \mathbb{F}[\mathbf{z}]_d$  be a polynomial s.t. it satisfies condition 17. Additionally, if  $G_g^k$  is connected, then the triple  $(\partial_{\mathbf{z}}=1, \langle \partial_{\mathbf{z}}=^k g \rangle, \langle \partial_{\mathbf{z}}=^{(k+1)} g \rangle)$  has a trivial adjoint algebra.*

For proof of the above lemma see the full version.

It is an easy exercise to see that for a *random* multilinear  $g$  condition 17 is satisfied and  $G_g^k$  is connected. We will now show the same holds for various polynomial families which includes monomials, determinant, permanent, elementary symmetric polynomial and iterated matrix multiplication. The argument for showing connectivity of  $G_g^k$  stems from this simple observation.

► **Observation 21.** *If  $m_1$  and  $m_2$  are degree  $k$  monomials with  $\Delta(m_1, m_2) = \delta$  and let  $m_1 = m^{(0)}, m^{(1)}, \dots, m^{(\delta)} = m_2$  be a path made of distance two monomials (i.e.  $\Delta(m^{(i-1)}, m^{(i)}) = 2$  for  $i \in [\delta]$ ) from  $m_1$  to  $m_2$  s.t.  $\partial_{\tilde{m}^{(i)}} g \neq 0$  ( $\tilde{m}^{(i)} := \text{lcm}(m^{(i)}, m^{(i+1)})$ ) then  $m_1$  and  $m_2$  are connected.*

► **Lemma 22.** *If  $g$  is one of the following polynomials  $\text{Det}_d$ ,  $\text{Perm}_d$  (with  $3k \leq d$ );  $\text{Sym}_{r,d}$ , monomial (with  $k+1 < d$ ) or  $\text{IMM}_{r,d}$  (with  $3k \leq d$ ) then  $G_g^k$  is connected and condition 17 is satisfied.*

For proof of the above lemma, see the full version.

### 3.4 Adjoint algebra for random $g_i$ 's

We will now show that the adjoint algebra is trivial for *random*  $g_i$ 's. This is done by showing that the adjoint algebra is trivial if the space spanned by  $k$ -th order partial derivatives applied to  $g$  have full dimension. Followed by observing that random  $g_i$ 's have this property.

► **Lemma 23.** *Let  $g \in \mathbb{F}[\mathbf{z}]_d$  be a polynomial such that  $\dim(\langle \partial_{\mathbf{z}}=^k g \rangle) = \binom{k+m-1}{k}$ . Also, let  $U_g = \langle \partial_{\mathbf{z}}=^k g \rangle$ ,  $V_g = \langle \partial_{\mathbf{z}}=^{k+1} g \rangle$  and  $D : U_g \rightarrow U_g$  and  $E : V_g \rightarrow V_g$  be any linear maps. If for all  $j \in [m]$  and  $p \in U_g$ ,  $\partial_{z_j} D(p) = E(\partial_{z_j} p)$ , then  $D$  and  $E$  are identity maps up to a scalar multiple. That is, the triple  $(\partial_{\mathbf{z}}=1, \langle \partial_{\mathbf{z}}=^k g \rangle, \langle \partial_{\mathbf{z}}=^{(k+1)} g \rangle)$  has a trivial adjoint algebra.*

For proof of the above lemma see the full version.

We can instantiate the above lemma for *random*  $g_i$ 's. Indeed, the condition  $\dim(\partial_{\mathbf{z}}=^k g) = \binom{k+m-1}{k}$  boils down to showing that the determinant of a matrix with dimension  $\binom{k+m-1}{k}$  is non-zero. And, there exist standard constructions of explicit polynomials with this property (see [8]). Thus, via the Schwartz-Zippel lemma, we get the following corollary.

► **Corollary 24.** *For a random choice of degree  $d$  homogeneous polynomials  $\{g_i\}_{i \in [s]}$  over a set  $S$ , the triple  $(\partial_{\mathbf{z}}=1, \langle \partial_{\mathbf{z}}=^k g_i \rangle, \langle \partial_{\mathbf{z}}=^{(k+1)} g_i \rangle)$  has a trivial adjoint algebra for all  $i \in [s]$ , with probability at least  $1 - \frac{sd \cdot \binom{m+k-1}{k}}{|S|}$ .*

Now, we can combine corollary 10, 24 and Lemma 22, 16 to show that a random generalized depth 3 circuit is non-degenerate with high probability.

► **Lemma 25** (Random generalized depth 3 circuits are non-degenerate). *Let  $\mathcal{C} \equiv \sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$  be a homogeneous generalized depth 3 circuit, where  $\{g_i\}_{i \in [s]}$  are homogeneous degree  $d$  polynomials, and  $n \geq (md)^2$ . Suppose the coefficients of  $\ell_{i,j}$ 's are chosen uniformly and independently at random from a set  $S \subset \mathbb{F}$  of size  $|S| \geq \text{poly}(n^d, s)$ . Additionally, suppose one of the following cases is true:*

1.  $g_i$ 's belong to one of the polynomial families:  $\text{Det}_d, \text{Perm}_d, \text{IMM}_{r,d}, \text{Sym}_{r,d}$ , monomials with  $s \leq n^{d/6}$ .
2. Coefficients of all  $g_i$ 's are chosen uniformly and independently at random from  $S$  and  $s \leq n^{d/4}$ .

Then, with probability  $1 - o(1)$ ,  $\mathcal{C}$  is non-degenerate.

The proof is immediate using union bound along with Remark 6 and hence omitted. As a direct consequence of Lemma 25 and Theorem 5, we get the following theorem about learning random generalized depth circuits.

► **Theorem 26** (Learning random generalized depth 3 circuits). *Let  $\mathcal{C} \equiv \sum_{i=1}^s g_i(\ell_{i,1}, \ell_{i,2}, \dots, \ell_{i,m})$  be a homogeneous generalized depth 3 circuit, where  $\{g_i\}_{i \in [s]}$  are homogeneous degree  $d$  polynomials, and  $n \geq (md)^2$ . Suppose the coefficients of  $\ell_{i,j}$ 's are chosen uniformly and independently at random from a set  $S \subset \mathbb{F}$  of size  $|S| \geq \text{poly}(n^d, s)$  and  $\text{char}(\mathbb{F}) > d$  or  $\text{char}(\mathbb{F}) = 0$ . Additionally, suppose one of the following cases is true:*

1.  $g_i$ 's belong to one of the polynomial families:  $\text{Det}_d, \text{Perm}_d, \text{IMM}_{r,d}, \text{Sym}_{r,d}$ , monomials with  $s \leq n^{d/6}$ .
2. Coefficients of all  $g_i$ 's are chosen uniformly and independently at random from  $S$  and  $s \leq n^{d/4}$ .

Then, given black-box access to  $\mathcal{C}$  we can reconstruct it in randomized  $\text{poly}(n, m, d, s)$  time.

Note that, the  $m$  subsumes the dependence on  $r$  in the above theorem. Also, the reconstruction algorithm of Theorem 26 is proper, i.e. it outputs a homogeneous generalized depth 3 circuit.

### 3.5 From black-box access to learning generalized depth three circuits

Theorem 5 gives a black-box for each  $g_i$ 's under the technical conditions discussed. It is natural to ask if we can find  $\ell_{i,j}$ 's and a generalized depth 3 representation as well. This is related to the well studied equivalence-testing problem, specifically to the search version of it. The equivalence-testing question is the following: given polynomials  $f$  and  $g$ , find an invertible linear transformation  $A$  on variables such that  $f = g(A\mathbf{x})$ , if such  $A$  exists. Observe that if  $g_i$  belongs to a family for which we can solve the equivalence-testing problem, then we can find  $\ell_{i,j}$ 's as well. This follows directly by seeing each blackbox as an instance of equivalence-testing (search version). Note that in our non-degenerate setting,  $\ell_{i,j}$ 's are linearly independent for each  $i \in [s]$  thus satisfies the requirement that the linear transformation has to be invertible. As a direct consequence of this we get the following corollary.

► **Corollary 27.** *Suppose we are given black-box access to  $f$ , an  $n$ -variate, homogeneous degree  $d$  polynomial computable by a generalized depth 3 circuit of size  $s$ , s.t. the non-degeneracy condition 1, 2 and 3 hold. Additionally, if each  $g_i$  belongs to a family of polynomials for which there exist a  $\text{poly}(n, m, d)$  time equivalence-testing algorithm. Then there exist a  $\text{poly}(s, n, d, m)$  time algorithm that learns a generalized depth 3 representation of  $f$ .*

Note that if  $g_i$  is just a monomial (the special case for depth 3 circuits) then equivalence-testing follows directly from black-box factoring [16]. Hence, when  $g_i$ 's are monomials the previous corollary along with Lemma 22 (monomials satisfy non-degeneracy condition 3) gives an algorithm for learning non-degenerate homogenous depth three circuits! Thus, our result is truly a generalization of the result by [22].

In general, equivalence-testing is considered to be a very hard problem (see [19, 18]) but it has been solved in several interesting cases, we list some of them below. For ease of representation, let us define some notation representing the complexity of the search version of

the polynomial equivalence problem over a particular field. Given  $m, d, r \in \mathbb{N}$  and black-box access to an  $m$ -variate polynomial  $g$  of degree  $d$ , let  $\text{Eqv}_{\mathbb{F}}(r, d, m, f)$  denote the randomized time complexity of finding an invertible linear transformation  $A$  s.t.  $g(\mathbf{x}) = f(A\mathbf{x})$  if it exists, otherwise output “no-solution”.

► **Theorem 28.** *Following results are known for equivalence-testing of special families of polynomials:*

1.  $\text{Eqv}_{\mathbb{F}}(r, d, m, \text{Sym}_{r,d}) = \text{poly}(r, d, m)$ , if  $\text{char}(\mathbb{F}) > d$  or 0. See [19].
2.  $\text{Eqv}_{\mathbb{F}}(r, d, m, \text{Perm}_r) = \text{poly}(r, d, m)$ . See [19].
3.  $\text{Eqv}_{\mathbb{F}}(r, d, m, \text{Det}_r) = \text{poly}(r, d, m)$  if  $\text{char}(\mathbb{F}) \nmid r(r-1)$  or  $\mathbb{F} = \mathbb{C}$ . See [19, 6].
4.  $\text{Eqv}_{\mathbb{F}}(r, d, m, \text{IMM}_{r,d}) = \text{poly}(r, d, m)$  if  $\text{char}(\mathbb{F}) = 0$  or greater than  $d^c$  ( $c$  some fixed constant). See [21].

Thus, corollary 27 along with Theorem 28 and 26 gives a randomized  $\text{poly}(n, d, m, s)$ -time algorithm that outputs a generalized depth three representation, assuming  $\ell_{i,j}$ ’s are chosen randomly,  $g_i$ ’s belong to one of the polynomial families:  $\text{Det}_d$ ,  $\text{Perm}_d$ ,  $\text{IMM}_{r,d}$ ,  $\text{Sym}_{r,d}$ , monomials and the corresponding assumptions on  $\mathbb{F}$  holds.

## 4 Conclusion and open problems

We design an algorithm for learning generalized depth three circuits in the non-degenerate case. We follow the learning from lower bounds framework of [22, 7] and design new tools for proving that non-degeneracy conditions hold for random circuits, which could be useful for other such problems. Our model captures widely applicable problems such as tensor decomposition and Tucker decomposition as special cases. We are hopeful that our algorithms will find powerful applications in machine learning. We list some of the most interesting open problems next.

1. **Going beyond tensor decomposition.** Can we capture more general and powerful problems in machine learning via the model of generalized depth three circuits?
2. **Making the algorithms noise-resilient.** Can we make our algorithms robust to noise? That is, if one is given (explicitly or black box access)  $f(\mathbf{x}) = \sum_{i=1}^s g_i(\ell_{i1}, \dots, \ell_{im}) + E(\mathbf{x})$ , for some error term  $E(\mathbf{x})$ , can we approximately recover the summands? Such a noise-resilient version is relevant for machine learning applications. While our algorithm may seem too algebraic to be made robust, it is in fact linear algebraic and there is a good chance it can be made noise-resilient using standard tools such as SVD etc.
3. **Learning other arithmetic circuit models.** Can we learn other arithmetic circuit models in the non-degenerate case, for which we already have lower bounds? Perhaps the most appealing model to go for next is that of constant depth set-multilinear circuits. There are even new lower bounds for this model now [28].

---

## References

- 1 Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000. Conference version appeared in the proceedings of FOCS 1996.
- 2 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction of depth-4 multilinear circuits. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2144–2160. SIAM, 2020.



- 3 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 809–822, 2021.
- 4 Lieven De Lathauwer. A survey of tensor methods. In *2009 IEEE International Symposium on Circuits and Systems*, pages 2773–2776. IEEE, 2009.
- 5 Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009. Conference version appeared in the proceedings of COLT 2006.
- 6 Ankit Garg, Nikhil Gupta, Neeraj Kayal, and Chandan Saha. Determinant equivalence test over finite fields and over  $\mathbb{q}$ . In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- 7 Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 889–899. IEEE, 2020.
- 8 Fulvio Gesmundo and Joseph M. Landsberg. Explicit polynomial sequences with maximal spaces of partial derivatives and a question of k. mulmuley. *Theory of Computing*, 15(3):1–24, 2019. doi:10.4086/toc.2019.v015a003.
- 9 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *J. ACM*, 61(6):33:1–33:16, 2014. Conference version appeared in the proceedings of CCC 2013.
- 10 Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Efficient Reconstruction of Random Multilinear Formulas. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 778–787, 2011.
- 11 Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 625–642, 2012.
- 12 Ankit Gupta, Neeraj Kayal, and Youming Qiao. Random arithmetic formulas can be reconstructed efficiently. *Computational Complexity*, 23(2):207–303, 2014. Conference version appeared in the proceedings of CCC 2013.
- 13 R Harshman. Foundations of the parafac procedure: Model and conditions for an explanatory factor analysis. *Technical Report UCLA Working Papers in Phonetics 16, University of California, Los Angeles, Los Angeles, CA*, 1970.
- 14 Johan Håstad. Tensor Rank is NP-Complete. *J. Algorithms*, 11(4):644–654, 1990. Conference version appeared in the proceedings of ICALP 1989.
- 15 Majid Janzamin, Rong Ge, Jean Kossaifi, Anima Anandkumar, et al. Spectral learning on matrices and tensors. *Foundations and Trends® in Machine Learning*, 12(5-6):393–536, 2019.
- 16 Erich Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990.
- 17 Zohar Shay Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 274–285, 2009.
- 18 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421, 2011.
- 19 Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 643–662, 2012.
- 20 Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Computational Complexity*, 28(4):749–828, 2019.

- 21 Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of Full Rank Algebraic Branching Programs. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 21:1–21:61, 2017.
- 22 Neeraj Kayal and Chandan Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 413–424, 2019.
- 23 Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009. Conference version appeared in the proceedings of FOCS 2006.
- 24 Adam R. Klivans and Amir Shpilka. Learning restricted models of arithmetic circuits. *Theory of Computing*, 2(10):185–206, 2006. Conference version appeared in the proceedings of COLT 2003.
- 25 Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223, 2001.
- 26 Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.
- 27 Sue E Leurgans, Robert T Ross, and Rebecca B Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993.
- 28 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *FOCS 2021*, 2022.
- 29 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.
- 30 K. Pratt. Waring rank, parameterized and exact algorithms. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 806–823, 2019. doi:10.1109/FOCS.2019.00053.
- 31 Yaroslav Shitov. How hard is the tensor rank? *arXiv*, abs/1611.01559, 2016. arXiv:1611.01559.
- 32 Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 31:1–31:53, 2016.

## A Adjoint algebra and vector space decomposition

In this section, we prove some basic facts about the adjoint algebra and vector space decomposition, for completeness. We will start by stating that we can compute black-box access to partial derivatives of  $f$  from black-box access to  $f$ .

► **Fact 29.** *Given black-box access to a  $(n, d)$  polynomial  $f$  and a monomial  $\mathbf{x}^\alpha$ , a black-box access to  $\partial_{\alpha}^k f$  can be computed in deterministic  $\text{poly}(n, d^k)$  time.*

This follows from the fact that black-box access to a first-order derivative of  $f$  can be computed in deterministic polynomial time from black-box access to  $f$ .

Next, we define the adjoint algebra.

► **Definition 30 (Adjoint algebra).** *Consider a collection of linear maps  $\mathcal{L}$  from vector space  $U$  to vector space  $V$  (over a field  $\mathbb{F}$ ). The adjoint algebra for this collection of linear maps is defined as follows:*

$$\text{Adj}(\mathcal{L}, U, V) = \{(D, E) \mid D : U \rightarrow U, E : V \rightarrow V \text{ are linear maps s.t. } LD = EL \text{ for all } L \in \mathcal{L}\}.$$

Next we define the notion of a vector space decomposition.



► **Definition 31** (Vector space decomposition). *Consider a collection of linear maps  $\mathcal{L}$  from vector space  $U$  to vector space  $V$ . We say that  $U = U_1 \oplus \cdots \oplus U_s$  and  $V = V_1 \oplus \cdots \oplus V_s$  is a vector space decomposition for the triple  $(\mathcal{L}, U, V)$  if  $\mathcal{L}(U_i) \subseteq V_i$  for all  $i \in [s]$  (and at least one of  $U_i, V_i$  is a non-trivial subspace). We say that the decomposition is further indecomposable if the triples  $(\mathcal{L}, U_i, V_i)$  are indecomposable for all  $i$ .*

The next definition is about when the adjoint algebra is trivial.

► **Definition 32** (Trivial Adjoint algebra). *Consider a collection of linear maps  $\mathcal{L}$  from vector space  $U$  to vector space  $V$ . Also consider a decomposition,  $U = U_1 \oplus \cdots \oplus U_s$ ,  $V = V_1 \oplus \cdots \oplus V_s$  that is further indecomposable. We say that the adjoint algebra is trivial if*

$$\text{Adj}(\mathcal{L}, U, V) = \{(D, E) : \exists \text{ scalars } \lambda_1, \dots, \lambda_s \text{ s.t. } D|_{U_i} = \lambda_i \mathbb{1}_{U_i}, E|_{V_i} = \lambda_i \mathbb{1}_{V_i} \text{ for all } i \in [s]\}.$$

Next we define what we mean by uniqueness of decomposition.

► **Definition 33** (Uniqueness of decomposition). *Consider a collection of linear maps  $\mathcal{L}$  from vector space  $U$  to vector space  $V$ . Also consider a decomposition,  $U = U_1 \oplus \cdots \oplus U_s$ ,  $V = V_1 \oplus \cdots \oplus V_s$  that is further indecomposable. We say that the decomposition is unique if for any other further indecomposable decomposition,  $U = U'_1 \oplus \cdots \oplus U'_{s'}$ ,  $V = V'_1 \oplus \cdots \oplus V'_{s'}$ , it turns out that  $s = s'$  and there exists a permutation  $\pi : [s] \rightarrow [s]$  s.t.  $U'_i = U_{\pi(i)}$  and  $V'_i = V_{\pi(i)}$  for all  $i \in [s]$ .*

The next lemma states the uniqueness of decomposition in the case when the adjoint algebra is trivial. The uniqueness of decomposition holds in a much more general setting by a reduction to the Krull-Schmidt theorem (see [7]) but here we only focus on a special case that is relevant to us.

► **Lemma 34.** *Consider a collection of linear maps  $\mathcal{L}$  from vector space  $U$  to vector space  $V$ . Also consider a decomposition,  $U = U_1 \oplus \cdots \oplus U_s$ ,  $V = V_1 \oplus \cdots \oplus V_s$  that is further indecomposable. Suppose the adjoint algebra is trivial. Then the above is the unique decomposition that is further indecomposable.*

For proof of the above lemma see the full version.

Next we state an algorithm for vector space decomposition. While an algorithm in a much more general setting follows from known algorithms for module decomposition (see [7]), the algorithm we state here has the advantage that it is simpler and works for large enough fields (as opposed to algebraically closed fields). This algorithm is also present in [7] but not in a very explicit form, so we restate it here for completeness as well.

► **Lemma 35.** *Algorithm 2 with parameter  $\ell$  computes the correct decomposition when the adjoint algebra is trivial, with probability at least  $1 - \binom{s}{2}/\ell$ .*

**Proof.** Since the adjoint algebra is trivial,

$$\text{Adj}(\mathcal{L}, U, V) = \{(D, E) : \exists \lambda_1, \dots, \lambda_s \text{ s.t. } D|_{U_i} = \lambda_i \mathbb{1}_{U_i}, E|_{V_i} = \lambda_i \mathbb{1}_{V_i} \text{ for all } i \in [s]\}$$

Let  $(\lambda_1^{(j)}, \dots, \lambda_s^{(j)})$  be the tuple corresponding to  $(D_j, E_j)$ . Then

$$D'|_{U_i} = \left( \sum_{j=1}^s \mu_j \lambda_i^{(j)} \right) \mathbb{1}_{U_i}.$$

■ **Algorithm 2** Vector space decomposition when adjoint algebra is trivial.

**Input:** Set of linear maps  $\mathcal{L}$  between vector spaces  $U$  and  $V$  s.t. the triple  $(\mathcal{L}, U, V)$  admits a further indecomposable decomposition  $U = U_1 \oplus \cdots \oplus U_s$ ,  $V = V_1 \oplus \cdots \oplus V_s$ . Also the adjoint algebra is trivial.

**Output:**  $s$  vector spaces  $U'_1, \dots, U'_s$  s.t. there exists a permutation  $\pi : [s] \rightarrow [s]$  s.t.  $U'_i = U_{\pi(i)}$ .

**Subroutine:** Diagonalizing a diagonalizable linear map  $D : U \rightarrow U$ .

**Parameters:** Randomness parameter  $\ell$ .

- 1: Compute a basis  $(D_1, E_1), \dots, (D_s, E_s)$  of the adjoint algebra  $\text{Adj}(\mathcal{L}, U, V)$  (this is a system of linear equations). (If dimension is not  $s$ , then abort).
- 2: Pick  $\mu_1, \dots, \mu_s$  uniformly at random from a set of size  $\ell$ . Set  $D' = \mu_1 D_1 + \cdots + \mu_s D_s$ .
- 3: Compute the eigenvalues of  $D'$ . If it has  $s$  distinct eigenvalues, call them  $\lambda_1, \dots, \lambda_s$ . If not (or it is not diagonalizable), abort.
- 4: Set  $U'_i$  to be the eigenspace of  $D'$  corresponding to  $\lambda_i$ .

We know that the vectors  $(\lambda_1^{(j)}, \dots, \lambda_s^{(j)})$ , for  $j \in [s]$ , are linearly independent. Hence the vectors  $(\lambda_i^{(1)}, \dots, \lambda_i^{(s)})$ , for  $i \in [s]$ , are also linearly independent. Hence for  $i \neq i'$ , the linear polynomial (in the  $\mu_j$ 's)  $\sum_{j=1}^s \mu_j (\lambda_i^{(j)} - \lambda_{i'}^{(j)})$  is non-zero and hence if the  $\mu_j$ 's are chosen at random from a set of size  $\ell$ , then with probability at least  $1 - 1/\ell$ ,

$$\sum_{j=1}^s \mu_j (\lambda_i^{(j)} - \lambda_{i'}^{(j)}) \neq 0.$$

By a union bound, with probability at least  $1 - \binom{s}{2}/\ell$ , for any  $i \neq i'$ ,

$$\sum_{j=1}^s \mu_j (\lambda_i^{(j)} - \lambda_{i'}^{(j)}) \neq 0.$$

Thus there are  $s$  distinct eigenvalues of  $D'$ , one each corresponding to the eigenspace  $U_i$ . This completes the proof. ◀

We next define the concept of isomorphism between tuples  $(\mathcal{L}, U, V)$  and  $(\mathcal{L}', U', V')$ , and relate the adjoint algebras for isomorphic tuples.

► **Definition 36** (Isomorphic tuples). *We say that  $(\mathcal{L}, U, V)$  and  $(\mathcal{L}', U', V')$  are isomorphic if there is an invertible linear transformation  $\phi : \langle \mathcal{L} \rangle \rightarrow \langle \mathcal{L}' \rangle$  and invertible linear maps  $T : U \rightarrow U'$ ,  $S : V \rightarrow V'$  s.t.  $\phi(L)T = SL$  for all  $L \in \mathcal{L}$ .*

► **Proposition 37** (Adjoint algebras under isomorphism). *Let  $(\mathcal{L}, U, V)$  and  $(\mathcal{L}', U', V')$  be isomorphic tuples. Then  $(D, E) \in \text{Adj}(\mathcal{L}, U, V)$  iff  $(TDT^{-1}, SES^{-1}) \in \text{Adj}(\mathcal{L}', U', V')$ .*

**Proof.** It suffices to prove one direction because of symmetry. Suppose  $(D, E) \in \text{Adj}(\mathcal{L}, U, V)$  i.e.  $LD = EL$  for all  $L \in \mathcal{L}$ . Then

$$\phi(L)TDT^{-1} = SLDT^{-1} = SELT^{-1} = SES^{-1}\phi(L)$$

for all  $L \in \mathcal{L}$ . Since  $\{\phi(L)\}_{L \in \mathcal{L}}$  span  $\langle \mathcal{L}' \rangle$ , we get that  $L'TDT^{-1} = SES^{-1}L'$  for all  $L' \in \mathcal{L}'$ . That is  $(TDT^{-1}, SES^{-1}) \in \text{Adj}(\mathcal{L}', U', V')$ . ◀

This yields the following corollary:

► **Corollary 38.** *Let  $(\mathcal{L}, U, V)$  and  $(\mathcal{L}', U', V')$  be isomorphic tuples. Then  $\text{Adj}(\mathcal{L}, U, V)$  is trivial iff  $\text{Adj}(\mathcal{L}', U', V')$  is trivial.*

Next we state an instantiation of the above corollary which we need for our analysis.

► **Corollary 39.** *Let  $g \in \mathbb{F}[\mathbf{z}]_d$ ,  $\mathbf{z} = (z_1, \dots, z_m)$ . Also  $h = g(\ell_1, \dots, \ell_m)$ , where  $\ell_i$ 's linearly independent linear forms in the  $\mathbf{z}$  variables. Then  $\text{Adj}(\partial_{\mathbf{z}}^=1, \langle \partial_{\mathbf{z}}^=k g \rangle, \langle \partial_{\mathbf{z}}^=(k+1) g \rangle)$  is trivial iff  $\text{Adj}(\partial_{\mathbf{z}}^=1, \langle \partial_{\mathbf{z}}^=k h \rangle, \langle \partial_{\mathbf{z}}^=(k+1) h \rangle)$  is trivial.*

For proof of the above lemma see the full version.

## B Linear algebra with black boxes

In Algorithm 1, we need to perform linear algebra given black boxes for polynomials. We give references for how to do this here. We will need the following lemma from [18].

► **Lemma 40** (Section A.1 in [18]). *Given black boxes for the polynomials  $f_1, \dots, f_\ell \in \mathbb{F}[\mathbf{x}]_d$ , there is a randomized  $\text{poly}(n, \ell, d)$  time algorithm that computes a basis for the following vector space*

$$(f_1, \dots, f_\ell)^\perp := \{(\alpha_1, \dots, \alpha_\ell) : \sum_{i=1}^{\ell} \alpha_i f_i = 0\}.$$

In particular, we get the following corollary.

► **Corollary 41.** *Given black boxes for the polynomials  $f_1, \dots, f_\ell \in \mathbb{F}[\mathbf{x}]_d$  which are linearly independent and for a  $p \in \mathbb{F}[\mathbf{x}]_d$  which linearly depends on  $f_1, \dots, f_\ell$ , there is a randomized  $\text{poly}(n, \ell, d)$  time algorithm that computes  $\beta_1, \dots, \beta_\ell$  s.t.*

$$p = \sum_{i=1}^{\ell} \beta_i f_i.$$

Using Corollary 41, one can compute the matrices corresponding to the linear maps  $\mathcal{L}$  in Algorithm 2 if one is given only black boxes for bases of  $U$  and  $V$ . One can also carry out the Step 3 in Algorithm 1 using Corollary 41.

## C Reducing the field size

In this section, we provide a sketch of how to reduce the field size in Theorem 1. For this, we will have to change the non-degeneracy conditions slightly. We state the new non-degeneracy conditions next for the circuit  $f = \sum_{i=1}^s g_i(\ell_{i1}, \dots, \ell_{im})$ .

1. For each  $i \in [s]$ , the linear forms  $(\ell_{i1}, \dots, \ell_{im})$  are linearly independent. Let us denote by  $d_{i,k} := \dim(\partial_{\mathbf{z}}^=k g_i(\mathbf{z}))$ . Consider the vector spaces  $U := \langle \partial^=k f \rangle$ ,  $V := \langle \partial^=(k+1) f \rangle$  (here the partials are w.r.t. the  $\mathbf{x}$  variables). We impose  $\dim(U) = \sum_{i=1}^s d_{i,k}$  and  $\dim(V) = \sum_{i=1}^s d_{i,k+1}$ .
2. We impose that  $\text{Adj}(\partial^=1, U, V)$  is trivial i.e.  $\dim(\text{Adj}(\partial^=1, U, V)) = s$ .
3. This is the same as the Item 3 in Section 2.1.

Let us first compare these conditions with the conditions in Section 2.1. It can be verified that Item 1 is the same as  $U = U_1 \oplus \cdots \oplus U_s$  and  $V = V_1 \oplus \cdots \oplus V_s$  i.e. Item 2 in Section 2.1. Item 2 here is new and assuming this implies uniqueness of decomposition and this can be used directly in the proof of Theorem 5 (instead of Item 1 in Section 2.1).


We now sketch the argument on why random  $\ell_{i,j}$ 's would satisfy these conditions. In Sections 3.1 and 3.2, we provide a particular setting of  $\ell_{i,j}$ 's s.t. Items 1 and 2 in Section 2.1) are satisfied. These imply that Items 1 and 2 stated here are satisfied (for Item 2, one would need to combine the proof of Theorem 5 and Item 3). So we just need the Schwartz-Zippel argument. First consider Item 1. The condition about  $U$ , for example, is about the rank of a matrix whose dimensions are  $\binom{n+k-1}{k} \times \binom{n+d-k-1}{d-k}$  and entries are homogeneous polynomials of degree  $k$  in the coefficients of  $\ell_{i,j}$ 's. We know that the rank is always at most  $D := \sum_{i=1}^s d_{i,k}$  and also that the rank is equal to  $D$  for a particular setting of  $\ell_{i,j}$ 's. This implies the existence of a  $D \times D$  minor which has full rank for a particular setting of  $\ell_{i,j}$ 's. Hence by Schwartz-Zippel lemma, we get that this minor is full rank for a random choice of  $\ell_{i,j}$ 's if the field size is at least  $\text{poly}(D, k) = \text{poly}(s \binom{n+k-1}{k}, k)$  which is  $\text{poly}(n, d, s)$  since we choose  $\Theta(\log(s)/\log(n))$ .

Regarding the condition on the adjoint, note that adjoint is the solution to a linear system of equations. Hence  $\dim(\text{Adj}(\partial^{=1}, U, V)) = s$  is equivalent to the corank of a matrix being at most  $s$  (it is at least  $s$  by definition). The dimensions of the matrix are  $(\dim(U)^2 + \dim(V)^2) \times (n \cdot \dim(U) \cdot \dim(V))$  and the entries are homogeneous polynomials of degree  $O(k)$  in the coefficients of  $\ell_{i,j}$ 's. Again here the Schwartz-Zippel argument can be carried out over a field of size  $\text{poly}(\dim(U), \dim(V), n, k)$  which is  $\text{poly}(n, d, s)$  because of the choice of  $k$ .

# Lower Bound Methods for Sign-Rank and Their Limitations

Hamed Hatami ✉🏠

School of Computer Science, McGill University, Montreal, Canada

Pooya Hatami ✉🏠

Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA

William Pires ✉

School of Computer Science, McGill University, Montreal, Canada

Ran Tao ✉

Department of Mathematics and Statistics, McGill University, Montreal, Canada

Rosie Zhao ✉

School of Computer Science, McGill University, Montreal, Canada

---

## Abstract

The *sign-rank* of a matrix  $A$  with  $\pm 1$  entries is the smallest rank of a real matrix with the same sign pattern as  $A$ . To the best of our knowledge, there are only three known methods for proving lower bounds on the sign-rank of explicit matrices: (i) Sign-rank is at least the VC-dimension; (ii) Forster’s method, which states that sign-rank is at least the inverse of the largest possible average margin among the representations of the matrix by points and half-spaces; (iii) Sign-rank is at least a logarithmic function of the density of the largest monochromatic rectangle.

We prove several results regarding the limitations of these methods.

- We prove that, qualitatively, the monochromatic rectangle density is the strongest of these three lower bounds. If it fails to provide a super-constant lower bound for the sign-rank of a matrix, then the other two methods will fail as well.
- We show that there exist  $n \times n$  matrices with sign-rank  $n^{\Omega(1)}$  for which none of these methods can provide a *super-constant* lower bound.
- We show that sign-rank is at most an exponential function of the deterministic communication complexity with access to an equality oracle. We combine this result with Green and Sanders’ quantitative version of Cohen’s idempotent theorem to show that for a large class of sign matrices (e.g., XOR-lifts), sign-rank is at most an exponential function of the  $\gamma_2$  norm of the matrix. We conjecture that this holds for all sign matrices.
- Towards answering a question of Linial, Mendelson, Schechtman, and Shraibman regarding the relation between sign-rank and discrepancy, we conjecture that sign-ranks of the  $\pm 1$  adjacency matrices of hypercube graphs can be arbitrarily large. We prove that none of the three lower bound techniques can resolve this conjecture in the affirmative.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity; Theory of computation  $\rightarrow$  Boolean function learning

**Keywords and phrases** Average Margin, Communication complexity, margin complexity, monochromatic rectangle, Sign-rank, Unbounded-error communication complexity, VC-dimension

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.22

**Category** RANDOM

**Related Version** *Full-Length Version*: <https://eccc.weizmann.ac.il/report/2022/079/>

**Funding** *Hamed Hatami*: Supported by an NSERC grant.

*Pooya Hatami*: Supported by NSF grant CCF-1947546.

**Acknowledgements** We wish to thank Shachar Lovett and Shay Moran for helpful discussions. We would like to thank Shay Moran for sharing the proof of Proposition B.3 with us.



© Hamed Hatami, Pooya Hatami, William Pires, Ran Tao, and Rosie Zhao;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 22; pp. 22:1–22:24



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

A *sign matrix* is a matrix with  $\pm 1$  entries. The *sign-rank* of a sign matrix  $A_{m \times n}$  is the smallest rank of a real matrix  $B_{m \times n}$  such that the entries of  $B$  are nonzero and have the same signs as their corresponding entries in  $A$ . This fundamental notion arises naturally in areas as diverse as learning theory [5, 27, 42, 44, 13, 14], discrete geometry and geometric graphs [1, 18, 19, 45, 12], communication complexity [36, 9, 41, 24], circuit complexity [37, 7, 43], and the theory of Banach spaces [33, 35].

The notion of sign-rank was formally defined in 1986 in connection with randomized communication complexity in the unbounded-error model [36]. After almost four decades of research, sign-rank remains one of the most elusive matrix parameters in discrete analysis. To the best of our knowledge, there are only three known methods for proving lower bounds on the sign-rank of an explicit matrix: VC-dimension, size of the largest monochromatic rectangle, and Forster's method, and among those, only Forster's method can imply super-logarithmic lower bounds.

The results presented in this paper arose from our attempts to solve two fundamental open problems about sign-rank, presented as Question 1.4 and Question 1.11 below. Attempting to give negative answers to these questions, we proved that none of the known techniques could yield adequate sign-rank lower bounds for these purposes. Of course, this observation does not necessarily imply that the techniques are inherently weak, as there is a possibility that the correct answer to both questions is positive. As a natural next step, we examined the limitations of these techniques more carefully and, among other things, proved the existence of  $n \times n$  matrices with sign-rank  $n^{\Omega(1)}$ , for which none of these methods could provide a *super-constant* lower bound.

We start by reviewing and reformulating the results that are relevant to this article.

### Counting argument

Shortly after the introduction of sign-rank in [36], Alon, Frankl, and Rödl [1] used results of [34, 46, 47] on the number of connected components of real algebraic varieties and obtained a linear lower bound on the sign-rank of random sign matrices. This argument was later refined in [2, Lemma 24] to the following bound on the number of low sign-rank matrices.

► **Lemma 1.1** (See [2, Lemma 24]). *For  $d \leq \frac{n}{2}$ , the number of  $n \times n$  sign matrices of sign-rank at most  $d$  does not exceed  $(O(n/d))^{2dn} \leq 2^{O(dn \log(n))}$ .*

It follows from Lemma 1.1 that most  $n \times n$  sign matrices have sign-rank  $\Omega(n)$ .

### The VC-dimension lower bound

The *Vapnik-Chervonenkis* (VC) dimension of a sign matrix  $A$  is the largest  $k$  such that  $A$  contains a submatrix with  $k$  columns and  $2^k$  distinct rows. To state the relation between the VC dimension and sign-rank, we discuss a geometric definition of sign-rank.

A real matrix  $B_{\mathcal{X} \times \mathcal{Y}}$  has rank  $d$  iff the entries of  $B$  can be represented as  $B_{xy} = \langle u_x, v_y \rangle$  for vectors  $u_x, v_y \in \mathbb{R}^d$ . Since the normalization of these vectors does not affect the signs of  $\langle u_x, v_y \rangle$ , we can restate the definition of sign-rank as follows.

► **Definition 1.2** (Sign-rank). *The sign-rank of a sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , denoted by  $\text{rank}_{\pm}(A)$ , is the smallest  $d$  such that there exist unit vectors  $u_x, v_y \in \mathbb{R}^d$  with  $A_{xy} = \text{sgn}(\langle u_x, v_y \rangle)$  for all  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ .*

The vectors in Definition 1.2 represent  $A$  as points and half-spaces in the  $d$ -dimensional space:  $A_{xy} = 1$  iff the point  $u_x$  belongs to the half-space  $\{z : \langle z, v_y \rangle > 0\}$ . Since the VC dimension of any such configuration of points and half-spaces in  $\mathbb{R}^d$  is at most  $d$ , we have

$$\text{rank}_{\pm}(A) \geq \text{VC}(A). \quad (1)$$

This lower bound was already implicit in the paper of Paturi and Simon [36, Theorem 4]. Since the VC dimension of every  $n \times n$  matrix is at most  $\log n$ , this method cannot prove super-logarithmic lower bounds on sign-rank. In addition, Alon, Moran, and Yehudayoff [2] established strong separations between the two parameters. For example, they showed that there are  $n \times n$  sign matrices of VC dimension 3 that have sign-rank  $\Omega\left(\frac{\sqrt{n}}{\log n}\right)$ .

### Margin and Discrepancy

There is another natural parameter that is associated with the representations of a sign matrix as points and half-spaces. The quantity  $\min_{x,y} |\langle u_x, v_y \rangle|$  is called the *margin* of such a representation; it measures the smallest distance between the points  $u_x$  and the hyperplanes defined by  $v_y$ .

► **Definition 1.3 (Margin).** *The margin of a sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  is*

$$m(A) := \sup \min_{x,y} |\langle u_x, v_y \rangle|,$$

where the supremum is over all  $d \in \mathbb{N}$  and unit vectors  $u_x, v_y \in \mathbb{R}^d$  with  $A_{xy} = \text{sgn}(\langle u_x, v_y \rangle)$ .

Linial and Shraibman [30] proved that margin essentially coincides with the well-studied parameter of *discrepancy* in communication complexity, defined as

$$\text{disc}(A) := \inf_{\mu} \max_{\substack{S \subseteq \mathcal{X} \\ T \subseteq \mathcal{Y}}} |\mathbb{E}_{xy \sim \mu} [A_{xy} \mathbf{1}_S(x) \mathbf{1}_T(y)]|, \quad (2)$$

where the infimum is over all probability distributions  $\mu$  on  $\mathcal{X} \times \mathcal{Y}$ . They proved

$$\text{disc}(A) \leq m(A) \leq 8 \text{disc}(A).$$

The notion of discrepancy is a well-understood parameter, and many lower bounds in communication complexity are established by proving that the discrepancy of the corresponding matrix is small. Such proofs often entail finding a “hard” distribution  $\mu$  such that the maximum in Equation (2) is small. We shall discuss this more later in the context of Forster’s lower bound method.

The problem of understanding the relation between sign-rank and margin is an important one because these notions optimize two fundamental attributes of the geometric representations of the matrix. Sign-rank minimizes the dimension while allowing the margin to be arbitrarily small. Margin maximizes the margin of the representation while allowing the dimension to be arbitrarily large.

Hatami, Hosseini, and Lovett [24] constructed  $n \times n$  sign matrices that have a very small margin (equivalently discrepancy) of  $O\left(\frac{\log(n)}{n^{1/8}}\right)$  while their sign-rank is only 3. The converse direction regarding the question of margin vs sign-rank remains open. Does large margin imply small sign-rank?

► **Question 1.4.** *Is there a function  $\eta$  such that for every sign matrix  $A$ , we have  $\text{rank}_{\pm}(A) \leq \eta(m(A)^{-1})$ ?*



Question 1.4 is essentially due to [29, Corollary 3.2, Lemma 4.2, and Section 8], where they proved the inequality  $\text{rank}_\pm(A) \leq m(A)^{-2} \cdot \log(n)$ , and asked whether the log factor in this inequality is necessary.

It is known that margin, discrepancy, public-coin randomized communication complexity, and approximate  $\gamma_2$  norms are all related, in the sense that each can be used to provide an upper bound on any other (see Section 3.3 for more details). Therefore, one can *equivalently* restate Question 1.4, with  $m(A)^{-1}$  replaced with any of the mentioned parameters. We propose the following conjecture that would imply a negative answer to Question 1.4, as  $m(Q_d)^{-1} = O(1)$  (see Proposition 3.4).

► **Conjecture 1.5** (Sign-rank of hypercube graphs). *Let  $Q_d$  be the  $2^{d-1} \times 2^{d-1}$  sign matrix whose rows and columns are indexed with, respectively, odd-parity and even-parity elements of  $\{0, 1\}^d$ , and  $Q_d(x, y) = -1$  iff  $x$  and  $y$  differ in exactly one coordinate. Then*

$$\lim_{d \rightarrow \infty} \text{rank}_\pm(Q_d) = \infty.$$

### Forster's sign-rank lower bound

For *explicit* matrices, the VC-dimension lower bound remained state of the art for almost two decades until the breakthrough work of Forster [15]. Forster used a convex geometric approach to prove a *linear* lower bound on the sign-rank of Hadamard matrices, establishing the first super-logarithmic lower bound on the sign-rank of an explicit matrix.

Forster's proof first transforms the vectors  $v_y$  to be in isotropic position, and then uses the anti-concentration of measure in low dimensions to show that the average  $\mathbb{E}_y |\langle u_x, v_y \rangle|$  is relatively large for every vector  $u_x$ . In other words, the “average margin” of such a representation is large. This powerful fact in convex geometry was first established by Barthe [4] as a key step in his proof of a reverse form of the Brascamp-Lieb inequality. It seems that Forster was unaware of Barthe's paper, and he gave a different proof in his paper [15].

We use a variation of the aforementioned geometric fact due to [25] in order to formulate a slight generalization of Forster's approach that allows arbitrary distributions on  $\mathcal{Y}$ .

► **Definition 1.6** (Average margin). *The average margin of a sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  with respect to a probability distribution  $\nu$  on  $\mathcal{Y}$  is defined as*

$$m_\nu^{\text{avg}}(A) = \sup_x \min_{y \sim \nu} \mathbb{E}_{y \sim \nu} |\langle u_x, v_y \rangle|,$$

where the supremum is over all sign-representations of  $A$  using unit vectors  $u_x, v_y \in \mathbb{R}^d$  for any  $d$ . The average margin of  $A$  is defined as  $m^{\text{avg}}(A) = \inf_\nu m_\nu^{\text{avg}}(A)$ .

Note that  $m^{\text{avg}}(A) \geq m(A)$  since  $\mathbb{E}_{y \sim \nu} |\langle u_x, v_y \rangle| \geq \min_y |\langle u_x, v_y \rangle|$ . A slightly different notion of average margin is studied by Kallweit and Simon in [26], however, since  $m^{\text{avg}}(A)$  is always smaller than Kallweit and Simon's notion of average margin, it provides a stronger lower bound on sign-rank in Theorem 1.7 below. We summarize Forster's approach as the following theorem.

► **Theorem 1.7** (Forster [15]). *For every sign-matrix  $A$ , we have*

$$\text{rank}_\pm(A) \geq m^{\text{avg}}(A)^{-1}.$$

Since our formulation of Forster's approach is slightly more general than the original statement of Forster's theorem [15], we provide a proof of Theorem 1.7 in the full-length version of this paper. Later, in Proposition 2.2, we show how Theorem 1.7 implies Forster's original statement [15] as well as Linial and Shraibman's refinement of it [31].

Forster's original paper [15] applies the average margin method to show that sign-rank is large when the spectral norm is small. Subsequent works [16, 17, 37] showed that, more generally, this method can extend discrepancy bounds to lower bounds on sign-rank if the witnessing hard distribution  $\mu$  in Equation (2) is well-spread on most of the entries. This is intuitive considering that discrepancy is equivalent to margin, and the lower bound in Theorem 1.7 is based on average margin.

The following proposition shows that VC dimension is essentially a weaker lower bound technique than Forster's method.

► **Proposition 1.8.** *For every sign matrix  $A$ , we have  $m^{\text{avg}}(A)^{-1} \geq \sqrt{\text{VC}(A)}$ .*

**Proof.** Suppose  $\text{VC}(A) = k$ . By the definition of the VC dimension,  $A$  contains a  $2^k \times k$  submatrix  $U_k$  with all the possible different rows. Note that

$$U_k^T U_k = 2^k \mathbf{I}_k.$$

In particular, we have  $\|U_k\| = 2^{k/2}$ , gives

$$m^{\text{avg}}(A) \leq m^{\text{avg}}(U_k) \leq \frac{2^{k/2}}{\sqrt{k} 2^k} = \frac{1}{\sqrt{k}},$$

where the second inequality follows from Proposition 2.2 below. ◀

### Monochromatic rectangles and sign-rank

A submatrix of a matrix  $A$  is called a *monochromatic rectangle* if all the entries in this submatrix have the same value. In addition to VC-dimension and Forster's method, there is a third known approach for proving super-constant lower bounds on sign-ranks of explicit matrices, which is based on the size of the largest monochromatic rectangle.

We define the following parameter based on the size of monochromatic rectangles.

► **Definition 1.9** (Monochromatic rectangle ratio). *For every sign-matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , define*

$$\text{rect}(A) = \inf_{\mu \times \nu} \max_R \mu \times \nu(R),$$

where the infimum is over all product probability measures  $\mu \times \nu$  on  $\mathcal{X} \times \mathcal{Y}$ , and the maximum is over all monochromatic rectangles in  $A$ .

Alon, Pach, Pinchasi, Radoičić and Sharir [3] proved that every  $m \times n$  sign matrix of sign-rank  $d$  contains an  $\frac{m}{2^{d+1}} \times \frac{n}{2^{d+1}}$  monochromatic rectangle. Similar bounds are obtained by Fox, Pach, and Suk [20] using the cutting lemma of Chazelle [10]. We provide a different proof in Proposition A.1. While Proposition A.1 follows from the result of [20], we believe our short and simple proof could provide some geometric intuition for why matrices of low sign-rank contain large monochromatic rectangles.

The following relation between sign-rank and monochromatic rectangle ratio follows from the bound of [3, Theorem 1.3].

► **Theorem 1.10** (See [3, Theorem 1.3]). *For every sign-matrix  $A$ , we have*

$$\text{rank}_{\pm}(A) \geq \frac{\log_2(\text{rect}(A)^{-1})}{2} - 1. \quad (3)$$

Note that similar to the VC-dimension, Theorem 1.10 cannot imply super-logarithmic lower bounds on sign-rank, since every  $n \times n$  sign matrix satisfies  $\text{rect}(A) \geq \frac{1}{2^n}$ . To see the latter claim, note that for every probability distribution  $\mu$  on the rows, there is always a row  $x$  with measure  $\geq \frac{1}{n}$ , and any probability distribution  $\nu$  over the columns has measure at least  $\frac{1}{2}$  on either the 1's or the -1's of this row.

### Sign-rank of semi-algebraic matrices, an open problem

A real semi-algebraic set in  $\mathbb{R}^d$  is the set of all points that satisfy a given finite Boolean combination of polynomial inequalities in the  $d$  coordinates. We say that such a set has *description complexity*  $t$  if in some representation, the number of inequalities and the degrees of the corresponding polynomials are all bounded from above by  $t$ .

Every collection of points  $u_1, \dots, u_m \in \mathbb{R}^d$  and semi-algebraic sets  $K_1, \dots, K_n \subseteq \mathbb{R}^d$  define a sign matrix  $A_{m \times n}$  where  $A_{ij} = 1$  iff  $u_i \in K_j$ . We say that  $A$  has a representation in  $\mathbb{R}^d$  with *description complexity*  $t$  if every  $K_i$  has description complexity  $t$ .

We call a class of sign matrices *semi-algebraic* if there exists  $d, t \in \mathbb{N}$  such that every matrix in this class has a representation in  $\mathbb{R}^d$  of description complexity at most  $t$ . Semi-algebraic classes of sign matrices capture natural geometric constructions of graphs on finite dimensional real spaces, such as interval graphs, incidence graphs, disc graphs, and more generally, all graph classes where vertices are points in a real Euclidean space and the edges are defined by a semi-algebraic relation of constant complexity.

An affirmative answer to the following question would imply that semi-algebraic classes of sign matrices coincide with bounded sign-rank classes.

► **Question 1.11** (Sign-rank of semi-algebraic matrices). *Is there a function  $\eta : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  such that every sign matrix with a  $d$ -dimensional representation of description complexity  $t$  has sign-rank at most  $\eta(d, t)$ ?*

For the converse direction, note that if  $\text{rank}_{\pm}(A) = \eta$ , then the corresponding sign-representation of  $A$  using vectors  $u_i, v_j \in \mathbb{R}^n$  is a representation with description complexity 1: We have  $A_{ij} = 1$  iff  $u_i \in \{x \in \mathbb{R}^d : \langle v_j, x \rangle > 0\}$ , and note that  $\langle v_j, x \rangle$  is a polynomial of degree 1 in the coordinates of  $x$ .

Let  $\Gamma : \{-1, 1\}^t \rightarrow \{-1, 1\}$  be a predicate and let  $A_1, \dots, A_t$  be  $m \times n$  sign matrices. Let  $\Gamma(A_1, \dots, A_t)$  denote the  $m \times n$  sign matrix with  $ij$ -entries  $\Gamma(A_1(i, j), \dots, A_t(i, j))$ . As we will discuss in Appendix B, a simple linearization trick shows that Question 1.11 can be reformulated as the following question.

► **Question 1.12** (First reformulation of Question 1.11). *Is there a function  $\eta : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  such that for every predicate  $\Gamma : \{-1, 1\}^t \rightarrow \{-1, 1\}$  and every set of  $m \times n$  sign matrices  $A_1, \dots, A_t$  with sign-ranks at most  $d$ , we have*

$$\text{rank}_{\pm}(\Gamma(A_1, \dots, A_t)) \leq \eta(d, t)?$$

The formulation in Question 1.12 is interesting from the perspective of learning theory: Consider a binary data set encoded as a sign matrix  $\Gamma$ . The entry  $\Gamma_{ij}$  is called the label of the data-point  $j$  according to the concept  $i$ . Suppose that these labels are determined by a few other binary labels. For example, whether a person  $i$  is likely to watch a movie  $j$  may be determined by whether  $j$  is the genre of movie that they like, whether  $j$  features some of their favorite actors, and whether  $j$  is available at a theater near them. Now suppose that each of these latter binary data sets has a low-dimensional representation. Does this mean that our data set has a low-dimensional representation?

The formulation in Question 1.12 is also interesting from the perspective of communication complexity: since the logarithm of sign-rank is equivalent to the unbounded-error communication complexity (see Equation (8) below), Question 1.12 asks whether a matrix constructed by the entrywise application of a logical predicate to matrices  $A_1, \dots, A_t$ , each with a small unbounded-error communication complexity, must have a small unbounded-error communication complexity. It is straightforward to show that a similar statement is indeed true in the *bounded-error* case.

Question 1.12 can be further simplified to a fascinating simple-to-state question. Let  $A \wedge B$  be the matrix whose  $ij$ -th entries are the point-wise minimums of the entries of  $A$  and  $B$ , corresponding to the Boolean AND operator. Let  $\neg A := -A$ . Recall that  $\{\wedge, \neg\}$  is a complete basis, i.e., it is a functionally complete set in the logical sense. Hence the function  $\Gamma$  in Question 1.12 can be implemented using the two operations  $\wedge$  and  $\neg$ , and since for every sign matrix  $A$ , we have  $\text{rank}_{\pm}(A) = \text{rank}_{\pm}(\neg A)$ , Question 1.12 is equivalent to the following.

► **Question 1.13** (Second reformulation of Question 1.11). *Is there a function  $\eta : \mathbb{N} \rightarrow \mathbb{N}$  such that for every two sign matrices  $A$  and  $B$  with sign-ranks at most  $d$ , we have  $\text{rank}_{\pm}(A \wedge B) \leq \eta(d)$ ?*

In comparison, let us consider the Hadamard product  $A \circ B$  of two matrices  $A$  and  $B$ , which corresponds to entrywise  $\oplus$  operator in the Boolean setting. It is well-known that  $\text{rank}(A \circ B) \leq \text{rank}(A) \cdot \text{rank}(B)$ , which implies that for every two  $m \times n$  sign matrices  $A$  and  $B$ , we have

$$\text{rank}_{\pm}(A \circ B) \leq \text{rank}_{\pm}(A) \cdot \text{rank}_{\pm}(B).$$

However, this cannot be used in a similar argument as the AND case above to answer Question 1.11, as  $\{\oplus, \neg\}$  is not a complete basis.

### Contributions and organization

For the following discussion, recall the three aforementioned lower bound techniques for sign-rank:

$$\text{VC}(A) \leq \text{rank}_{\pm}(A), \quad \text{m}^{\text{avg}}(A)^{-1} \leq \text{rank}_{\pm}(A), \quad \frac{\log_2(\text{rect}(A)^{-1})}{2} - 1 \leq \text{rank}_{\pm}(A),$$

and note that all these lower bounds are non-increasing when restricting to submatrices: For every submatrix  $M$  of  $A$ , we have

$$\text{VC}(M) \leq \text{VC}(A), \quad \text{m}^{\text{avg}}(M)^{-1} \leq \text{m}^{\text{avg}}(A)^{-1}, \quad \text{rect}(M)^{-1} \leq \text{rect}(A)^{-1}.$$

- In Section 3.1, we study the relation between the average margin and the rectangle ratio. In Theorem 3.1, we prove that

$$\text{m}^{\text{avg}}(A)^{-1} \leq \text{rect}(A)^{-1},$$

which, combined with Proposition 1.8, shows

$$\sqrt{\text{VC}(A)} \leq \text{m}^{\text{avg}}(A)^{-1} \leq \text{rect}(A)^{-1}. \quad (4)$$

These inequalities demonstrate that if the monochromatic rectangle ratio cannot provide a super-constant lower bound for the sign-rank of a matrix, then the other two methods will fail as well.

The significance of Theorem 3.1 is that proving an upper bound on  $\text{rect}(A)^{-1}$  is often much easier than directly analyzing the average margin. This allows us to demonstrate some limitations of Forster's method.

- In Section 3.2, we combine Theorem 3.1 with a counting argument to prove our main separation result: In Theorem 3.2, we show the existence of  $n \times n$  sign matrices  $A$  that have sign-rank  $n^{\Omega(1)}$  but  $\text{VC}(A)$ ,  $\text{m}(A)^{-1}$  and  $\text{rect}(A)^{-1}$  are all  $O(1)$ . In other words, there exists matrices of very large sign-rank such that none of the known lower bound techniques can provide a lower bound that is larger than  $O(1)$ .
- In Section 3.3, we discuss the limitation of sign-rank lower bounds in answering Question 1.4 and Conjecture 1.5. In particular, in Proposition 3.4 we observe that  $\text{rect}(Q_d)^{-1} = O(1)$ , and thus none of the known lower bound methods can prove Conjecture 1.5.
- In Section 3.4, we study a question that is closely related to the relation between margin and sign-rank (i.e., Question 1.4). As discussed above, one can equivalently rephrase Question 1.4 in terms of upper-bounding sign-rank by a function of the approximate  $\gamma_2$  norm (see Definition 2.1). As stated in Conjecture 1.5, we believe the answer to be negative. However, one can strengthen the assumption and ask whether the sign-rank can be upper-bounded by a function of the  $\gamma_2$  norm instead:

► **Conjecture 1.14.** *There exists a function  $\eta$  such that for every sign matrix  $A$ , we have  $\text{rank}_{\pm}(A) \leq \eta(\|A\|_{\gamma_2})$ .*

Towards proving Conjecture 1.14, in Theorem 3.8, we show that

$$\text{rank}_{\pm}(A) \leq 4^{\text{D}^{eq}(A)}, \quad (5)$$

where  $\text{D}^{eq}(A)$  denotes the deterministic communication complexity of the matrix  $A$  with access to an equality oracle. In Corollary 3.9, we combine this with Green and Sanders' [21, 22, 39, 38, 40] quantitative versions of Cohen's idempotent theorem and a theorem of [23] to verify Conjecture 1.14 for a broad class of sign-matrices: We prove there exists a function  $\eta$  such that if  $f : G \rightarrow \{-1, 1\}$  for a finite group  $G$ , and  $A_{G \times G}$  is the sign matrix with entries  $A(x, y) = f(xy^{-1})$ , then

$$\text{rank}_{\pm}(A) \leq \eta(\|A\|_{\gamma_2}).$$

In the case of abelian  $G$ , we have

$$\text{rank}_{\pm}(A) \leq \exp(\exp(C\|A\|_{\gamma_2}^4)),$$

where  $C$  is a universal constant. Note that taking  $G = \mathbb{Z}_2^n$  corresponds to the class of XOR-lifts.

Equation (5) is also interesting from the point of view of communication complexity. It implies

$$\text{U}(A) \leq 2\text{D}^{eq}(A) + O(1).$$

where  $\text{U}(A)$  denotes the unbounded-error randomized communication complexity of  $A$ , formally defined in Equation (8).

- In Appendix B, we study the sign-rank of semi-algebraic sign matrices. In Corollary B.1, we prove that if  $A$  and  $B$  are two sign matrices of sign-rank at most  $d$ , then

$$\text{VC}(A \wedge B) \leq 20d \quad \text{and} \quad \text{m}^{\text{avg}}(A \wedge B)^{-1} \leq \text{rect}(A \wedge B)^{-1} \leq 2^{4d+4}.$$

These demonstrate the inability of the known lower bound techniques to give a negative answer to Question 1.11 by providing a super-constant lower bound on the sign-rank of semi-algebraic matrices.

- In Appendix C we prove that sign matrices of sign-rank  $d$  have small communication complexity in the average communication model over any product distribution.

## 2 Notation, Background, and Basic Observations

We will use the standard computer science asymptotic notations [11] of  $O(\cdot)$ ,  $\Omega(\cdot)$ ,  $\Theta(\cdot)$ ,  $o(\cdot)$ , and  $\omega(\cdot)$ . We denote the indicator function of a set  $S$  by  $\mathbf{1}_S$ , that is,  $\mathbf{1}_S(x) := 1$  if  $x \in S$ , and  $\mathbf{1}_S(x) := 0$  otherwise. For  $i = 1, \dots, d$ , we denote the  $i$ -th standard vector by  $\mathbf{e}_i \in \mathbb{R}^d$ . For a vector  $u \in \mathbb{R}^d$ , we denote the Euclidean norm of  $u$  by  $\|u\|$ .

For a real matrix  $B_{\mathcal{X} \times \mathcal{Y}}$ , we denote by  $\text{sgn}(B)$  the sign matrix corresponding to the signs of the entries of  $B$ . We say that the unit vectors  $u_x, v_y \in \mathbb{R}^d$  *sign-represent*  $A_{\mathcal{X} \times \mathcal{Y}}$  if  $A_{xy} = \text{sgn}(\langle u_x, v_y \rangle)$  for all  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .

A finite set of vectors  $v_1, \dots, v_m \in \mathbb{R}^d$  are in *isotropic position* if for every unit vector  $u \in \mathbb{R}^d$ , we have

$$\frac{1}{m} \sum_{i=1}^m |\langle u, v_i \rangle|^2 = \frac{1}{d}.$$

All matrices in this article are real and finite, and all normed spaces are defined over the reals. The spectral norm of a matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  is defined as

$$\|A\| = \max_{x \in \mathcal{X}, \|x\|=1} \|Ax\|,$$

and its trace norm is defined as

$$\|A\|_{\text{tr}} = \text{tr}(\sqrt{A^t A}) = \sum_{i=1}^{\min(|\mathcal{X}|, |\mathcal{Y}|)} \sigma_i,$$

where  $\sigma_i$  are the singular values of  $A$ . Next, we define the  $\gamma_2$  norm of a matrix, which is an important tool for proving lower and upper bounds in discrepancy theory and communication complexity [31].

► **Definition 2.1** ( $\gamma_2$  norm). *The  $\gamma_2$  norm of a matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , denoted by  $\|A\|_{\gamma_2}$ , is the smallest  $c \geq 0$  such that there exists  $d \in \mathbb{N}$  and vectors  $u_x, v_y \in \mathbb{R}^d$  with  $\max_{x,y} \|u_x\| \cdot \|v_y\| \leq c$  and  $A_{xy} = \langle u_x, v_y \rangle$  for all  $x, y$ .*

For  $\epsilon \in [0, 1)$ , the *approximate  $\gamma_2$  norm* of  $A_{\mathcal{X} \times \mathcal{Y}}$  with error parameter  $\epsilon$  is defined as

$$\|A\|_{\gamma_2, \epsilon} = \inf_B \|B\|_{\gamma_2},$$

where the infimum is over all real matrices  $B_{\mathcal{X} \times \mathcal{Y}}$  with  $\max_{x,y} |A_{xy} - B_{xy}| \leq \epsilon$ . Note that despite what the notation might suggest,  $\|\cdot\|_{\gamma_2, \epsilon}$  is not a norm.

By definition, a matrix  $B_{\mathcal{X} \times \mathcal{Y}}$  satisfies  $\|B\|_{\gamma_2} = 1$  if and only if for some  $d \in \mathbb{N}$ , there exist unit vectors  $u_x, v_y \in \mathbb{R}^d$  with  $B_{xy} = \langle u_x, v_y \rangle$  for all  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ . Hence, one can reformulate Definition 1.3 and Definition 1.6 in terms of the  $\gamma_2$  norm as

$$m(A) = \sup_{\substack{B: \|B\|_{\gamma_2}=1 \\ \text{sgn}(B)=A}} \min_{x,y} |B_{xy}|,$$

and

$$m_\nu^{\text{avg}}(A) = \sup_{\substack{B: \|B\|_{\gamma_2}=1 \\ \text{sgn}(B)=A}} \min_x \mathbb{E}_{y \sim \nu} |B_{xy}|.$$

Finally, note that the dual of the  $\gamma_2$  norm is

$$\|A\|_{\gamma_2^*} := \sup_{B: \|B\|_{\gamma_2}=1} \text{tr}(AB^t) = \sup_{B: \|B\|_{\gamma_2}=1} \sum_{x,y \in \mathcal{X} \times \mathcal{Y}} A_{xy} B_{xy}, \quad (6)$$

where both  $A$  and  $B$  are  $\mathcal{X} \times \mathcal{Y}$  matrices.

Note that for any matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  and unit vectors  $u_x, v_y \in \mathbb{R}^d$ , we have

$$\sum_{x,y} A_{xy} \langle u_x, v_y \rangle \leq \|A\| \sqrt{|\mathcal{X}| |\mathcal{Y}|}.$$

Therefore, by Equation (6), we have  $\|A\|_{\gamma_2^*} \leq \|A\| \sqrt{|\mathcal{X}| |\mathcal{Y}|}$ . Forster's original paper [15] shows  $\text{rank}_{\pm}(A) \geq \frac{\sqrt{|\mathcal{X}| |\mathcal{Y}|}}{\|A\|}$ . Later [31] improved this bound to  $\text{rank}_{\pm}(A) \geq \frac{|\mathcal{X}| |\mathcal{Y}|}{\|A\|_{\gamma_2^*}^2}$ . The following proposition, which is based on [31, 15], recovers these bounds, as  $\text{rank}_{\pm}(A) \geq m^{\text{avg}}(A)^{-1}$  by Theorem 1.7.

► **Proposition 2.2.** *For every sign-matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , we have*

$$m^{\text{avg}}(A)^{-1} \geq \frac{|\mathcal{X}| |\mathcal{Y}|}{\|A\|_{\gamma_2^*}^2} \geq \frac{\sqrt{|\mathcal{X}| |\mathcal{Y}|}}{\|A\|}.$$

### 3 Main Results

#### 3.1 Monochromatic rectangle ratio vs average margin

Our first theorem relates the monochromatic rectangle ratio of a sign matrix to its average margin. We omit the proof which is based on the minimax theorem due to space limitations.

► **Theorem 3.1.** *For every sign matrix  $A$ , we have*

$$m^{\text{avg}}(A)^{-1} \leq \text{rect}(A)^{-1}.$$

#### 3.2 Sign-rank vs. current lower bound methods

Our next theorem shows a significant limitation for the three discussed lower bound methods. It shows that there are matrices with polynomially large sign-rank, while neither of the known methods can yield super constant bounds.

► **Theorem 3.2 (Main Theorem).** *There exists  $n \times n$  sign matrices  $A$  with sign-rank  $\Omega(\frac{n^{1/3}}{\log(n)})$  that satisfy*

$$\text{VC}(A) \leq 2 \quad \text{and} \quad m^{\text{avg}}(A)^{-1} \leq \text{rect}(A)^{-1} \leq 2^{15}.$$

**Proof.** The idea is to construct a large collection of sign matrices, each with a large monochromatic rectangle ratio. The statement then would follow from the upper bound on the number of matrices of small sign-rank, presented in Lemma 1.1.

Let  $N$  be a positive integer, and consider the sets

$$\mathcal{P} = \{(x, y) \in \mathbb{Z}^2 : 1 \leq x \leq N, 1 \leq y \leq 2N^2\}$$

and

$$\mathcal{L} = \{(a, b) \in \mathbb{Z}^2 : 1 \leq a \leq N, 1 \leq b \leq 2N^2\}.$$

We think of the elements  $\ell = (a, b) \in \mathcal{L}$  as lines  $y = ax + b$  in  $\mathbb{R}^2$ , and we consider  $(x, y) \in \mathcal{P}$  as points in  $\mathbb{R}^2$ .

Define the sign matrix  $F_{\mathcal{L}, \mathcal{P}}$  by point line incidences:

$$F_{\ell, p} = \begin{cases} -1 & p \in \ell \\ 1 & p \notin \ell \end{cases}.$$

Set  $n = N^3$  and note that  $F$  is a  $2n \times 2n$  matrix, and for every  $\ell = (a, b)$  and  $p = (x, y)$ , we have

$$\begin{aligned} F_{\ell, p} &= \text{sgn} \left( (ax + b - y)^2 - \frac{1}{2} \right) \\ &= \text{sgn} \left( a^2 x^2 - 2axy + y^2 + 2abx - 2by + \left( b^2 - \frac{1}{2} \right) \right). \end{aligned}$$

Since each term in the last line corresponds to a rank 1 matrix, we have

$$\text{rank}_{\pm}(F) \leq 6.$$

Additionally,  $F$  has the following useful properties:

1. Since any two distinct lines have at most one point in common,  $F$  does not contain any  $2 \times 2$   $(-1)$ -monochromatic subrectangles.
2. Each line  $\ell = (a, b)$  with  $b \leq N^2$  goes through  $N = n^{1/3}$  points from  $\mathcal{P}$ . Consequently,  $F$  contains at least  $n^{4/3}$  negative entries.

Consider all  $2n \times 2n$  sign matrices  $A$  that can be obtained from  $F$  by changing the sign of a subset of the negative entries to positive. There are at least  $2^{n^{4/3}}$  such matrices. By Lemma 1.1, most such matrices  $A$  have sign-rank  $\Omega(n^{1/3}/\log n)$ . Let  $A$  be any such matrix obtained from  $F$ , so that  $\text{rank}_{\pm}(A) = \Omega(n^{1/3}/\log n)$ .

Since  $A$  is obtained from a submatrix of  $F$  by only changing its  $-1$  entries,  $A$  also satisfies the first property above. That is,  $A$  does not contain any  $2 \times 2$   $(-1)$ -monochromatic subrectangle, and consequently  $\text{VC}(A) \leq 2$  as desired.

We proceed to bounding the rectangle ratio and hence also the average margin of  $A$ . Let  $\mu \times \nu$  be any product distribution on  $\mathcal{L} \times \mathcal{P}$ . Since  $\text{rank}_{\pm}(F) \leq 6$ , by Theorem 1.10, there exists a monochromatic rectangle  $R$  of  $F$  with

$$\mu \times \nu(R) \geq 2^{-14}.$$

If  $R$  is a 1-monochromatic rectangle in  $F$ , then it is also a 1-monochromatic rectangle in  $A$ . On the other hand, if  $R$  is a  $(-1)$ -monochromatic rectangle in  $F$ , then by the first property above, it is either a  $1 \times k$  or a  $k \times 1$  rectangle for some  $k$ . In both cases  $R$  contains a subrectangle  $R' \subseteq R$  that is monochromatic in  $A$  and satisfies

$$\mu \times \nu(R') \geq \frac{\mu \times \nu(R)}{2} \geq 2^{-15}.$$

We conclude that

$$\text{rect}(A) \geq 2^{-15}.$$

Finally, by Theorem 3.1, we have  $\text{m}^{\text{avg}}(A)^{-1} \leq \text{rect}(A)^{-1} \leq 2^{15}$ . ◀



### 3.3 Does large margin imply small sign-rank?

Next, we discuss the relation between sign-rank and margin, namely Question 1.4 and Conjecture 1.5. We start with a short discussion of the equivalence of margin and several other complexity and analytic parameters associated with sign matrices. We have already mentioned the result of Linial and Shraibman [30] stating

$$\text{disc}(A) \leq m(A) \leq 8 \text{disc}(A).$$

Let  $R_\epsilon(A)$  denote the public-coin randomized communication complexity of the matrix  $A$  with two-sided error  $\epsilon$ . We refer the reader to [28] for a formal definition of this complexity measure. The following folklore proposition shows that for any fixed  $\epsilon \in (0, \frac{1}{2})$ , the gap between  $\text{disc}(A)^{-1}$  and  $R_\epsilon(A)$  is at most exponential.

► **Proposition 3.3** (folklore). *For every  $\epsilon \in (0, \frac{1}{2})$  and every sign-matrix  $A$ , we have*

$$\log((1 - 2\epsilon) \cdot \text{disc}(A)^{-1}) \leq R_\epsilon(A) \leq O\left(\log\left(\frac{1}{\epsilon}\right) \text{disc}(A)^{-2}\right). \quad (7)$$

By Proposition 3.3, one can equivalently consider  $R_\epsilon(A)$  instead of  $m(A)^{-1}$  in Question 1.4 and Conjecture 1.5. This is particularly interesting in light of the equivalence of the logarithm of sign-rank and the unbounded-error communication complexity  $U(A)$ , due to Paturi and Simon [36]:

$$U(A) := \lim_{\epsilon \nearrow \frac{1}{2}} R_\epsilon^{\text{prv}}(A) = \log(\text{rank}_\pm(A)) + O(1). \quad (8)$$

We refer the reader to [28] for the definition of the private-coin randomized communication complexity  $R_\epsilon^{\text{prv}}(A)$ .

Finally, let us discuss the equivalence to approximate  $\gamma_2$  norms. The following relationship with public-coin randomized communication complexity is known

$$\log \|A\|_{\gamma_2, \epsilon} \leq R_{\frac{\epsilon}{2}}(A) \leq O\left(\frac{\log(1/\epsilon)}{(1 - \epsilon)^2} \|A\|_{\gamma_2, \epsilon}^2\right), \quad (9)$$

where  $A$  is a sign matrix and  $\epsilon \in (0, 1)$ . The lower bound is from [31] and the upper bound is proven in [23, Corollary 2.8 (c)]. However, since those papers use a different notation, for the convenience of the reader, we provide a proof in Proposition D.2.

To summarize, for every fixed  $\epsilon \in (0, \frac{1}{2})$ , we have

$$m(A)^{-1} \approx \text{disc}(A)^{-1} \approx \|A\|_{\gamma_2, \epsilon} \approx R_\epsilon(A), \quad (10)$$

where the equivalence notation  $\approx$  means that each parameter can be bounded by applying a universal function (that could depend on  $\epsilon$ ) to the other parameter.

The following proposition shows that a positive answer to Conjecture 1.5 is beyond the reach of the current known lower bound techniques.

► **Proposition 3.4** (Barrier to Conjecture 1.5). *Let  $Q_d$  be the sign matrix defined in Conjecture 1.5. There exists a constant  $c$  such that for every  $d \in \mathbb{N}$ , we have*

$$m(Q_d)^{-1} \leq c,$$

and

$$\text{VC}(Q_d), m^{\text{avg}}(Q_d)^{-1}, \text{rect}(Q_d)^{-1} \leq c.$$

**Proof.** The bound  $m(Q_d)^{-1} = O(1)$  follows from the equivalence of the margin and the randomized communication complexity discussed above, and the fact that  $R_{1/3}(Q_d) = O(1)$ , due to [48]. Since  $\sqrt{VC(A)} \leq m^{\text{avg}}(Q_d)^{-1} \leq m(Q_d)^{-1}$ , it only remains to show  $\text{rect}(Q_d)^{-1} = O(1)$ .

Next, we will show how to bound  $\text{rect}(Q_d)^{-1}$ . Let  $\mathcal{X}$  and  $\mathcal{Y}$  be the set of odd-parity and even-parity elements of  $\{0, 1\}^d$  corresponding, respectively, to the rows and columns of  $Q_d$ . Let  $\mu$  and  $\nu$  be distributions, respectively, over  $\mathcal{X}$  and  $\mathcal{Y}$ . Recall that  $Q_d(x, y) = -1$  iff  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$  differ in exactly one coordinate. We will use the fact that  $Q_d$  does not contain any  $2 \times 3$  or  $3 \times 2$   $(-1)$ -monochromatic rectangles, which also directly implies  $VC(Q_d) \leq 3$ . We will consider two cases.

**Case 1.** Suppose

$$\Pr_{x \sim \mu, y \sim \nu} [Q_d(x, y) = -1] \geq c := 1/2.$$

Applying Jensen's inequality twice, we have

$$\begin{aligned} c^6 &\leq (\mathbb{E}_{x \sim \mu, y \sim \nu} [\mathbf{1}_{Q_d(x, y) = -1}])^6 \leq (\mathbb{E}_{x \sim \mu} (\mathbb{E}_{y \sim \nu} [\mathbf{1}_{Q_d(x, y) = -1}])^3)^2 \\ &= \left( \mathbb{E}_{x \sim \mu} \mathbb{E}_{y_1, y_2, y_3 \sim \nu} \left[ \prod_j \mathbf{1}_{Q_d(x, y_j) = -1} \right] \right)^2 \\ &\leq \mathbb{E}_{y_1, y_2, y_3 \sim \nu} \left( \mathbb{E}_{x \sim \mu} \left[ \prod_{j=1}^3 \mathbf{1}_{Q_d(x, y_j) = -1} \right] \right)^2 \\ &= \mathbb{E}_{x_1, x_2 \sim \mu, y_1, y_2, y_3 \sim \nu} \left[ \prod_{i,j} \mathbf{1}_{Q_d(x_i, y_j) = -1} \right]. \end{aligned}$$

The last term is the probability that the random rectangle  $\{x_1, x_2\} \times \{y_1, y_2, y_3\}$  is a  $(-1)$ -monochromatic rectangle of  $Q_d$ . Since  $Q_d$  does not contain any  $2 \times 3$   $(-1)$ -monochromatic rectangle, we must have

$$\Pr[x_1 = x_2 \vee |\{y_1, y_2, y_3\}| \leq 2] \geq c^6.$$

Therefore, one of the two distributions  $\mu$  or  $\nu$  has noticeable collision probability. Specifically, either  $\Pr_{x, x' \sim \mu} [x = x'] \geq c^6/4$  or  $\Pr_{y, y' \sim \nu} [y = y'] \geq \frac{1}{3} \Pr[|\{y_1, y_2, y_3\}| \leq 2] \geq c^6/4$ . Without loss of generality, assume that the former is true. In this case

$$\Pr[x = x'] = \sum_{a \in \mathcal{X}} \Pr[x = a]^2 \leq \max_{a \in \mathcal{X}} \Pr[x = a].$$

Therefore, there is an  $a \in \mathcal{X}$  such that  $\Pr[x = a] \geq c^6/8$ . Now, note that the  $a$ 'th row of  $Q_d$  either has a  $\mu \times \nu$ -measure of at least  $c^6/16$  on its  $(-1)$ 's or on its  $1$ 's.

**Case 2.** If Case 1 does not hold, then

$$\Pr_{x \sim \mu, y \sim \nu} [|x - y|_1 \geq 3] \geq 1/2, \tag{11}$$

where  $|x - y|_1$  denotes the Hamming distance between  $x$  and  $y$ . For a subset  $S \subseteq [d]$ , let  $\phi_S : \{0, 1\}^d \rightarrow \{0, 1, 2, 3\}$  be defined as  $\phi_S(x) = \sum_{i \in S} x_i \pmod 4$ .

## 22:14 Lower Bound Methods for Sign-Rank and Their Limitations

For  $x, y \in \{0, 1\}^d$  satisfying  $|x - y|_1 \geq 3$ , let  $j_1, j_2, j_3$  be distinct indices where they differ. Pick  $S \subseteq [d]$  uniformly at random by first picking a random subset  $S_1 \subseteq [d] \setminus \{j_1, j_2, j_3\}$  and then taking its union with a random  $S_2 \subseteq \{j_1, j_2, j_3\}$ . For every choice of  $S_1$ , there exists at least one choice of  $S_2$  such that  $|\phi_S(x) - \phi_S(y)| = 2$ . Therefore,

$$\Pr_S [|\phi_S(x) - \phi_S(y)| = 2] \geq 1/8.$$

Combining with Equation (11), we have

$$\Pr_{\substack{S \\ x \sim \mu, y \sim \nu}} [|\phi_S(x) - \phi_S(y)| = 2] \geq \Pr_S [|\phi_S(x) - \phi_S(y)| = 2 \mid |x - y|_1 \geq 3] \Pr_{x \sim \mu, y \sim \nu} [|x - y|_1 \geq 3] \geq 1/16. \quad (12)$$

Hence, there is a choice of  $S \subseteq [d]$  such that

$$\Pr_{x \sim \mu, y \sim \nu} [|\phi_S(x) - \phi_S(y)| = 2] \geq 1/16.$$

Hence, there exist  $r, t \in \{0, 1, 2, 3\}$  with  $|r - t| = 2$  such that

$$\Pr_{x \sim \mu, y \sim \nu} [\phi_S(x) = r \text{ and } \phi_S(y) = t] \geq 2^{-8}.$$

In this case, the set  $\{x \mid \phi_S(x) = r\} \times \{y \mid \phi_S(y) = t\}$  is a 1-monochromatic rectangle of measure at least  $2^{-8}$ .  $\blacktriangleleft$

In light of Proposition 3.4 it might seem worthwhile to seek a different candidate sign matrix for establishing a negative answer to Question 1.4. By Proposition 1.8 and the definition of average margin, for every sign matrix  $A$ , we have

$$\sqrt{\text{VC}(A)} \leq m^{\text{avg}}(A)^{-1} \leq m(A)^{-1}, \quad (13)$$

and thus Forster's method and the VC dimension method cannot imply a negative answer to Question 1.4. Therefore,  $\text{rect}(A)^{-1}$  remains the only known approach.

The following conjecture of Chattopadhyay, Lovett, and Vinyals [8, Problem 6.1] (see also [23, Conjecture I]), if true, would imply that  $\text{rect}(A)^{-1}$  is also small if  $m(A)^{-1}$  is small.

► **Conjecture 3.5** (Chattopadhyay, Lovett, Vinyals [8]). *There exists a function  $\eta$  such that every sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  contains an  $\frac{|\mathcal{X}|}{k} \times \frac{|\mathcal{Y}|}{k}$  monochromatic rectangle for  $k \leq \eta(m(A)^{-1})$ .*

Conjecture 3.5 is in fact equivalent to the existence of a function  $\eta$  such that every sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , we have

$$\text{rect}(A)^{-1} \leq \eta(m(A)^{-1}).$$

In particular, assuming Conjecture 3.5, even  $\text{rect}(A)^{-1}$  cannot be used towards giving a negative answer to Question 1.4.

### 3.4 Communication Complexity with Equality Oracle

In Section 3.3, we showed that Question 1.4 can be formulated in terms of the approximate  $\gamma_2$  norm: Is it true that for sign matrices,  $\|A\|_{\gamma_2, \epsilon} = O(1)$  implies  $\text{rank}_{\pm}(A) = O(1)$ ? As we mentioned in Conjecture 1.5, we believe that the answer to this question is negative. However, it seems plausible that such a statement could hold if we strengthen the assumption by replacing the approximate  $\gamma_2$  norm with the  $\gamma_2$  norm:

► **Conjecture 3.6** (Conjecture 1.14 restated). *There exists a function  $\eta$  such that for every sign matrix  $A$ , we have  $\text{rank}_{\pm}(A) \leq \eta(\|A\|_{\gamma_2})$ .*

Zero-one valued matrices that satisfy  $\|A\|_{\gamma_2} = O(1)$  are important in operator theory as they correspond to the bounded idempotents of the algebra of Schur multipliers. Inspired by Cohen's idempotent theorem in harmonic analysis, a characterization of these matrices was conjectured in [23]. To state this conjecture, we need to introduce the notion of a blocky matrix. We call a zero-one valued matrix  $M_{\mathcal{X} \times \mathcal{Y}}$  *blocky* if

$$\{(x, y) \mid M_{xy} = 1\} = \bigcup_i \mathcal{X}_i \times \mathcal{Y}_i,$$

for disjoint sets  $\mathcal{X}_i \subseteq \mathcal{X}$  and disjoint sets  $\mathcal{Y}_i \subseteq \mathcal{Y}$ . A simple example of a blocky matrix is the *identity matrix*. Note that every blocky matrix can be obtained from the identity matrix by duplicating rows and columns and adding all zero rows and columns. Since the  $\gamma_2$  norm is invariant under these operations, every non-zero blocky matrix  $M$  satisfies  $\|M\|_{\gamma_2} = 1$ . It is shown in [32] that blocky matrices are precisely the set of Boolean matrices with  $\|M\|_{\gamma_2} \leq 1$ .

Blocky matrices are related to deterministic communication complexity with access to an equality oracle. In this model, a protocol for a sign matrix  $A$  corresponds to a binary tree. Each non-leaf node  $v$  in the tree corresponds to a query to  $eq(a_v(x), b_v(y))$ , where  $eq(a, b) = 1$  if  $a = b$  and  $-1$  otherwise. Note that  $a_v(x)$  and  $b_v(y)$  can be computed, respectively, by the first and the second party in the communication protocol. Every input  $(x, y)$  naturally corresponds to a path from the root of the tree to a leaf, and it is required that the leaf is labeled with the correct value  $A_{xy}$ . The cost of the protocol is the depth of the tree. The deterministic communication complexity of the matrix  $A$  with access to an equality oracle, denoted by  $D^{eq}(A)$ , is the smallest depth of such a protocol for  $A$ .

Note that for any two functions  $a(x)$  and  $b(y)$ , the function  $(x, y) \mapsto eq(a(x), b(y))$  corresponds to an  $\mathcal{X} \times \mathcal{Y}$  blocky matrix as its 1's consist of a union of row-disjoint and column-disjoint rectangles.

► **Conjecture 3.7** ([23, Conjecture III]). *For every sign-matrix  $A$ , if  $\|A\|_{\gamma_2} = O(1)$ , then  $A$  can be expressed as a  $\pm 1$ -linear combination of  $O(1)$  blocky matrices, equivalently  $D^{eq}(A) = O(1)$ .*

The following theorem shows that if Conjecture 3.7 is true, then the answer to Conjecture 1.14 is positive.

► **Theorem 3.8.** *For every sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , we have*

$$\text{rank}_{\pm}(A) \leq 4^{D^{eq}(A)}.$$

**Proof.** We proceed by induction on  $d := D^{eq}(A)$ . When  $d = 1$ ,  $A$  corresponds to a blocky matrix, which in fact has  $\text{rank}_{\pm}(A) \leq 3$ . For larger  $d$ , consider a cost  $d$  protocol for a sign matrix  $A$  and suppose the equality query at the root of tree is  $eq(a(x), b(y))$ . We may assume without loss of generality that  $a(x), b(y) \in \mathbb{N}$ . Let  $S_{\mathcal{X} \times \mathcal{Y}}$  be the matrix with entries  $S_{xy} = \mathbf{1}_{a(x)=b(y)}$ . We branch according to the output of the first query either to the left or the right subtree of the root, each corresponding to a protocol of cost at most  $d - 1$ . Let the corresponding matrices for these protocols be  $\Pi_1$  and  $\Pi_2$ , and note that

$$A = S \circ \Pi_1 + (\mathbf{J} - S) \circ \Pi_2,$$

where  $\mathbf{J} := \mathbf{J}_{\mathcal{X} \times \mathcal{Y}}$  is the all-ones matrix. By the induction hypothesis,  $\Pi_1$  and  $\Pi_2$  have sign-rank at most  $\leq 4^{d-1}$ . Let  $\tilde{\Pi}_1$  and  $\tilde{\Pi}_2$  be real matrices with rank at most  $4^{d-1}$  that satisfy  $\text{sgn}(\tilde{\Pi}_1) = \Pi_1$  and  $\text{sgn}(\tilde{\Pi}_2) = \Pi_2$ . Let  $E_{\mathcal{X} \times \mathcal{Y}}$  be the rank-3 matrix with entries  $E_{xy} = (a(x) - b(y))^2$ . Note that for a sufficiently large  $k$ , we have

$$A = \text{sgn}(\tilde{\Pi}_1 + kE \circ \tilde{\Pi}_2).$$

Finally, we have

$$\text{rank}(\tilde{\Pi}_1 + k\tilde{\Pi}_2 \circ E) \leq \text{rank}(\tilde{\Pi}_1) + \text{rank}(\tilde{\Pi}_2) \cdot \text{rank}(E) \leq 4^{d-1} + 3 \cdot 4^{d-1} = 4^d. \quad \blacktriangleleft$$

Conjecture 3.7 is inspired by quantitative versions of Cohen's seminal idempotent theorem in harmonic analysis, developed by Green and Sanders [21, 22] and Sanders [39, 38, 40]. As it is noticed in [23], these theorems verify Conjecture 3.7 for a large natural class of matrices: sign matrices  $A_{G \times G}$  where  $G$  is a finite group and the entries are defined as  $A_{xy} = f(xy^{-1})$  for some  $f : G \rightarrow \{-1, 1\}$ . Note that taking  $G = \mathbb{Z}_2^n$  corresponds to the class of XOR-lifts, which is a well studied class of functions in communication complexity. The following corollary is proved by combining these results with Theorem 3.8 to verify Conjecture 1.14 for this class of matrices.

► **Corollary 3.9.** *There exists a function  $\eta$  such that the following holds. If  $f : G \rightarrow \{-1, 1\}$  for a finite group  $G$ , and  $A_{G \times G}$  is the sign matrix with entries  $A(x, y) = f(xy^{-1})$ , then*

$$\text{rank}_{\pm}(A) \leq \eta(\|A\|_{\gamma_2}).$$

In the case of abelian  $G$ , we have

$$\text{rank}_{\pm}(A) \leq \exp(\exp(C\|A\|_{\gamma_2}^4)),$$

## 4 Concluding remarks

In light of the results in the present paper, the following open problem captures the limitation of the currently known lower bound techniques for sign-rank.

► **Problem 4.1.** *Construct an explicit sequence of matrices  $A_n$  such that  $\text{rect}(A_n)^{-1} = O(1)$  and*

$$\lim_{n \rightarrow \infty} \text{rank}_{\pm}(A_n) = \infty.$$

By Theorem 3.2, we know such sequences of matrices exist. On the other hand, by Theorem 3.1 and Proposition 1.8, we have

$$\sqrt{\text{VC}(A)} \leq m^{\text{avg}}(A)^{-1} \leq \text{rect}^{-1}(A),$$

and thus none of the known lower bound techniques are capable of solving Problem 4.1. Note that a positive answer to Conjecture 1.5 would solve Problem 4.1.

Finally, let us mention that it is unclear whether the proof of Proposition B.3 can be generalized to infinite matrices, which raises the following intriguing question.

► **Question 4.2.** *Is the sign-rank of an infinite sign matrix  $A_{\mathbb{N} \times \mathbb{N}}$  finite if the sign-rank of every finite submatrix of  $A_{\mathbb{N} \times \mathbb{N}}$  is bounded by a fixed constant  $d$ ?*

## References

- 1 Noga Alon, Peter Frankl, and Vojtech Rödl. Geometrical realization of set systems and probabilistic communication complexity. In *26th Annual Symposium on Foundations of Computer Science, FOCS 1985*, pages 277–280. IEEE Computer Society, 1985.
- 2 Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank versus VC dimension. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016*, volume 49, pages 47–80, 2016.
- 3 Noga Alon, János Pach, Rom Pinchasi, Radoš Radoičić, and Micha Sharir. Crossing patterns of semi-algebraic sets. *Journal of Combinatorial Theory, Series A*, 111(2):310–326, 2005.
- 4 Franck Barthe. On a reverse form of the Brascamp-Lieb inequality. *Invent. Math.*, 134(2):335–361, 1998.
- 5 Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. Limitations of learning via embeddings in Euclidean half spaces. *J. Mach. Learn. Res.*, 3(Spec. Issue Comput. Learn. Theory):441–461, 2002.
- 6 Mark Bun, Nikhil S. Mande, and Justin Thaler. Sign-rank can increase under intersection. *ACM Trans. Comput. Theory*, 13(4):Art. 24, 17, 2021. doi:10.1145/3470863.
- 7 Mark Bun and Justin Thaler. Improved Bounds on the Sign-Rank of AC0. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:14, 2016.
- 8 Arkadev Chattopadhyay, Shachar Lovett, and Marc Vinyals. Equality alone does not simulate randomness. In *34th Computational Complexity Conference (CCC 2019)*, 2019.
- 9 Arkadev Chattopadhyay and Nikhil S. Mande. Separation of unbounded-error models in multi-party communication complexity. *Theory Comput.*, 14(1):1–23, 2018.
- 10 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
- 11 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA; McGraw-Hill Book Co., Boston, MA, second edition, 2001.
- 12 Marek Eliáš, Jiří Matoušek, Edgardo Roldán-Pensado, and Zuzana Safernová. Lower bounds on geometric Ramsey functions. *SIAM J. Discrete Math.*, 28(4):1960–1970, 2014.
- 13 Vitaly Feldman. A general characterization of the statistical query complexity. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017*, volume 65 of *Proceedings of Machine Learning Research*, pages 785–830, 2017.
- 14 Vitaly Feldman, Cristóbal Guzmán, and Santosh Vempala. Statistical query algorithms for mean vector estimation and stochastic convex optimization. *Math. Oper. Res.*, 46(3):912–945, 2021.
- 15 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. System Sci.*, 65(4):612–625, 2002. Special issue on complexity, 2001 (Chicago, IL). doi:10.1016/S0022-0000(02)00019-3.
- 16 Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *FST TCS 2001: Foundations of software technology and theoretical computer science*, volume 2245, pages 171–182. Springer, 2001.
- 17 Jürgen Forster and Hans Ulrich Simon. On the smallest possible dimension and the largest possible margin of linear arrangements representing given concept classes. *Theoret. Comput. Sci.*, 350(1):40–48, 2006. doi:10.1016/j.tcs.2005.10.015.
- 18 Jacob Fox, Mikhail Gromov, Vincent Lafforgue, Assaf Naor, and János Pach. Overlap properties of geometric expanders. *J. Reine Angew. Math.*, 671:49–83, 2012.
- 19 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz’s problem. *J. Eur. Math. Soc. (JEMS)*, 19(6):1785–1810, 2017.
- 20 Jacob Fox, János Pach, and Andrew Suk. A polynomial regularity lemma for semialgebraic hypergraphs and its applications in geometry and property testing. *SIAM Journal on Computing*, 45(6):2199–2223, 2016. doi:10.1137/15M1007355.

- 21 Ben Green and Tom Sanders. Boolean functions with small spectral norm. *Geometric and Functional Analysis*, 18(1):144–162, 2008. doi:10.1007/s00039-008-0654-y.
- 22 Ben Green and Tom Sanders. A quantitative version of the idempotent theorem in harmonic analysis. *Ann. of Math. (2)*, 168(3):1025–1054, 2008.
- 23 Lianna Hambardzumyan, Hamed Hatami, and Pooya Hatami. Dimension-free bounds and structural results in communication complexity. *Israel J. Math.*, 2021. To appear.
- 24 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Sign rank vs discrepancy. In *35th Computational Complexity Conference*, volume 169 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Art. No. 18, 14. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 25 Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1034–1044. IEEE Computer Society, 2020.
- 26 Michael Kallweit and Hans Ulrich Simon. A close look to margin complexity and related parameters. In Sham M. Kakade and Ulrike von Luxburg, editors, *COLT 2011 - The 24th Annual Conference on Learning Theory*, volume 19 of *JMLR Proceedings*, pages 437–456. JMLR.org, 2011.
- 27 Adam R. Klivans and Alexander A. Sherstov. Unconditional lower bounds for learning intersections of halfspaces. *Mach. Learn.*, 69(2-3):97–114, 2007.
- 28 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.
- 29 Nati Linial, Shahar Mendelson, Gideon Schechtman, and Adi Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007.
- 30 Nati Linial and Adi Shraibman. Learning complexity vs. communication complexity. *Combin. Probab. Comput.*, 18(1-2):227–245, 2009.
- 31 Nati Linial and Adi Shraibman. Lower bounds in communication complexity based on factorization norms. *Random Structures & Algorithms*, 34(3):368–394, 2009.
- 32 Leo Livshits. A note on 0-1 Schur multipliers. *Linear Algebra Appl.*, 222:15–22, 1995. doi:10.1016/0024-3795(93)00268-5.
- 33 Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel J. Math.*, 93:333–344, 1996.
- 34 J. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:275–280, 1964. doi:10.2307/2034050.
- 35 Assaf Naor. Metric dimension reduction: a snapshot of the Ribe program. In *Proceedings of the International Congress of Mathematicians—Rio de Janeiro 2018. Vol. I. Plenary lectures*, pages 759–837. World Sci. Publ., Hackensack, NJ, 2018.
- 36 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *Journal of Computer and System Sciences*, 33(1):106–123, 1986.
- 37 Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of  $AC^0$ . *SIAM J. Comput.*, 39(5):1833–1855, 2010. doi:10.1137/080744037.
- 38 Tom Sanders. A quantitative version of the non-abelian idempotent theorem. *Geom. Funct. Anal.*, 21(1):141–221, 2011.
- 39 Tom Sanders. Boolean functions with small spectral norm, revisited. *Math. Proc. Cambridge Philos. Soc.*, 167(2):335–344, 2019.
- 40 Tom Sanders. Bounds in Cohen’s idempotent theorem. *J. Fourier Anal. Appl.*, 26(2):Paper No. 25, 64, 2020.
- 41 Alexander A. Sherstov. The unbounded-error communication complexity of symmetric functions. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS ’08, pages 384–393, 2008.
- 42 Alexander A. Sherstov. Halfspace matrices. *Comput. Complexity*, 17(2):149–178, 2008.
- 43 Alexander A. Sherstov and Pei Wu. Near-optimal lower bounds on the threshold degree and sign-rank of  $AC^0$ . In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 401–412, New York, NY, USA, 2019. Association for Computing Machinery.



- 44 Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *Learning theory*, volume 3559 of *Lecture Notes in Comput. Sci.*, pages 545–560. Springer, Berlin, 2005.
- 45 Andrew Suk. Semi-algebraic Ramsey numbers. *J. Combin. Theory Ser. B*, 116:465–483, 2016. doi:10.1016/j.jctb.2015.10.001.
- 46 René Thom. Sur l’homologie des variétés algébriques réelles. In *Differential and Combinatorial Topology (A Symposium in Honor of Marston Morse)*, pages 255–265. Princeton Univ. Press, Princeton, N.J., 1965.
- 47 Hugh E. Warren. Lower bounds for approximation by nonlinear manifolds. *Trans. Amer. Math. Soc.*, 133:167–178, 1968.
- 48 Zhiqiang Zhang and Yaoyun Shi. Communication complexities of symmetric XOR functions. *Quantum Inf. Comput.*, 9(3-4):255–263, 2009.

## A

 Small sign-rank implies large monochromatic rectangles

In this section, we provide a short and robust geometric argument for the fact that sign matrices of small sign-rank contain large monochromatic rectangles. Our proof is quite different from the proof of [20], which is based on the divide-and-conquer cutting lemma of Chazelle [10]. However, we note that our bound is slightly weaker than the  $\frac{n}{2^{O(d \log d)}} \times \frac{n}{O(1)}$  bound of [20].

► **Proposition A.1.** *There exists a constant  $c > 0$  such that the following holds. Every sign matrix  $A_{n \times n}$  with sign-rank  $d$  contains a monochromatic rectangle of size*

$$\frac{n}{2^{cd \log d}} \times \frac{n}{4d}.$$

**Proof.** Let  $S^{d-1}$  denote the unit sphere in  $\mathbb{R}^d$ . Consider a sign representation of  $A$  with unit vectors  $u_i, v_j \in \mathbb{R}^d$ . Without loss of generality, we can assume that the  $v_j$ ’s are in isotropic position.

For every  $u \in S^{d-1}$ , consider the spherical cap of height  $\alpha := \frac{1}{\sqrt{2d}}$ , defined as

$$C_u = \left\{ x \in S^{d-1} : \langle u, x \rangle \geq \sqrt{1 - \alpha^2} \right\},$$

and the equator region

$$E_u = \left\{ x \in S^{d-1} : |\langle u, x \rangle| \leq \alpha \right\}.$$

Note that the sets

$$R_u^+ := \{i : u_i \in C_u\} \times \{j : v_j \notin E_u, \langle v_j, u \rangle > 0\}$$

and

$$R_u^- := \{i : u_i \in C_u\} \times \{j : v_j \notin E_u, \langle v_j, u \rangle < 0\}$$

correspond, respectively, to a (+1)-monochromatic and a (−1)-monochromatic rectangle in  $A$ .

Since the  $v_j$ ’s are in isotropic position, for every  $u \in S^{d-1}$ , we have

$$\sum_{j=1}^n \langle u, v_j \rangle^2 = \frac{n}{d}.$$



On the other hand

$$\sum_{j: v_j \in E_u} \langle u, v_j \rangle^2 \leq n\alpha^2 = \frac{n}{2d},$$

which shows

$$|\{j : v_j \notin E_u\}| \geq \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d}.$$

In particular

$$|\{j : v_j \notin E_u, \langle v_j, u \rangle > 0\}| \geq \frac{n}{4d} \quad \text{or} \quad |\{j : v_j \notin E_u, \langle v_j, u \rangle < 0\}| \geq \frac{n}{4d}.$$

To estimate the surface area of  $C_u$ , recall that the surface area of the  $d$ -dimensional sphere of radius  $r$  is given by

$$A_d(r) := \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1} = \int_{-1}^1 A_{d-1}(\sqrt{1-h^2}) dh = \frac{2\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})} \int_{-1}^1 (\sqrt{1-h^2})^{d-2} dh.$$

Hence the ratio between the surface area of  $C_u$  and the whole sphere  $S^{d-1}$  can be estimated as

$$\begin{aligned} \frac{|C_u|}{A_d(1)} &= \frac{\int_{\sqrt{1-\alpha^2}}^1 (\sqrt{1-h^2})^{d-2} dh}{\int_{-1}^1 (\sqrt{1-h^2})^{d-2} dh} \geq \frac{\int_{\sqrt{1-\alpha^2}}^{\sqrt{1-\frac{\alpha^2}{4}}} (\sqrt{1-h^2})^{d-2} dh}{\int_{-1}^1 1 dh} \\ &\geq \frac{\sqrt{1-\frac{\alpha^2}{4}} - \sqrt{1-\alpha^2}}{2} \times (\alpha/2)^{d-2} = 2^{-O(d \log d)}. \end{aligned}$$

Picking a  $u \in S^{d-1}$  uniformly at random, with positive probability, one of the rectangles  $R_u^+$  or  $R_u^-$  satisfies the assertion of the theorem.  $\blacktriangleleft$

## B Sign-rank of Semi-algebraic matrices, an open problem

We start by discussing why Question 1.11, Question 1.12, and Question 1.13 are equivalent. Recall that a  $d$ -dimensional *semi-algebraic* set of description complexity  $t$  is of the form

$$\{y \in \mathbb{R}^d : \Gamma(\mathbf{1}_{p_1(y) \geq 0}, \dots, \mathbf{1}_{p_t(y) \geq 0}) = 1\}.$$

for a predicate  $\Gamma : \{0, 1\}^t \rightarrow \{0, 1\}$  and polynomials  $p_1, \dots, p_t$  on  $d$  variables.

**Proof of Equivalence of Question 1.11 and Question 1.12.** Clearly, Question 1.12 is a special case of Question 1.11. In order to prove the nontrivial direction of this equivalence, consider a semi-algebraic sign-matrix  $A$  defined by points  $u_1, \dots, u_m \in \mathbb{R}^d$  and semi-algebraic sets  $K_1, \dots, K_n \subseteq \mathbb{R}^d$ , each with description complexity  $t$ . Note that there are only  $2^{2^t}$  different possible predicates  $\{0, 1\}^t \rightarrow \{0, 1\}$ , and hence in Question 1.11, we can assume without loss of generality that all the sets  $K_i$  are defined using the same predicate  $\Gamma : \{0, 1\}^t \rightarrow \{0, 1\}$ .

Let  $p \in \mathbb{R}[x_1, \dots, x_d]$  be a polynomial of degree  $t$ . Let  $I_{d,t}$  denote the set of all  $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{Z}_{\geq 0}^d$  with  $\sum_{i=1}^d \alpha_i \leq t$ . The monomials of degree at most  $t$  in variables  $x_1, \dots, x_d$  are indexed by  $\alpha \in I_{d,t}$  with the correspondence  $x^\alpha = x_1^{\alpha_1} \dots x_d^{\alpha_d}$ . Note that every polynomial  $p(x) = \sum_{\alpha \in I_{d,t}} a_\alpha x^\alpha$  of degree at most  $t$  corresponds to an inner product

$$p(x) = \langle \Psi_t(p), \Phi_t(x) \rangle,$$

where  $\Psi_t(p) := (a_\alpha)_{\alpha \in I_{d,t}} \in \mathbb{R}^{|I_{d,t}|}$  and  $\Phi_t(x) := (x^\alpha)_{\alpha \in I_{d,t}} \in \mathbb{R}^{|I_{d,t}|}$ . Applying this linearization idea to all the defining polynomials of the semi-algebraic sets allows us to view the matrix  $A$  as a single predicate applied to a collection of sign matrices, each of sign-rank at most  $|I_{d,t}|$  each.  $\blacktriangleleft$

Hence, Question 1.11, Question 1.12, and Question 1.13 are all equivalent. Question 1.13, in particular, has a simple statement. Regarding this formulation, Bun, Mande, and Thaler [6] used Forster's method to show the existence of matrices  $A$  and  $B$  of sign-rank  $d$  such that  $\text{rank}_\pm(A \wedge B) \geq 2^{\Omega(\log^2 d)}$ . However, the following corollary of Theorem 3.1 shows that neither of the known methods can imply a negative answer to Question 1.13.

► **Corollary B.1** (Corollary to Theorem 3.1). *If  $A$  and  $B$  are two  $m \times n$  sign matrices of sign-rank at most  $d$ , then*

$$\text{VC}(A \wedge B) \leq 20d \quad \text{and} \quad m^{\text{avg}}(A \wedge B)^{-1} \leq \text{rect}(A \wedge B)^{-1} \leq 2^{4d+4}.$$

### Intersections of Half-spaces

The problem of bounding the sign-rank of  $A \wedge B$  is closely related to bounding the sign-rank of the matrices that are defined by points and intersections of pairs of half-spaces. For distinct  $y, y' \in \mathbb{R}^d$ , let  $I_{y,y'} = \{z \mid \langle y, z \rangle > 0\} \cap \{z \mid \langle y', z \rangle > 0\} \subset \mathbb{R}^d$  denote the intersection of the two half-spaces defined by  $y$  and  $y'$ , respectively. We refer to these sets as half-space intersections. Given a finite set of points  $\mathcal{X} \subseteq \mathbb{R}^d$  and a finite set of half-space intersections  $\mathcal{I}$  in  $\mathbb{R}^d$ , define the matrix  $F_{\mathcal{X} \times \mathcal{I}}$  as

$$F_{x,I} = \begin{cases} 1 & x \in I \\ -1 & x \notin I \end{cases}.$$

Is the sign-rank of  $F$  bounded by a constant  $c_d$ ? Note that for  $x \in \mathcal{X}$  and  $I_{y,y'} \in \mathcal{I}$ , we have  $F_{x,I} = \text{sgn} \langle x, y \rangle \wedge \text{sgn} \langle x, y' \rangle$ , and thus  $F$  can be expressed as the  $\wedge$  of two sign matrices of sign-rank  $d$ . Consequently, such a constant  $c_d$  exists if the answer to Question 1.13 regarding the sign-rank of  $A \wedge B$  is positive.

It turns out that the opposite direction is also true, but with a slight increase in the value of  $d$ .

▷ **Claim B.2.** If the constant  $c_{2d-1}$  exists, then for sign matrices  $A$  and  $B$  with  $\text{rank}_\pm(A) \leq d$  and  $\text{rank}_\pm(B) \leq d$ , we have  $\text{rank}_\pm(A \wedge B) = O(c_{2d-1})$ .

It is communicated to us by Shay Moran that it is known that the matrix  $F$  defined by half-space intersections in  $\mathbb{R}^3$  has bounded sign-rank. We omit the proof in the present version.

► **Proposition B.3** (Communicated by Shay Moran). *There exists a constant  $c_3$  such that given a finite set  $\mathcal{X}$  of points  $x \in \mathbb{R}^3$  and a finite set  $\mathcal{I}$  of half-space intersections  $I_{y,y'}$  in  $\mathbb{R}^3$ , the matrix  $F_{\mathcal{X} \times \mathcal{I}}$  with entries*

$$F_{x,I} = \begin{cases} 1 & x \in I \\ -1 & x \notin I \end{cases}$$

*satisfies  $\text{rank}_\pm(F) \leq c_3$ .*

Proposition B.3 combined with Claim B.2 implies the following special case of Question 1.13 for sign matrices of sign-rank at most 2.

► **Corollary B.4.** *There is a constant  $c > 0$  such that for every two sign matrices  $B_{m \times n}$  and  $C_{m \times n}$  with sign-rank at most 2, we have  $\text{rank}_\pm(B \wedge C) \leq c$ .*

## C Average Communication Complexity

In this section, we observe a simple connection between sign-rank and another model of communication complexity, average communication complexity. For any distribution  $\mu$  over  $\mathcal{X} \times \mathcal{Y}$ , let  $\text{CC}_\mu^{\text{avg}}(A)$  be the smallest expected communication complexity of a deterministic protocol that computes  $A$  correctly on all inputs. Moreover, define

$$\text{CC}^{\text{avg}}(A) = \sup_{\mu} \text{CC}_\mu^{\text{avg}}(A),$$

where  $\mu$  ranges over all *product* distributions over  $\mathcal{X} \times \mathcal{Y}$ .

► **Proposition C.1.** *For every sign-matrix  $A_{\mathcal{X} \times \mathcal{Y}}$ , we have*

$$\text{CC}^{\text{avg}}(A) \leq 2 \text{rect}(A)^{-1}.$$

**Proof.** Let  $\mu$  be any distribution on  $\mathcal{X} \times \mathcal{Y}$  and let  $\delta = \text{rect}(A)$ . By definition,  $A$  has a monochromatic rectangle  $R = S \times T$  such that  $\mu(R) \geq \delta$ . The two parties recursively proceed as follows. Given  $x$  and  $y$  as inputs, after communicating the two bits  $\mathbf{1}_{x \in S}$  and  $\mathbf{1}_{y \in T}$ , they can agree on whether  $(x, y) \in R$ . At which point, they have reduced their search to one of the four matrices  $A_{S \times T}$ ,  $A_{S^c \times T}$ ,  $A_{S \times T^c}$ , and  $A_{S^c \times T^c}$ . Note that in the first case, both parties know the answer and can conclude the protocol. In all the other three cases, the  $\mu$ -measure of the search-space has been reduced to at most  $1 - \delta$ , and they can recurse on the resulting submatrix according to the same protocol applied to  $\mu$  conditioned on the submatrix.

For a distribution  $\mu$ , let  $c_\mu$  denote the average cost of the above protocol, and let  $\mu$  be the maximizer for  $c_\mu$ . Let  $\mu_1, \mu_2, \mu_3$  denote  $\mu$  conditioned on  $S \times T^c$ ,  $S^c \times T^c$ , and  $S^c \times T$  respectively. We have

$$c_\mu \leq 2 \Pr[(x, y) \in R] + \Pr[(x, y) \notin R] \cdot (2 + \max_i c_{\mu_i}) = 2 + \Pr[(x, y) \notin R] \max_i c_{\mu_i} \leq 2 + (1 - \delta) c_\mu.$$

Therefore,  $c_\mu \leq 2/\delta$  as claimed. ◀

Combined with Theorem 1.10, we get the following bound in terms of sign-rank.

► **Corollary C.2.** *For every sign matrix  $A$  we have*

$$\text{CC}^{\text{avg}}(A) \leq 2^{2 \text{rank}_\pm(A) + 3}.$$

Theorem 3.2 shows that there is no converse to Corollary C.2. In particular, there are  $n \times n$  sign matrices  $A$  with sign-rank  $n^{\Omega(1)}$  and  $\text{rect}(A) = O(1)$ . By Proposition C.1, we have  $\text{CC}^{\text{avg}}(A) = O(1)$ , and thus there is a strong separation between sign-rank and  $\text{CC}^{\text{avg}}(A)$ .

## D Gamma-2 norm and randomized communication complexity

Recall the following well-known inequality.

► **Lemma D.1** (Hoeffding's inequality). *For  $i = 1, \dots, n$ , let  $X_i$  be independent random variables taking values from range  $[a_i, b_i]$  and let  $X = \sum_{i=1}^n X_i$ . Then,*

$$\Pr[|X - \mathbb{E}[X]| \geq t] < 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

The next proposition proves the equivalence of the approximate  $\gamma_2$  norm and the randomized communication complexity.

► **Proposition D.2** ([31, 23]). *For every sign matrix  $A_{\mathcal{X} \times \mathcal{Y}}$  and every  $\epsilon \in (0, 1)$ , we have*

$$\log \|A\|_{\gamma_2, \epsilon} \leq R_{\frac{\epsilon}{2}}(A) \leq O\left(\frac{\log(1/\epsilon)}{(1-\epsilon)^2} \|A\|_{\gamma_2, \epsilon}^2\right).$$

**Proof.**

**Lower-bound:** Consider a randomized protocol  $\pi_R$  of cost  $c = R_{\frac{\epsilon}{2}}(A)$  that computes  $A_{\mathcal{X} \times \mathcal{Y}}$  with two-sided error at most  $\frac{\epsilon}{2}$ . In this notation, the subscript  $R$  denotes the random variable that corresponds to the randomness in the protocol, and any fixation of  $R$  to a value  $r$  corresponds to a deterministic protocol  $\pi_r$  of communication cost at most  $c$ . Let  $\Pi_r$  denote the matrix that corresponds to the output of the deterministic protocol  $\pi_r$ . A deterministic communication protocol  $\pi_r$  of cost  $c$  provides a partition of  $\mathcal{X} \times \mathcal{Y}$  into at most  $2^c$  rectangles, and thus  $\Pi_r$  can be written as a sum of at most  $2^c$  rank-1 sign matrices. Since the  $\gamma_2$  norm of a non-zero rank-1 sign matrix is 1, we have  $\|\Pi_r\|_{\gamma_2} \leq 2^c$ . By convexity

$$\|\mathbb{E}_R[\Pi_R]\|_{\gamma_2} \leq \mathbb{E}_R[\|\Pi_R\|_{\gamma_2}] \leq \max_r \|\Pi_r\|_{\gamma_2} \leq 2^c.$$

Since  $\pi_R$  has error at most  $\epsilon/2$ , we have

$$|A_{xy} - \mathbb{E}_R[\pi_R(x, y)]| = 2 \cdot \Pr[A_{xy} \neq \pi_R(x, y)] \leq \epsilon,$$

which implies  $\|A\|_{\gamma_2, \epsilon} \leq 2^c$  as desired.

**Upper-bound:** The approximate norm  $\|A\|_{\gamma_2, \epsilon}$  is defined as the *infimum* of  $\|B\|_{\gamma_2}$  such that  $\|A - B\| \leq \epsilon$ . Hence, for every  $\eta > 0$ , there exists a real matrix  $B$  with  $\|B\|_{\gamma_2} \leq \|A\|_{\gamma_2, \epsilon}$  and  $\|A - B\|_{\infty} \leq \epsilon + \eta$ . Pick a small positive  $\eta < \frac{1-\epsilon}{2}$ , and consider such a  $B$ .

As it is stated in [30, Equation (2.3)], it follows from Grothendieck's inequality that the  $\gamma_2$  norm is equivalent to the so-called  $\nu$ -norm. In particular, there exist rank-1 sign matrices  $B_1, \dots, B_m$  and real numbers  $\lambda_1, \dots, \lambda_m \in \mathbb{R}$  with  $L := \sum_{i=1}^m |\lambda_i| \leq \frac{\pi}{2 \ln(1+\sqrt{2})} \|B\|_{\gamma_2}$  such that

$$B = \sum_{i=1}^m \lambda_i B_i.$$

We will convert this to a randomized protocol. Pick  $D$  randomly from  $\{B_1, \dots, B_m\}$  according to the probability distribution

$$\Pr[D = B_i] = \frac{|\lambda_i|}{\sum_{i=1}^m |\lambda_i|}.$$

Note that for every  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ , we have  $\mathbb{E}[D_{xy}] = B_{xy}/L$  and  $|D_{xy}| = 1$ . Let  $\delta = \frac{1-\epsilon}{2}$  and  $N = 2\delta^{-2} L^2 \log(4/\epsilon) = \frac{8L^2 \log(4/\epsilon)}{(1-\epsilon)^2}$ . Let  $D_1, \dots, D_N$  be i.i.d. copies of  $D$  and define  $\tilde{D} = \frac{L}{N} \sum_{i=1}^N D_i$ .

Note that for every  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ , by applying Hoeffding's inequality (Lemma D.1), we have

$$\Pr[|\tilde{D}_{xy} - B_{xy}| \geq \delta] < 2 \exp\left(-\frac{2\delta^2}{4N \cdot (L/N)^2}\right) \leq \frac{\epsilon}{2},$$

where the last inequality is by the choice of  $N$ .

## 22:24 Lower Bound Methods for Sign-Rank and Their Limitations

Let  $E$  be the  $\pm 1$  rounding of  $\tilde{D}$ , that is  $E_{xy} = 1$  iff  $\tilde{D}_{xy} \geq 0$ . Since  $\|B - A\|_\infty \leq \epsilon + \eta$ , for every  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ , we have

$$\begin{aligned} \Pr[E_{xy} \neq A_{xy}] &\leq \Pr[|\tilde{D}_{xy} - B_{xy}| \geq 1 - \epsilon - \eta] \leq \Pr[|\tilde{D}_{xy} - B_{xy}| \geq \frac{1 - \epsilon}{2}] \\ &\leq \Pr[|\tilde{D}_{xy} - B_{xy}| \geq \delta] \leq \frac{\epsilon}{2}. \end{aligned}$$

Each  $D_i$  can be computed with communication cost at most 2. Since  $\tilde{D}_{xy}$  can be computed by rounding a linear combination of  $N$  such  $D_i$ 's, it can be computed with communication cost at most  $2N = O\left(\frac{\log(1/\epsilon)}{(1-\epsilon)^2} \|A\|_{\gamma_2, \epsilon}^2\right)$ . This concludes the statement.  $\blacktriangleleft$

# Black-Box Identity Testing of Noncommutative Rational Formulas of Inversion Height Two in Deterministic Quasipolynomial Time

V. Arvind ✉

The Institute of Mathematical Sciences, HBNI, Chennai, India

Abhranil Chatterjee ✉

Indian Institute of Technology Bombay, India

Partha Mukhopadhyay ✉

Chennai Mathematical Institute, India

---

## Abstract

Hrubeš and Wigderson [11] initiated the complexity-theoretic study of noncommutative formulas with inverse gates. They introduced the Rational Identity Testing (RIT) problem which is to decide whether a noncommutative rational formula computes zero in the free skew field. In the white-box setting, there are deterministic polynomial-time algorithms due to Garg, Gurvits, Oliveira, and Wigderson [10] and Ivanyos, Qiao, and Subrahmanyam [13].

A central open problem in this area is to design an efficient deterministic *black-box* identity testing algorithm for rational formulas. In this paper, we solve this for the first nested inverse case. More precisely, we obtain a deterministic quasipolynomial-time black-box RIT algorithm for noncommutative rational formulas of inversion height two via a hitting set construction. Several new technical ideas are involved in the hitting set construction, including concepts from matrix coefficient realization theory [19] and properties of cyclic division algebras [15]. En route to the proof, an important step is to embed the hitting set of Forbes and Shpilka for noncommutative formulas [9] inside a cyclic division algebra of small index.

**2012 ACM Subject Classification** Theory of computation → Algebraic complexity theory; Theory of computation

**Keywords and phrases** Rational Identity Testing, Black-box Derandomization, Cyclic Division Algebra, Matrix coefficient realization theory

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.23

**Category** RANDOM

**Related Version** *Full Version*: <https://eccc.weizmann.ac.il/report/2022/067/>

**Acknowledgements** We thank the anonymous reviewers of an earlier version for their valuable comments that have helped us to improve the presentation.

## 1 Introduction

The broad goal of *algebraic complexity* is to study the complexity of computing polynomials and rational functions using basic arithmetic operations: additions, multiplications, and inverses. *Arithmetic circuits* and *arithmetic formulas* are two extensively studied models of computation. An important sub-area of algebraic complexity is *noncommutative computation*: the set of monomials over variables  $X$  is the free monoid  $X^*$  of all words. In particular, variables in  $X$  do not commute (i.e.  $xy \neq yx$ ). If we allow only the addition and multiplication gates in the noncommutative formulas/circuits, they compute noncommutative polynomials (similar to the commutative case) in the free algebra.



© V. Arvind, Abhranil Chatterjee, and Partha Mukhopadhyay;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 23; pp. 23:1–23:22



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the commutative case, the role of inverses is well understood, but in the noncommutative world it is quite subtle. To elaborate, it is known that *any* commutative rational expression can be expressed as  $fg^{-1}$  where  $f$  and  $g$  are two commutative polynomials [18]. However, noncommutative rational expressions (formulas with inverses) such as  $x^{-1} + y^{-1}$  or  $xy^{-1}x$  cannot be represented as  $fg^{-1}$  or  $f^{-1}g$ . If we have *nested inverses*, it makes the rational expression more complicated, for example  $(z + xy^{-1}x)^{-1} - z^{-1}$ . Moreover, a noncommutative rational expression is not always defined on a matrix substitution. For a noncommutative rational expression  $\mathfrak{r}$ , its *domain of definition* is the set of matrix tuples (of any dimension) where  $\mathfrak{r}$  is defined. We denote it by  $\text{dom}(\mathfrak{r})$ . Two rational expressions  $\mathfrak{r}_1$  and  $\mathfrak{r}_2$  are *equivalent* if they agree on  $\text{dom}(\mathfrak{r}_1) \cap \text{dom}(\mathfrak{r}_2)$ . This induces an equivalence relation on the set of all noncommutative rational expressions (with nonempty domain of definition). It was used by Amitsur in his characterization of the *universal* free skew field [2] and the equivalence classes are called *noncommutative rational functions*.

The *inversion height* of a rational formula is the maximum number of inverse gates in a path from an input gate to the output gate. It is known [11] that the inversion height of a rational formula of size  $s$  is bounded by  $O(\log s)$ . Hrubeš and Wigderson [11] consider the *rational identity testing* problem (RIT) of testing the equivalence of two rational formulas. It is the same as testing whether a rational formula computes the zero function in the free skew field. In other words, decide whether there exists a matrix tuple (of any dimension) such that the rational formula evaluates to nonzero on that substitution. Rational expressions exhibit peculiar properties which seem to make the RIT problem quite different from polynomial identity testing. For example, Bergman has constructed an explicit rational expression, of inversion height two, which is an identity for  $3 \times 3$  matrices but not an identity for  $2 \times 2$  matrices [4]. Also, the apparent lack of *canonical representations*, like a sum of monomials representation for polynomials, and the use of nested inverses in noncommutative rational expressions complicate the problem. For example, the rational expression  $(x + xy^{-1}x)^{-1} + (x + y)^{-1} - x^{-1}$  of inversion height two is a rational identity, known as Hua's identity [12]. However, Hrubeš and Wigderson give an efficient reduction from the RIT problem to the singularity testing problem of linear pencils.

A *linear pencil*  $L$  of size  $s$  over noncommuting variables  $\underline{x} = \{x_1, \dots, x_n\}$  is a  $s \times s$  matrix whose entries are linear forms in  $\underline{x}$  variables, i.e.  $L = A_0 + \sum_{i=1}^n A_i x_i$ , where each  $A_i$  is an  $s \times s$  matrix over the field  $\mathbb{F}$ . A rational function  $\mathfrak{r}$  in  $\mathbb{F}\langle \underline{x} \rangle$  has a *linear pencil representation*  $L$  of size  $s$ , if for some  $i, j \in [s]$ ,  $\mathfrak{r} = (L^{-1})_{i,j}$ . In particular, if  $\mathfrak{r}$  is a rational formula of size  $s$ , Hrubeš and Wigderson have shown that  $\mathfrak{r}$  has a linear pencil representation  $L$  of size at most  $2s$  such that  $\mathfrak{r}$  is defined on a matrix tuple if and only if  $L$  is invertible on that tuple [11]. Using this connection, they reduce the RIT problem to the problem of testing whether a given linear pencil is invertible over the free skew field in deterministic polynomial time.

The latter is the noncommutative SINGULAR problem, whose commutative analog is the symbolic determinant identity testing problem. The deterministic complexity of symbolic determinant identity testing is completely open [14] in the commutative setting. In contrast, the SINGULAR problem in noncommutative setting has deterministic polynomial-time algorithms in the white-box model due to [10, 13]. The algorithm in [10] is based on operator scaling and the algorithm in [13] is based on the second Wong sequence and a constructive version of the *regularity lemma*. As a consequence, a deterministic polynomial-time white-box RIT algorithm follows.

A central open problem is to design an efficient *deterministic* RIT algorithm in the black-box case [10]. There is a randomized polynomial-time black-box algorithm for the problem [7]. Can we derandomize this result even in some restricted settings, for example

when the inversion height of the input rational formula is small? Notice that inversion height zero rational formulas are just noncommutative formulas, and a result of Forbes and Shpilka have shown a deterministic quasipolynomial-time identity testing for those (more generally, for noncommutative ABPs) via a hitting set construction [9]. Whether their approach can be extended to the RIT problem for rational formulas is a natural direction and we prove the following theorem which is our main result.

► **Theorem 1.1.** *For the class of rational formulas in  $\mathbb{Q}\langle x_1, \dots, x_n \rangle$  of inversion height two and size at most  $s$ , we can construct a hitting set  $\mathcal{H} \subseteq \text{Mat}_d^n(\mathbb{Q})$  of size  $(ns)^{O(\log ns)}$  in deterministic  $(ns)^{O(\log ns)}$ -time. The parameter  $d$  is  $\text{poly}(s, n)$  bounded.*

Before this work, no such hitting set construction was known that could handle nested inverses. As we discuss in the next section, even to derandomize RIT for the special case of inversion height two, we need to accumulate several ideas involving cyclic division algebras [15] and matrix coefficient realization theory [19] combined with the hitting set construction in [9].

## Proof Idea

Consider the following noncommutative rational formula,  $\mathbf{r} = [x, y]^{-1} = (xy - yx)^{-1}$ . Clearly there is no point in  $\text{dom}(\mathbf{r})$  from the ground field, and the natural idea is to expand the series around a matrix point. Let  $(p_1, p_2)$  be a matrix pair such that  $[p_1, p_2]$  is invertible and let  $\mathbf{r}(p_1, p_2) = [p_1, p_2]^{-1} = q$ . Then,

$$\mathbf{r}(x + p_1, y + p_2) = ([p_1, p_2] - [p_2, x] - [y, p_1] - [y, x])^{-1}.$$

Simplifying this we can write  $\mathbf{r}(x + p_1, y + p_2) = (I - g(x, y))^{-1}q$  where  $g(x, y) = q([p_2, x] + [y, p_1] + [y, x])$ . Now expanding this using  $(I - g(x, y))^{-1} = \sum_{i \geq 0} (g(x, y))^i$ , we can see that every term in the expansion looks like  $a_0 z_1 a_1 z_2 \dots a_{d-1} z_d a_d$  where each  $a_j$  is a matrix and  $z_j \in \{x, y\}$ . In the language of matrix coefficient realization theory [19], such terms (resp. series) are called generalized words or monomials (resp. generalized series). In fact if a rational formula  $\mathbf{r}$  of size  $s$  has a defined point  $\underline{u}$  in some dimension  $l$  (in other words  $\underline{u} \in \text{dom}(\mathbf{r})$ , and we use it interchangeably), Volčič shows that one can associate a special class of generalized series, a recognizable generalized series to the shifted rational formula [19]:

$$\mathbf{r}(\underline{x} + \underline{u}) = \mathbf{c} \left( I_{2ls} - \sum_{j=1}^n A^{x_j} \right)^{-1} \mathbf{b}.$$

Here  $\mathbf{c} \in (\text{Mat}_l(\mathbb{F}))^{1 \times 2s}$  and  $\mathbf{b} \in (\text{Mat}_l(\mathbb{F}))^{2s \times 1}$ . The matrices  $A^{x_j}, 1 \leq j \leq n$  are of dimension  $2s \times 2s$  as a block matrix and  $(k_1, k_2)^{th}$  entry of  $A^{x_j}$  is given by a generalized linear form  $C_{k_1, k_2, j} x_j C'_{k_1, k_2, j}$  where  $C_{k_1, k_2, j}, C'_{k_1, k_2, j} \in \text{Mat}_l(\mathbb{F})$ .

Focusing on our problem for rational formulas of inversion height two, the first step is to construct a quasipolynomial-size set  $\mathcal{H}_1$  of matrix tuples of small dimension such that for every nonzero rational formula  $\mathbf{r}$  of inversion height two, there exists a point  $\underline{u} \in \mathcal{H}_1$  on which  $\mathbf{r}$  is defined. Given such a point, testing whether  $\mathbf{r}$  is zero or not reduces to testing whether the generalized series  $\mathbf{r}(\underline{x} + \underline{u})$  is zero or not. This is formally stated in Theorem 2.8. For a recognizable series in algebraic automata theory, a standard result by Schützenberger shows that the identity testing of such infinite series is equivalent to the identity testing of polynomial obtained by truncation of the series up to a small degree [8, Corollary 8.3]. We can adapt this result in the case of generalized series too and observe that the truncated



generalized polynomial (of small degree  $d$ ) can be represented by an algebraic branching program with edge labels are linear forms over matrices. Such ABPs can be identity tested efficiently using an adaptation of the hitting set construction shown by Forbes-Shpilka [9].

Although it is not clear how to carry out the truncation in the black-box setting, we can show that a suitable scaling of the hitting set for such generalized ABPs is good enough to hit the generalized series too. To fit the dimension correctly, throughout the computation the coefficient matrices should be embedded in the matrix algebra of dimension  $dl$  using the inclusion map  $\iota : a \rightarrow a \otimes I_d$ . This is shown in Proposition 5.1.

Clearly,  $\tau$  is defined at a point  $\underline{u}$  if and only if all the maximal sub-formulas of inversion height one in  $\tau$  evaluate to invertible matrices on  $\underline{u}$ . One can consider the product of all such maximal sub-formulas of inversion height one. Observe that the product is also of inversion height one and of size at most  $s$ . Thus our goal is now re-defined: construct  $\mathcal{H}_1$  such that for every size- $s$  rational formula  $\tau$  of inversion height one, there is a point  $\underline{u} \in \mathcal{H}_1$  at which  $\tau(\underline{u})$  is invertible. We call such a hitting set a *strong hitting set*. We give the formal definition.

► **Definition 1.2 (Strong hitting set).** *For a class of rational functions (resp. polynomials) a hitting set  $\mathcal{H}$  is strong if any nonzero rational function (resp. polynomial) in that class evaluates to an invertible matrix at some point in  $\mathcal{H}$ .*

Before we describe our construction of a strong hitting set for rational formulas of inversion height one, notice that a rational formula  $\tau$  of inversion height one is defined at a point  $\underline{v}$  if and only if all sub-formulas which are input to inverse gates evaluate to invertible matrices on  $\underline{v}$ . These sub-formulas are just noncommutative formulas. Since the Forbes-Shpilka hitting set [9] for noncommutative formulas consists of tuples of nilpotent matrices, it is not directly applicable to our problem.

However, it is possible to adapt their construction and get a strong hitting set, also of quasipolynomial size, such that every size- $s$  nonzero noncommutative formula evaluates to an invertible matrix on some matrix tuple in the strong hitting set<sup>1</sup>. Expanding  $\tau$  around such a point would again lead to a generalized series, and (a somewhat more involved) truncation and scaling argument show that we can get a strong hitting set for  $\tau$  by constructing a strong hitting set for *generalized* ABPs whose edges are labeled by linear forms over matrices. This is the essence of the second part of Proposition 5.1.

At this point, we face a serious obstacle. How do we find invertible matrices in the image of the generalized ABPs? In other words, how to construct a strong hitting set for generalized ABPs? The main insight is that, if the matrices present in the linear forms of the generalized ABPs are from a division algebra, then one can construct a strong hitting set from a hitting set. To implement this, we construct the hitting set for noncommutative formulas (which are of inversion height zero) over a division algebra of small index and expand the rational formula with respect to the points in that hitting set. Why does it work? Roughly speaking, as already mentioned it is easier to find a nonzero in the image of generalized ABPs and if the computation occurs in a division algebra then a computed nonzero element is also invertible.

Section 4 elaborates on this idea. In particular, Lemma 4.2 provides an existential argument showing that if the linear forms of the generalized ABP are defined over a division algebra  $D$  of dimension  $\ell$ , then there *exists* a substitution to the variables from  $D$  such that the generalized ABP evaluates to an invertible matrix. The proof uses two ideas. Firstly, we show that such a point exists in the full matrix algebra of dimension  $\ell$ . Then we use

---

<sup>1</sup> This was first explicitly constructed in [3].

Proposition 2.13 to find such a certificate in  $D$ . Once we establish the existential argument, we can use a reduction to the hitting set construction of ROABPs (in unknown order) [1] to construct the hitting set in quasi-polynomial time. To work out the technical details we need to employ the inclusion map  $\iota' : a \rightarrow I_d \otimes a$  for the coefficients which are now elements of division algebra. In ring theory the maps  $\iota$  and  $\iota'$  are compatible: by the Skolem-Noether theorem [16, Theorem 3.1.2] there is an invertible matrix  $q_0$  such that  $q_0(I_d \otimes a)q_0^{-1} = a \otimes I_d$  for all  $a$ . However, in our case, we give a simple explicit construction of a permutation matrix  $q_0$ .

In the remaining part of the proof sketch, we informally describe how to find a hitting set for noncommutative formulas (more generally for noncommutative ABPs) in a division algebra of a small index. For simplicity, suppose the ABP degree is  $2^d$ . The Forbes-Shpilka hitting set [9] has a recursive construction and it is by a reduction to the hitting set construction for ROABPs (read-once algebraic branching programs) over the commutative variables  $u_1, u_2, \dots, u_{2^d}$ . The recursive step in the construction is by combining hitting sets (via hitting set generator  $\mathcal{G}_{d-1}$ ) for two halves of degree  $2^{d-1}$  [9] with a rank preserving step of matrix products to obtain the generator  $\mathcal{G}_d$  at the  $d^{\text{th}}$  step. More precisely,  $\mathcal{G}_d$  is a map from  $\mathbb{F}^{d+1} \rightarrow \mathbb{F}^{2^d}$  that stretches the seed  $(\alpha_1, \dots, \alpha_{d+1})$  to a  $2^d$  tuple for the read-once variables.

For our purpose, we take a classical construction of cyclic division algebras [15, Chapter 5]. The division algebra  $D = (K/F, \sigma, z)$  is defined using an indeterminate  $x$  as the  $\ell$ -dimensional vector space:

$$D = K \oplus Kx \oplus \dots \oplus Kx^{\ell-1},$$

where the (noncommutative) multiplication for  $D$  is defined by  $x^\ell = z$  and  $xb = \sigma(b)x$  for all  $b \in K$ . Here  $\sigma : K \rightarrow K$  is an automorphism of the Galois group  $\text{Gal}(K/F)$ . The field  $F = \mathbb{Q}(z)$  and  $K = F(\omega)$ , where  $z$  is an indeterminate and  $\omega$  is an  $\ell^{\text{th}}$  primitive root of unity. The matrix representation of a general element in  $D$  is of the following form:

$$\begin{bmatrix} 0 & b & 0 & \dots & 0 \\ 0 & 0 & \sigma(b) & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \sigma^{\ell-2}(b) \\ z\sigma^{\ell-1}(b) & 0 & \dots & 0 & 0 \end{bmatrix}.$$

To embed the hitting set of [9], we need to choose  $\ell = 2^L$  appropriately larger than  $2^d$ . As it turns out the construction of the division algebra requires a tower of extension fields of  $F$ , with a higher-order root of unity at each stage.

Specifically, let  $\omega_i = \omega^{2^{a_i}}$  for  $a_1 > a_2 > \dots > a_d > 0$ , where  $a_i$  are positive integers suitably chosen. Let  $K_i = F(\omega_i)$  be the cyclic Galois extension for  $1 \leq i \leq d$  giving a tower of extension fields

$$F \subset F(\omega_1) \subset F(\omega_2) \subset \dots \subset F(\omega_d) \subset F(\omega).$$

As we show in Section 3 that we require two properties of  $\omega_i, 1 \leq i \leq d$ . Firstly, for the hitting set generator  $\mathcal{G}_i$  we will choose the root of unity as  $\omega_i$  and the variable  $\alpha_i$  will take values only in the set  $W_i = \{\omega_i^j \mid 1 \leq j \leq 2^{L-a_i}\}$ . We also require that the  $K$ -automorphism  $\sigma$  has the property that for all  $1 \leq i \leq d$  the map  $\sigma^{2^i}$  fixes  $\omega_i$ . In fact we will ensure that  $\sigma^{2^i}$  has  $F(\omega_i)$  as its fixed field. The construction of  $D$  satisfying the above properties is the main technical step in Section 3.

Implementing all these steps we get a quasipolynomial-size hitting set over  $\mathbb{Q}(\omega, z)$ . Then we show how to transfer the hitting set over  $\mathbb{Q}$  itself by a relatively standard idea that treats the parameters  $\omega$  and  $z$  as *fresh indeterminates*  $t_1, t_2$  and vary them over a suitably chosen polynomial-size set. This is sketched in Section 5.

We include a brief discussion in the full version about possibly extending our approach to any constant inversion height formula.

## Organization

In Section 2, we collect some background results from algebraic complexity theory, matrix coefficient realization theory, and cyclic division algebra. Section 3 contains the proof that the Forbes-Shpilka hitting set can be embedded in a cyclic division algebra of small index. In Section 4, we construct a quasipolynomial-size strong hitting set for generalized ABPs over division algebra. Finally, in Section 5 we combine the results developed in Section 3 and Section 4 to obtain our main result which gives a quasipolynomial-size hitting set for rational formulas of inversion height two.

## 2 Background and Notation

Throughout the paper, we use  $\mathbb{F}, F, K$  for fields. The notation  $\text{Mat}_m(\mathbb{F})$  (respectively,  $\text{Mat}_m(F)$ ,  $\text{Mat}_m(K)$ ) are used for  $m$  dimensional matrix algebra over  $\mathbb{F}$  (respectively over  $F, K$ ) where  $m$  is clear from the context.  $D$  is used to denote cyclic division algebras. Let  $\underline{x}$  be the set of variables  $\{x_1, \dots, x_n\}$ . Sometime we use notation like  $\underline{u}, \underline{v}, \underline{p}, \underline{q}$  to denote the matrix tuples in suitable matrix algebras. The free noncommutative ring of polynomials over a field  $\mathbb{F}$  is denoted by  $\mathbb{F}\langle \underline{x} \rangle$ . The ring of *formal power series* is denoted by  $\mathbb{F}\langle\langle \underline{x} \rangle\rangle$ . For a series (or polynomial)  $S$ , the coefficient of a monomial (word) in  $S$  is denoted by  $[m]S$ .

### 2.1 Algebraic Complexity

► **Definition 2.1** (Algebraic Branching Program). *An algebraic branching program (ABP) is a layered directed acyclic graph. The vertex set is partitioned into layers  $0, 1, \dots, d$ , with directed edges only between adjacent layers ( $i$  to  $i + 1$ ). There is a source vertex of in-degree 0 in the layer 0, and one out-degree 0 sink vertex in layer  $d$ . Each edge is labeled by an affine  $\mathbb{F}$ -linear form. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the ordered product of affine forms labeling the path edges.*

The *size* of the ABP is defined as the total number of nodes and the *width* is the maximum number of nodes in a layer. The ABP model is defined for computing commutative or noncommutative polynomials. ABPs of width  $r$  can also be seen as iterated matrix multiplication  $\mathbf{c} \cdot M_1 M_2 \cdots M_\ell \cdot \mathbf{b}$ , where  $\mathbf{c}, \mathbf{b}$  are  $1 \times r$  and  $r \times 1$  vectors respectively and each  $M_i$  is a  $r \times r$  matrix, whose entries are affine linear forms over  $\underline{x}$ .

We also consider commutative set-multilinear ABPs and read-once oblivious ABPs (ROABPs). For the set-multilinear case, the (commutative) variable set is partitioned as  $Y = Y_1 \sqcup Y_2 \sqcup \cdots \sqcup Y_d$  where for each  $j \in [d]$ ,  $Y_j = \{y_{ij}\}_{i=1}^n$ . An ABP  $B$  is homogeneous set-multilinear if each edge in the  $j^{\text{th}}$  layer of the ABP is labelled by linear forms over  $Y_j$ . For ROABP, a different variable is used for each layer, and the edge labels are univariate polynomials. Therefore, an ROABP of  $d$  layers can be represented as  $\mathbf{c} \cdot M_1(v_1) M_2(v_2) \cdots M_{v_d}(d) \cdot \mathbf{b}$ . We say that the ROABP respects the variable order  $v_1 < v_2 < \cdots < v_d$ .

## Identity testing results

We say a set  $\mathcal{H} \subseteq \mathbb{F}^n$  is a hitting set for a circuit class  $\mathcal{C}$  if for every  $n$ -variate polynomial  $f$  in  $\mathcal{C}$ ,  $f \not\equiv 0$  if and only if  $f(\underline{a}) \neq 0$  for some  $\underline{a} \in \mathcal{H}$ . For the class of ROABPs, Forbes and Shpilka [9] obtained the first quasipolynomial-time black-box algorithm by constructing a hitting set of the same size.

► **Theorem 2.2.** *For the class of polynomials computable by a width  $r$ , depth  $D$ , individual degree  $< n$  ROABPs of known order, if  $|\mathbb{F}| \geq (2Dnr^3)^2$ , there is a  $\text{poly}(D, n, r)$ -explicit hitting set of size at most  $(2Dn^2r^4)^{\lceil \log D+1 \rceil}$ .*

Indeed, they proved something more general.

► **Definition 2.3** (Hitting Set Generator). *A polynomial map  $\mathcal{G} : \mathbb{F}^t \rightarrow \mathbb{F}^n$  is a generator for a circuit class  $\mathcal{C}$  if for every  $n$ -variate polynomial  $f$  in  $\mathcal{C}$ ,  $f \equiv 0$  if and only if  $f \circ \mathcal{G} \equiv 0$ .*

► **Theorem 2.4** ([9, Construction 3.13, Lemma 3.21]). *For the class of polynomials computable by a width  $r$ , depth  $D$ , individual degree  $< n$  ROABPs of known order, one can construct a hitting set generator  $\mathcal{G} : \mathbb{F}^{\lceil \log D+1 \rceil} \rightarrow \mathbb{F}^D$  of degree  $Dnr^4$  efficiently.*

The hitting set is defined as  $\mathcal{H} \subseteq \text{Mat}_d^n(\mathbb{F})$  for any class of noncommutative polynomials. For the black-box case, Forbes and Shpilka [9], have shown an efficient construction of quasipolynomial-size hitting set for noncommutative ABPs. Consider the class of noncommutative ABPs of width  $w$ , and depth  $d$  computing polynomials in  $\mathbb{F}\langle X \rangle$ . The result of Forbes and Shpilka provide an explicit construction (in quasipolynomial-time) of a set  $\mathcal{H}_{w,d,n}$  contained in  $\text{Mat}_{d+1}(\mathbb{F})$ , such that for any ABP (with parameters  $w$  and  $d$ ) computing a nonzero polynomial  $f$ , there always exists  $(p_1, \dots, p_n) \in \mathcal{H}_{w,d,n}$  such that  $f(\underline{p}) \neq 0$ .

► **Theorem 2.5** (Forbes and Shpilka [9]). *For all  $w, d, n \in \mathbb{N}$ , if  $|\mathbb{F}| \geq \text{poly}(d, n, w)$ , then there is a hitting set  $\mathcal{H}_{w,d,n} \subset \text{Mat}_{d+1}(\mathbb{F})$  for noncommutative ABPs of parameters  $w, d, n$  such that  $|\mathcal{H}_{w,d,n}| \leq (wdn)^{O(\log d)}$  and there is a deterministic algorithm to output the set  $\mathcal{H}_{w,d,n}$  in time  $(wdn)^{O(\log d)}$ .*

## Recognizable series

A comprehensive treatment is in the book by Berstel and Reutenauer [5]. We will require the following concepts. Recall that  $\mathbb{F}\langle\langle x \rangle\rangle$  is the formal power series ring over a field  $\mathbb{F}$ . A series  $S$  in  $\mathbb{F}\langle\langle x \rangle\rangle$  is *recognizable* if it has the following linear representation: for some integer  $s$ , there exists a row vector  $\underline{c} \in \mathbb{F}^{1 \times s}$ , a column vector  $\underline{b} \in \mathbb{F}^{s \times 1}$  and an  $s \times s$  matrix  $M$  whose entries are homogeneous linear forms over  $x_1, \dots, x_n$  i.e.  $\sum_{i=1}^n \alpha_i x_i$  such that  $S = \underline{c} \left( \sum_{k \geq 0} M^k \right) \underline{b}$ . Equivalently,  $S = \underline{c}(I - M)^{-1} \underline{b}$ . We say,  $S$  has a representation  $(\underline{c}, M, \underline{b})$  of size  $s$ .

The following theorem is a basic result in algebraic automata theory.

► **Theorem 2.6.** *A recognizable series with representation  $(\underline{c}, M, \underline{b})$  of size  $s$  is nonzero if and only if  $\underline{c} \left( \sum_{k \leq s-1} M^k \right) \underline{b}$  is nonzero.*

It has a simple linear algebraic proof [8, Corollary 8.3, Page 145]. This result is generally attributed to Schützenberger. For this paper, the theorem is used to apply that the truncated series is computable by a small noncommutative ABP, therefore, reducing zero-testing of recognizable series to the identity testing of noncommutative ABPs.

## 2.2 Matrix Coefficient Realization Theory

We require some basic notions and results about generalized automata and generalized recognizable series from Volčič's article [19]. A detailed exposition is given in it [19].

A *generalized word* or a *generalized monomial* in  $x_1, \dots, x_n$  over the matrix algebra  $\text{Mat}_m(\mathbb{F})$  allows the matrices to interleave between variables. That is to say, a generalized monomial is of the form:  $a_0 x_{k_1} a_2 \cdots a_{d-1} x_{k_d} a_d$ , where  $a_i \in \text{Mat}_m(\mathbb{F})$ , and its degree is the number of variables  $d$  occurring in it. A finite sum of generalized monomials is a *generalized polynomial* in the ring  $\text{Mat}_m(\mathbb{F})\langle \underline{x} \rangle$ . A *generalized formal power series* over  $\text{Mat}_m(\mathbb{F})$  is an infinite sum of generalized monomials such that the sum has finitely many generalized monomials of degree  $d$  for any  $d \in \mathbb{N}$ . The ring of generalized series over  $\text{Mat}_m(\mathbb{F})$  is denoted  $\text{Mat}_m(\mathbb{F})\langle\langle \underline{x} \rangle\rangle$ .

A generalized series (resp. polynomial)  $S$  over  $\text{Mat}_m(\mathbb{F})$  admits the following canonical description. Let  $E = \{e_{i,j}, 1 \leq i, j \leq m\}$  be the set of elementary matrices. Express each coefficient matrix  $a$  in  $S$  in the  $E$  basis by a  $\mathbb{F}$ -linear combination and then expand  $S$ . Naturally each monomial of degree- $d$  in the expansion looks like  $e_{i_0, j_0} x_{k_1} e_{i_1, j_1} x_{k_2} \cdots e_{i_{d-1}, j_{d-1}} x_{k_d} e_{i_d, j_d}$  where  $e_{i_l, j_l} \in E$  and  $x_{k_l} \in \underline{x}$ . We say the series  $S$  (resp. polynomial) is identically zero if and only if it is zero under such expansion i.e. the coefficient associated with each generalized monomial is zero.

The evaluation of a generalized series over  $\text{Mat}_m(\mathbb{F})$  is defined on any  $k'm \times k'm$  matrix algebra for some integer  $k' \geq 1$  [19]. To match the dimension of the coefficient matrices with the matrix substitution, we use an inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{k'm}(\mathbb{F})$ , for example,  $\iota$  can be defined as  $\iota(a) = a \otimes I_{k'}$  or  $\iota(a) = I_{k'} \otimes a$ . Now, a generalized monomial  $a_0 x_{k_1} a_1 \cdots a_{d-1} x_{k_d} a_d$  over  $\text{Mat}_m(\mathbb{F})$  on matrix substitution  $(p_1, \dots, p_n) \in \text{Mat}_{k'm}^n(\mathbb{F})$  evaluates to

$$\iota(a_0) p_{k_1} \iota(a_1) \cdots \iota(a_{d-1}) p_{k_d} \iota(a_d)$$

under some inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{k'm}(\mathbb{F})$ . All such inclusion maps are known to be compatible by the Skolem-Noether theorem [16, Theorem 3.1.2]. Therefore, if a series  $S$  is zero with respect to some inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{k'm}(\mathbb{F})$ , then it is zero w.r.t. any such inclusion map.

The two notions of zeroness are equivalent [19, Proposition 3.13].

► **Definition 2.7** ([19]). A generalized series  $S$  in  $\text{Mat}_m(\mathbb{F})\langle\langle \underline{x} \rangle\rangle$  is said to be recognizable if it has the following linear representation. For some integer  $s$ , there exists a row-tuple of matrices  $\mathbf{c} \in (\text{Mat}_m(\mathbb{F}))^{1 \times s}$ , and  $\mathbf{b} \in (\text{Mat}_m(\mathbb{F}))^{s \times 1}$  and an  $s \times s$  matrix  $M$  whose entries are homogeneous generalized linear forms over  $x_1, \dots, x_n$  i.e.  $\sum_{i=1}^n \tilde{p}_i x_i \hat{p}_i$  where each  $\tilde{p}_i, \hat{p}_i \in \text{Mat}_m(\mathbb{F})$  such that  $S = \mathbf{c}(I - M)^{-1} \mathbf{b}$ . We say,  $S$  has a linear representation  $(\mathbf{c}, M, \mathbf{b})$  of size  $s$  over  $\text{Mat}_m(\mathbb{F})$ .

The linear representation is said to be over a subalgebra  $\mathfrak{A} \subseteq \text{Mat}_m(\mathbb{F})$  if  $\mathbf{c} \in \mathfrak{A}^{1 \times s}$ , and  $\mathbf{b} \in \mathfrak{A}^{s \times 1}$  and each  $\tilde{p}_i, \hat{p}_i \in \mathfrak{A}$ .

► **Theorem 2.8** ([19, Corollary 5.1, Proposition 3.13]).

1. Given a noncommutative rational formula  $\tau$  of size  $s$  over  $x_1, \dots, x_n$  and a matrix tuple  $\underline{p} \in \text{Mat}_m^n(\mathbb{F})$  in the domain of definition of  $\tau$ ,  $\tau(\underline{x} + \underline{p})$  is a recognizable generalized series with a representation of size at most  $2s$  over  $\text{Mat}_m(\mathbb{F})$ . Moreover, if  $\mathfrak{A} \subseteq \text{Mat}_m(\mathbb{F})$  is the subalgebra generated by the matrices  $p_1, \dots, p_n$  then  $\tau(\underline{x} + \underline{p})$  has, in fact, a linear representation over the subalgebra  $\mathfrak{A}$ .
2. Additionally,  $\tau(\underline{x})$  is zero in the free skew field if and only if  $\tau(\underline{x} + \underline{p})$  is zero as a generalized series.

**Proof.** For the first part, see Corollary 5.1 and Remark 5.2 of [19].

To see the second part, suppose  $\mathbf{r}(\underline{x})$  is zero in the free skew field. Then the fact that  $\mathbf{r}(\underline{x} + \underline{p})$  is a zero series follows from [19, Proposition 3.13]. If  $\mathbf{r}(\underline{x})$  is nonzero in the free skew field, then there exists a matrix tuple  $(q_1, \dots, q_n) \in \text{Mat}_l^n(\mathbb{F})$  such that  $\mathbf{r}(\underline{q})$  is nonzero. W.l.o.g. we can assume  $l = k'm$  for some integer  $k'$ . Fix an inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{k'm}(\mathbb{F})$ . Define a matrix tuple  $(q'_1, \dots, q'_n) \in \text{Mat}_{k'm}^n(\mathbb{F})$  such that  $q'_i = q_i - \iota(p_i)$ . Therefore, the series  $\mathbf{r}(\underline{x} + \underline{p})$  on  $(q'_1, \dots, q'_n)$  evaluates to  $\mathbf{r}(\underline{q})$ , under the inclusion map  $\iota$ , which is nonzero [19, Remark 5.2]. Therefore,  $\mathbf{r}(\underline{x} + \underline{p})$  is nonzero.  $\blacktriangleleft$

► **Remark 2.9.** Moreover we have the following [19, Section 5]. Let  $\mathbf{r}(\underline{x})$  be a rational formula of size  $s$  and  $\underline{p} \in \text{Mat}_m^n(\mathbb{F})$  be in the domain of definition of  $\mathbf{r}$ . Then  $\mathbf{r}(\underline{x} + \underline{p})$  has a linear representation  $(\mathbf{c}, M, \mathbf{b})$  of size  $2s$  over  $\text{Mat}_m(\mathbb{F})$ . Then  $M$  is a  $2s \times 2s$  matrix with entries of the form  $\sum_{i=1}^n \tilde{p}_i x_i \hat{p}_i$ ,  $\tilde{p}_i, \hat{p}_i \in \text{Mat}_m(\mathbb{F})$ . For an inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{k'm}(\mathbb{F})$  and a matrix tuple  $\underline{q} \in \text{Mat}_{k'm}^n(\mathbb{F})$ , replacing each  $\sum_{i=1}^n \tilde{p}_i x_i \hat{p}_i$  by  $\sum_{i=1}^n \iota(\tilde{p}_i) q_i \iota(\hat{p}_i)$ , we obtain a  $2sk'm \times 2sk'm$  matrix  $\iota(M)(\underline{q})$ . Then,

$$\mathbf{r}(\underline{q} + \iota(\underline{p})) = \iota(\mathbf{c}) \left( I_{2sk'm} - \iota(M)(\underline{q}) \right)^{-1} \iota(\mathbf{b}),$$

where  $\iota(\mathbf{c})$  and  $\iota(\mathbf{b})$  are an  $k'm \times k'ms$  and an  $k'ms \times k'm$  matrix respectively obtained by applying  $\iota$  on every  $m \times m$  blocks of  $\mathbf{c}$  and  $\mathbf{b}$ .

### 2.3 Cyclic Division Algebras

A division algebra  $D$  is an associative algebra over a (commutative) field  $\mathbb{F}$  such that all nonzero elements in  $D$  are units (they have a multiplicative inverse). In the context of this paper, we are interested in finite-dimensional division algebras. Specifically, we focus on cyclic division algebras and their construction [15, Chapter 5]. Let  $F = \mathbb{Q}(z)$ , where  $z$  is a commuting indeterminate. Let  $\omega$  be an  $\ell^{\text{th}}$  primitive root of unity. To be specific, let  $\omega = e^{2\pi i/\ell}$ . Let  $K = F(\omega) = \mathbb{Q}(\omega, z)$  be the cyclic Galois extension of  $F$  obtained by adjoining  $\omega$ . The elements of  $K$  are polynomials in  $\omega$  (of degree at most  $\ell - 1$ ) with coefficients from  $F$ .

Define  $\sigma : K \rightarrow K$  by letting  $\sigma(\omega) = \omega^k$  for some  $k$  relatively prime to  $\ell$  and stipulating that  $\sigma(a) = a$  for all  $a \in F$ . Then  $\sigma$  is an automorphism of  $K$  with  $F$  as fixed field and it generates the Galois group  $\text{Gal}(K/F)$ .

The division algebra  $D = (K/F, \sigma, z)$  is defined using a new indeterminate  $x$  as the  $\ell$ -dimensional vector space:

$$D = K \oplus Kx \oplus \dots \oplus Kx^{\ell-1},$$

where the (noncommutative) multiplication for  $D$  is defined by  $x^\ell = z$  and  $xb = \sigma(b)x$  for all  $b \in K$ . Then  $D$  is a division algebra of dimension  $\ell^2$  over  $F$  [15, Theorem 14.9]. The *index* of  $D$  is defined to be the square root of the dimension of  $D$  over  $F$ . In our example,  $D$  is of index  $\ell$ . Its elements have matrix representations in  $K^{\ell \times \ell}$  (the regular matrix representation defined by multiplication from the left) given below:

The matrix representation  $M(x)$  of  $x$  and  $M(b)$  of  $b \in K$  are:

$$M(x) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ z & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad M(b) = \begin{bmatrix} b & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma(b) & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2(b) & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^{\ell-2}(b) & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^{\ell-1}(b) \end{bmatrix}.$$



► **Remark 2.10.** We note that  $M(x)$  has a “circulant” matrix structure and  $M(b)$  is a diagonal matrix. For a vector  $v \in K^\ell$ , it is convenient to write  $\text{circ}(v_1, v_2, \dots, v_\ell)$  for the  $\ell \times \ell$  matrix with  $(i, i+1)^{\text{th}}$  entry  $v_i$  for  $i \leq \ell-1$ ,  $(\ell, 1)^{\text{th}}$  entry as  $v_\ell$  and remaining entries zero. Thus, we have  $M(x) = \text{circ}(1, 1, \dots, 1, z)$ . Similarly, we write  $\text{diag}(v_1, v_2, \dots, v_\ell)$  for the diagonal matrix with entries  $v_i$ .

► **Fact 2.11.** *The  $F$ -algebra generated by  $M(x)$  and  $M(b), b \in K$  is an isomorphic copy of the cyclic division algebra in the matrix algebra  $\text{Mat}_\ell(K)$ .*

► **Proposition 2.12.** *For all  $b \in K$ ,  $\text{circ}(b, \sigma(b), \dots, z\sigma^{\ell-1}(b)) = M(b) \cdot M(x)$ .*

Define  $C_{i,j} = M(\omega^{j-1}) \cdot M(x^{i-1})$  for  $1 \leq i, j \leq \ell$ . Observe that,  $\mathbb{B} = \{C_{ij}, i, j \in [\ell]\}$  be a  $F$ -generating set for the division algebra  $D$ .

A standard fact is the following.

► **Proposition 2.13** ([15, Section 14(14.13)]). *Then  $K$  linear span of  $\mathbb{B}$  is the entire matrix algebra  $\text{Mat}_\ell(K)$ .*

### 3 Embedding Forbes-Shpilka Hitting Set in a Division Algebra

Given any noncommutative algebraic branching program of size  $s$  computing a polynomial  $h \in \mathbb{F}\langle x_1, \dots, x_n \rangle$  of degree  $\tilde{d}$ , the hitting set  $\mathcal{H}$  contains a matrix tuple  $(p_1, \dots, p_n)$  such that  $h(p_1, \dots, p_n)$  is nonzero. Forbes and Shpilka [9] have shown a quasipolynomial-size hitting set construction contained in  $\text{Mat}_{\tilde{d}+1}^n(\mathbb{F})$ . For ABPs over  $\mathbb{Q}$ , we will show the construction of a hitting set  $\mathcal{H}$  which is contained in  $D^n$  such that  $D$  is a cyclic division algebra of index  $\ell$  where  $\ell$  is suitably chosen depending on  $n, \tilde{d}$  and  $s$ .

Before we present our construction, we recall the matrix substitutions from the Forbes-Shpilka hitting set construction. Their idea is to reduce PIT for noncommutative ABPs to PIT for commutative read-once oblivious ABP (ROABP) and to design a hitting set generator for the latter. Without loss of generality, we can assume that the given ABP is a  $\tilde{d}$ -product of  $r \times r$  matrices  $M = A_1 \cdot A_2 \cdots A_{\tilde{d}}$ , where the entries of each matrix  $A_i$  are homogeneous linear forms in  $x_1, x_2, \dots, x_n$ . The matrix  $A_i$  corresponds to the  $i^{\text{th}}$  of the ABP. The polynomial  $f$  in  $\mathbb{F}\langle x_1, \dots, x_n \rangle$  that the ABP computes is of degree  $\tilde{d} = 2^d$ , and  $f$  is an entry of this matrix product  $M$ .

We can write  $A_j = \sum_{i=1}^n A_{ij}x_i$ ,  $1 \leq j \leq \tilde{d}$ , where  $A_{ij} \in \mathbb{F}^{r \times r}$ . The entries  $M_{ij}$  of the matrix  $M$  are homogeneous polynomials in  $\mathbb{F}\langle x \rangle$ . The polynomial  $f$  is computed at some entry of  $M$  as the output polynomial. Let  $\{u_1, \dots, u_{\tilde{d}}\}$  be distinct commuting indeterminates. In [9], the authors make the following  $(\tilde{d}+1) \times (\tilde{d}+1)$  matrix substitution for each  $x_i$ , the only nonzero entries are in the super-diagonal and for each  $j$ , the  $(j, j+1)^{\text{th}}$  entry is  $u_j^i$ .

Evaluating the ABP for  $f$  on this matrix substitution  $x_i \leftarrow M(x_i)$  produces a  $(\tilde{d}+1) \times (\tilde{d}+1)$  matrix whose  $(1, \tilde{d}+1)^{\text{th}}$  entry is an ROABP as it effectively replaces each  $x_i$  variable at layer  $j$  by  $u_j^i$ . Therefore, the index  $j$  of  $u_j$  encodes the layer of the noncommutative ABP.

The black-box PIT algorithm then follows from the construction of a hitting set generator for commutative ROABPs:

$$\mathcal{G}_d : (\alpha_1, \dots, \alpha_d, \alpha_{d+1}) \mapsto (f_0(\alpha_1, \dots, \alpha_d, \alpha_{d+1}), f_1(\alpha_1, \dots, \alpha_d, \alpha_{d+1}), \dots, f_{2^d-1}(\alpha_1, \dots, \alpha_d, \alpha_{d+1})),$$

where each  $f_i$  is a polynomial of degree  $\text{poly}(2^d, r, n)$ . The actual points of the hitting set are obtained by choosing values for each variable  $\alpha_i$  from a subset of scalars  $U \subseteq \mathbb{F}$  of  $\text{poly}(2^d, r, n)$  size. This makes the size of the hitting set quasipolynomial. The final substitution for each  $x_i$  variable in the noncommutative ABP is the following:

$$M(x_i) = \begin{bmatrix} 0 & f_0^i(\alpha_1, \dots, \alpha_{d+1}) & 0 & \cdots & 0 \\ 0 & 0 & f_1^i(\alpha_1, \dots, \alpha_{d+1}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & f_{2^d-1}^i(\alpha_1, \dots, \alpha_{d+1}) \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \quad (1)$$

Therefore, one approach to embedding the matrix substitutions in a cyclic division algebra  $D = (K/F, \sigma, z)$  (where  $F = \mathbb{Q}(z)$ ) of index  $\ell$  (where  $\ell$  is the index of  $D$  which is larger than  $2^d$  that we fix later) would be to find a hitting set generator

$$\mathcal{G}_d : (\alpha_1, \dots, \alpha_d, \alpha_{d+1}) \mapsto (f_0(\alpha_1, \dots, \alpha_d, \alpha_{d+1}), f_1(\alpha_1, \dots, \alpha_d, \alpha_{d+1}), \dots, f_{2^d-1}(\alpha_1, \dots, \alpha_d, \alpha_{d+1})),$$

with the following additional property:  $f_{i+1}(\alpha_1, \dots, \alpha_{d+1}) = \sigma(f_i(\alpha_1, \dots, \alpha_{d+1}))$  for each  $0 \leq i \leq \ell - 2$ . In that case, consider the following  $\ell \times \ell$  matrix substitutions:

$$M(x_i) = \left[ \begin{array}{ccccc|ccc} 0 & f_0^i(\alpha) & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & f_1^i(\alpha) & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & f_{\tilde{d}-1}^i(\alpha) & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & f_{\tilde{d}}^i(\alpha) & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & f_{\ell-2}^i(\alpha) \\ z f_{\ell-1}^i(\alpha) & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array} \right].$$

Notice that the top-left  $(\tilde{d} + 1) \times (\tilde{d} + 1)$  submatrix of this substitution is exactly the substitution described in Equation 1. Therefore, evaluating a degree- $\tilde{d}$  noncommutative ABP  $B$  over  $\{x_1, \dots, x_n\}$  on these matrices will output the evaluation of corresponding ROABP in the  $(1, \tilde{d} + 1)^{th}$  entry as in [9]. Moreover, by Proposition 2.12, we can ensure that each  $M(x_i)$  is in the cyclic division algebra  $D$  assuming that each  $f_i(\alpha) \in K$ . Therefore, the output will also be in the division algebra  $D$  only. To conclude, for a nonzero noncommutative ABP, the image will be nonzero and in a division algebra, hence invertible.

Our goal is now to find a cyclic division algebra  $D = (K/F, \sigma, z)$  (where  $F = \mathbb{Q}(z)$ ) of index  $\ell$  (more than  $\tilde{d}$ ) and to construct a hitting set generator  $\mathcal{G}_d : \alpha \mapsto (f_0(\alpha), \dots, f_{2^d-1}(\alpha))$  for commutative ROABPs with the additional property that  $f_{i+1}(\alpha_1, \dots, \alpha_{d+1}) = \sigma(f_i(\alpha_1, \dots, \alpha_{d+1}))$  for each  $0 \leq i \leq \ell - 2$ .

We now examine the Forbes-Shpilka construction to incorporate these aspects. The construction is recursive. Suppose that we have the construction for degree  $2^{d-1}$ .

The hitting set for degree  $2^d$  is obtained in [9] by combining two copies of the hitting set for degree  $2^{d-1}$  using the following key technical lemma, [9, Lemma 3.7], rephrased below in somewhat different notation.

Let  $p_{\ell'}(v)$ ,  $1 \leq \ell' \leq r^2$  denote the Lagrange interpolation polynomials, defining a basis for univariate polynomials interpolating values from  $[r^2]$ . Given  $\beta_1, \dots, \beta_{r^2} \in \mathbb{F}$ , the Lagrange interpolation polynomials with respect to  $r^2$  and the  $\beta_i$ 's are the unique polynomials  $p_{\ell'}(v)$  of degree less than  $r^2$  such that

$$p_{\ell'}(\beta_i) = \begin{cases} 1 & \text{if } \ell' = i \\ 0 & \text{otherwise.} \end{cases}$$



► **Lemma 3.1** ([9, Lemma 3.7]). Let  $M_i$  and  $N_i, 0 \leq i \leq 2^{d-1} - 1$ , be  $r \times r$  matrices with entries from  $\mathbb{F}[x]$  of degree less than  $n$ . Let  $(f_0(u), f_1(u), \dots, f_{2^{d-1}-1}(u)) \in \mathbb{F}[u]$  be polynomials of degree at most  $m$ . Let  $\omega \in \mathbb{F}$  (or in an extension field) be an element of order at least  $(2^d nm)^2$ . Define polynomials in one indeterminate  $v$ :

$$\begin{aligned} f'_i &= \sum_{\ell'=1}^{r^2} f_i(\omega^{\ell'} \alpha_d) p_{\ell'}(v), \quad 0 \leq i \leq 2^{d-1} - 1 \\ f'_{i+2^{d-1}} &= \sum_{\ell'=1}^{r^2} f_i((\omega^{\ell'} \alpha_d)^\mu) p_{\ell'}(v), \quad 0 \leq i \leq 2^{d-1} - 1, \end{aligned}$$

where  $\mu = 2^{\kappa+d-1} + 1$  and  $\kappa$  is chosen such that  $2^\kappa \geq 2^d nm$ .

Then, for all but at most  $(2^d nmr)^2$  many values of  $\alpha_d$ , the  $\mathbb{F}$ -linear span of the matrix coefficients of the matrix product  $\prod_{i=0}^{2^{d-1}-1} M_i(f_i(x)) \prod_{i=0}^{2^{d-1}-1} N_i(f_i(y))$  is contained in the  $\mathbb{F}$ -linear span of the matrix coefficients of the product  $\prod_{i=0}^{2^{d-1}-1} M_i(f'_i(v)) \prod_{i=2^{d-1}}^{2^d-1} N_i(f'_i(v))$ .

Lemma 3.1 essentially gives the construction for going from the degree  $2^{d-1}$  hitting set generator to the degree  $2^d$  hitting set generator as proved in [9].

► **Remark 3.2.** In our modified construction we will use different roots of unity (for the element  $\omega$ ) for different stages of the recursive construction. In particular, roots of unity  $\omega_i, i < d$ , used in stages  $i < d$  will be of lower order. We explain below in detail, the choice of the parameters:  $\ell, \kappa, \omega_i$ , and  $\alpha_i$  for the modified construction.

We now adapt Lemma 3.1 to ensure the additional properties that will guarantee that the points of the hitting set are from  $D^n$ , for a suitably large cyclic division algebra  $D$ .

► **Theorem 3.3.** In deterministic quasipolynomial-time, we can construct a hitting set  $\mathcal{H}$  of size  $(nr\tilde{d})^{O(\log \tilde{d})}$  in  $D^n$  for the class of noncommutative polynomials in  $\mathbb{Q}\langle x_1, \dots, x_n \rangle$  computed by ABPs of width at most  $r$  with  $\tilde{d}$  many layers where the index of the cyclic division algebra  $D$ , the parameter  $\ell(> \tilde{d})$  is bounded by  $\text{poly}(r, n, \tilde{d})$ .

The detailed proof has been added in Section A of the appendix. Note that  $\mathcal{H}$  is a strong hitting set for any such noncommutative ABP.

## 4 Strong Hitting Set for Generalized ABPs over Division Algebra

In this section, we first define the notion of generalized ABPs and ABPs over a division algebra. Then we show the construction of a quasipolynomial-size strong hitting set for generalized ABPs over a division algebra such that any nonzero generalized ABP will evaluate to an invertible matrix at some point in the hitting set.

► **Definition 4.1.** A generalized ABP over the matrix algebra  $\text{Mat}_m(\mathbb{F})$  is defined in the same way as a noncommutative ABP, except for the fact that the linear forms labeling the edges are of the form  $\sum_{i=1}^n a_i x_i b_i$ , where  $a_i, b_i \in \text{Mat}_m(\mathbb{F})$ . Such an ABP computes a generalized polynomial in the generalized polynomial ring  $\text{Mat}_m(\mathbb{F})\langle X \rangle$ , where the polynomial is defined as the sum of products of the linear forms along all  $s$ -to- $t$  paths of the ABP, where  $s$  is the source node and  $t$  is the sink node of the directed acyclic graph underlying the ABP.

For a division algebra  $D$ , if the linear forms labeling the edges of the ABP are of the form  $\sum_{i=1}^n a_i x_i b_i, a_i, b_i \in D$  then it is a generalized ABP over the division algebra  $D$ .

Let  $D = (K/F, \sigma, z)$  (here  $F = \mathbb{Q}(z)$ ) be a cyclic division algebra of index  $\ell$  as defined in Section 2.3. Let  $\mathfrak{B} = \{C_{ij}\}_{i,j \in [\ell]}$  be the  $F$ -basis of  $D$  for  $i, j \in [\ell]$  as described in Section 2. Informally, our idea is to reduce the problem of finding strong hitting set for generalized ABPs over division algebra to the hitting set construction of a product of commutative ROABPs.

► **Lemma 4.2.** *For any nonzero generalized ABP  $B$  of degree  $d$  over  $D\langle \underline{x} \rangle$ , there exists a substitution for each  $x_k$  of the following form:*

$$M(x_k) = \begin{bmatrix} 0 & p_{k1} & 0 & \cdots & 0 \\ 0 & 0 & p_{k2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p_{k(d-1)} \\ p_{kd} & 0 & \cdots & 0 & 0 \end{bmatrix},$$

such that for each  $l \in [d]$ ,  $p_{kl}$  is in  $D$  and image of  $B$  is invertible on that substitution under the inclusion map  $a \mapsto I_d \otimes a$  where  $a \in D$ .

**Proof.** Let  $\ell$  be the index of the division algebra  $D$ . We first prove that for any nonzero generalized ABP  $B$  of degree  $d$  over  $D\langle \underline{x} \rangle$ , there exists a substitution for each  $x_k$  of the following form:

$$M(x_k) = \begin{bmatrix} 0 & q_{k1} & 0 & \cdots & 0 \\ 0 & 0 & q_{k2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & q_{k(d-1)} \\ q_{kd} & 0 & \cdots & 0 & 0 \end{bmatrix},$$

such that for each  $l \in [d]$ ,  $q_{kl}$  is in  $\text{Mat}_\ell(K)$  and the image of  $B$  is nonzero on that substitution with a block-diagonal structure. To evaluate  $B$  on such matrix substitution the coefficients  $a \in D$  (which have matrix representations in  $\text{Mat}_\ell(K)$ ) are fit to the correct dimension using the inclusion map  $\iota' : \text{Mat}_\ell(K) \rightarrow \text{Mat}_{d\ell}(K)$  where  $\iota'(a) = I_d \otimes a$ .

Let  $\psi$  be the substitution map that replaces each variables  $\{x_k\}_{k \in [n]}$  by an  $\ell \times \ell$  matrix of noncommuting variables  $\{z_{ijk}\}_{i,j \in [\ell], k \in [n]}$ . One can naturally extend the definition of  $\psi : \text{Mat}_\ell(K)\langle \underline{x} \rangle \rightarrow \text{Mat}_\ell(K)\langle \underline{z} \rangle$  i.e.  $\psi$  maps a generalized polynomial over matrix algebra  $\text{Mat}_\ell(K)$  to an  $\ell \times \ell$  matrix of noncommutative polynomials in  $K\langle \underline{z} \rangle$ . Indeed, the map  $\psi$  is identity preserving (see [19, Equation 3.10] for example).

Introduce a new set of commuting variables  $\tilde{Z} = \{\tilde{z}_{ijkl}\}$  where  $i, j \in [\ell]$ ,  $k \in [n]$  and  $l \in [d]$  and consider the following substitution for each  $x_k$ :

$$\tilde{Z}_k = \begin{bmatrix} 0 & \tilde{Z}_{k1} & 0 & \cdots & 0 \\ 0 & 0 & \tilde{Z}_{k2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \tilde{Z}_{k(d-1)} \\ \tilde{Z}_{kd} & 0 & \cdots & 0 & 0 \end{bmatrix},$$

where  $\tilde{Z}_{kl} = (\tilde{z}_{ijkl})_{1 \leq i,j \leq \ell}$ . In effect, the substitution of the  $x_k$  variables by the matrices  $\tilde{Z}_k$  is just set-multilinearization of  $\psi(B)$  position-wise and hence identity preserving.

What is the effect of this substitution on a degree- $d$  generalized word? To understand that consider a generalized word  $w = a_0 x_{k_1} a_1 x_{k_2} \cdots a_{d-1} x_{k_d} a_d$  where each  $a_i \in \text{Mat}_\ell(K)$ . Observe that  $w(\tilde{Z}_1, \dots, \tilde{Z}_n)$  is a block-diagonal matrix (using the inclusion map  $\iota'$ ) with  $(i, i)^{\text{th}}$  block

entry  $a_0 \tilde{Z}_{k_1 \pi_i(1)} a_1 \cdots a_{d-1} \tilde{Z}_{k_d \pi_i(d)} a_d$  where  $\pi_i, 1 \leq i \leq d$  is the cyclic permutations on  $[d]$  such that  $\pi_i(1) = i, \pi_i(2) = i + 1$  and so on. To elaborate further, we give an example for the degree three case in the full version.

Let  $B = \sum a_0 x_{k_1} a_1 x_{k_2} a_2 \cdots a_{d-1} x_{k_d} a_d$ . So the  $(i, i)^{th}$  entry of  $B(\tilde{Z}_1, \dots, \tilde{Z}_n)$  is

$$B^{\pi_i} = \sum a_0 \tilde{Z}_{k_1 \pi_i(1)} a_1 \tilde{Z}_{k_2 \pi_i(2)} \cdots a_{d-1} \tilde{Z}_{k_d \pi_i(d)} a_d.$$

Hence the final output matrix will be the following:

$$B(\tilde{Z}) = \begin{bmatrix} B^{\pi_1} & & & \\ & B^{\pi_2} & & \\ & & \ddots & \\ & & & B^{\pi_d} \end{bmatrix}.$$

We now claim the following.

▷ **Claim 4.3.** For each  $i \in [d]$ ,  $B^{\pi_i}$  is nonzero.

*Proof.* As  $B$  in  $D\langle x \rangle$  is nonzero and  $\psi$  is an identity preserving substitution,  $\psi(B) \in \text{Mat}_\ell(K\langle z \rangle)$  is also nonzero. We now consider the entry-wise set-multilinearization of  $\psi(B)$  with respect to the cyclic permutation  $\pi_i$  i.e. encoding any word using  $\pi_i(j)$  as the position index for the  $j^{th}$  position for each entry of  $\psi(B)$ . Notice that, it outputs the matrix  $B^{\pi_i}$ . Moreover, as  $\psi(B)$  is nonzero,  $B^{\pi_i}$  must be nonzero as set-multilinearization preserves identity. ◁

Hence, there exist substitutions  $q_{kl}$  from  $\text{Mat}_\ell(K)$  for the  $\tilde{Z}$  variables such that  $B$  is nonzero.

Now we use Proposition 2.13 which says that  $K$ -linear span of  $\mathfrak{B}$  is the entire matrix algebra  $\text{Mat}_\ell(K)$ . The above argument shows that if we replace each  $q_{kl}$  in  $M(x_k)$  by a linear combination

$$\sum_{i,j} y_{ijkl} C_{ij},$$

each diagonal block matrix of the output matrix obtained from the image of  $B$  on this evaluation is still nonzero over the  $\{y_{ijkl}\}$  variables. We now find substitutions for the  $Y$  variables from the ground field  $F$  to make each diagonal block matrix nonzero. As any  $F$ -linear combination of  $C_{ij}$  is in the division algebra, each such linear combinations is in  $D$ . So, define  $p_{kl} = \sum_{i,j} \beta_{ijkl} C_{ij} \in D$  where  $\beta_{ijkl}$  are the substitutions for  $y_{ijkl}$  variables from  $F$ . In fact the values for the variables  $\beta_{ijkl}$  can be found from  $\mathbb{Q}$  itself by a standard use of Polynomial Identity Lemma [6, 20, 17]. Notice that, each diagonal block will also be in  $D$ . Since each diagonal block matrix is nonzero and in  $D$  it is invertible. Therefore, the image of  $B$  is also invertible on the chosen matrix tuple. ◀

We are now ready to prove the main result of this section.

► **Theorem 4.4.** *Given the parameters  $n, \ell, r, d$ , in deterministic quasipolynomial-time we can construct strong hitting set  $\mathcal{H}'$  of size  $(nrd\ell)^{O(\log nd\ell)}$  for any nonzero generalized ABP  $B$  of degree  $d$  and width  $r$  over  $D\langle x \rangle$  where  $\ell$  is the index of  $D$ .*

Because of the space constraints, the proof is added in Section B of the appendix.

## 5 Putting all together

In this section we prove our main result, the construction of a hitting set for noncommutative rational formulas of inversion height two. An intermediate step is to construct a strong hitting set for rational formulas of inversion height one. En route to our proof, we crucially use the connection of rational identity testing with the identity testing of generalized ABPs. We make it explicit in Proposition 5.1.

Now we are ready to prove the main proposition.

► **Proposition 5.1.** *Let  $\mathbf{r}$  be a noncommutative rational formula over  $x_1, \dots, x_n$  of size  $s$  and  $(q_1, \dots, q_n) \in \text{Mat}_m^n(\mathbb{F})$  be a matrix tuple such that  $\mathbf{r}$  is defined on  $\underline{q}$ . Suppose,  $\mathbf{r}(\underline{x} + \underline{q})$  is a recognizable generalized series over  $\text{Mat}_m(\mathbb{F})\langle\langle \underline{x} \rangle\rangle$  with a linear representation  $(\mathbf{c}, M, \mathbf{b})$  of size at most  $2s$  over  $\text{Mat}_m(\mathbb{F})$ . Define  $S^{\{d\}} = \mathbf{c} \cdot M^d \cdot \mathbf{b}$  computing a generalized polynomial in  $\text{Mat}_m(\mathbb{F})\langle\langle \underline{x} \rangle\rangle$ . Then  $\mathbf{r}$  is nonzero in  $\mathbb{F}\langle\langle \underline{x} \rangle\rangle$  if and only if  $S^{\{d\}}$  is nonzero for some  $d \leq 2sm - 1$ . Additionally, there exists some  $N \leq \text{poly}(ksm)$  such that if  $|\mathbb{F}| > N$ , then for any  $T \subseteq \mathbb{F}$ ,  $|T| = N$  and for some matrix tuple  $(p_1, \dots, p_n) \in \text{Mat}_{km}^n(\mathbb{F})$ , evaluating  $S^{\{d\}}$  at  $\underline{p}$  under the inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{km}(\mathbb{F})$  (where  $\iota(a) = a \otimes I_k$ ) for each  $d \leq 2sm - 1$ ,*

1. *If  $S^{\{d\}}$  evaluates nonzero for some  $d$ , then there exists an  $\alpha \in T$  such that  $\mathbf{r}$  is nonzero at the following matrix tuple:*

$$(\alpha p_1 + q_1 \otimes I_k, \dots, \alpha p_n + q_n \otimes I_k).$$

2. *If  $S^{\{d\}}(\underline{p})$  is invertible for some  $d$ , there exists an  $\alpha \in T$  such that  $\mathbf{r}$  is invertible at the following matrix tuple:*

$$(\alpha p_1 + q_1 \otimes I_k, \dots, \alpha p_n + q_n \otimes I_k).$$

The proof is added in Section C of the appendix.

### Strong hitting set for rational formulas of inversion height one

We now show the construction of a strong hitting set for noncommutative rational formulas of inversion height one.

► **Theorem 5.2.** *Given  $n, s$ , we can construct a strong hitting set  $\tilde{\mathcal{H}}_1$  of size  $(ns)^{O(\log ns)}$  over  $\text{Mat}_d^n(K)$  for the class of noncommutative rational formulas  $\mathbf{r} \in \mathbb{Q}\langle\langle x_1, \dots, x_n \rangle\rangle$  of size  $s$  and of inversion height one. The parameter  $d'$  is  $\text{poly}(n, s)$  and  $K = \mathbb{Q}(\omega, z)$  is the extension field by adjoining a primitive root of unity  $\omega$  of order  $\ell$  where  $\ell = \text{poly}(n, s)$ .*

**Proof.** Let  $\mathbf{r}(\underline{x})$  be a rational formula of inversion height one in  $\mathbb{Q}\langle\langle \underline{x} \rangle\rangle$  of size  $s$ . Let  $h_1, \dots, h_k$  be all the sub-formulas input to the inverse gates in the rational formula for  $\mathbf{r}$ . Consider the noncommutative formula  $h = h_1 h_2 \cdots h_k$  in  $\mathbb{Q}\langle\langle \underline{x} \rangle\rangle$  which is of size at most  $s$  and degree is also bounded by  $s$ .

By Theorem 3.3, we construct a hitting set  $\mathcal{H}_0$  in  $D^n$  where  $D = (K/F, \sigma, z)$  is a cyclic division algebra of index  $\ell = \text{poly}(n, s)$  for noncommutative ABPs in  $\mathbb{Q}\langle\langle \underline{x} \rangle\rangle$  of width and layers at most  $s$ . Then there is a point  $\underline{q} \in \mathcal{H}_0$  such that  $h(\underline{q})$  is invertible and hence  $\mathbf{r}(\underline{q})$  is defined.

Following Theorem 2.8, if  $\mathbf{r}(\underline{x})$  is nonzero then  $\mathbf{r}(\underline{x} + \underline{q})$  can be represented as a nonzero recognizable generalized series. Indeed, it is a recognizable generalized series over  $D$  following Theorem 2.8. Moreover, using the second part of Proposition 5.1, to obtain a strong hitting set for  $\mathbf{r}(\underline{x})$ , it suffices to find a strong hitting set of a generalized ABP over  $D$  of width

$r \leq 2s$  and degree  $d \leq 2s\ell - 1$ . We now use the strong hitting set  $\mathcal{H}_1$  in  $\text{Mat}_{d\ell}^n(K)$  (recall that  $K = \mathbb{Q}(z, \omega)$  where  $\omega$  is the primitive root of unity of order  $\ell$ ) for generalized ABPs of degree  $d$  over  $D$  (here  $\ell$  is the index of  $D$ ) obtained in Theorem 4.4. Inspecting the proof of Proposition 5.1, the final quasipolynomial-size hitting set is the following:

$$\widehat{\mathcal{H}}_1 = \{\alpha \underline{p} + \underline{q} \otimes I_d : \underline{p} \in \mathcal{H}_1, \underline{q} \in \mathcal{H}_0, \alpha \in T\} \subseteq \text{Mat}_{d\ell}^n(K).$$

Here  $T \subseteq \mathbb{Q}$  of size  $\text{poly}(n, s)$  that we can find efficiently.  $\blacktriangleleft$

### Hitting set for rational formulas of inversion height two

We are now ready to prove our main theorem.

**Proof of Theorem 1.1.** Let  $\mathbf{r}(x)$  be a rational formula of inversion height two in  $\mathbb{Q}\langle x \rangle$  of size  $s$ . Let  $\mathcal{F}$  be the collection of all those inverse gates in the formula such that for every  $\mathbf{g} \in \mathcal{F}$ , the path from the root to  $\mathbf{g}$  does not contain any inverse gate. For each  $\mathbf{g}_i \in \mathcal{F}$ , let  $h_i$  be the sub-formula input to  $\mathbf{g}_i$ . Consider the formula  $h = h_1 h_2 \cdots h_k$  (where  $k = |\mathcal{F}|$ ) which is of size at most  $s$  since for each  $i, j$ ,  $h_i$  and  $h_j$  are disjoint.  $h$  is of inversion height one. By Theorem 5.2, we construct a strong hitting set  $\widehat{\mathcal{H}}_1$  in  $\text{Mat}_d(K)$  where  $d = \text{poly}(n, s)$ . Then there is a point  $\underline{q} \in \widehat{\mathcal{H}}_1$  such that  $h(\underline{q})$  is invertible and hence  $\mathbf{r}(\underline{q})$  is defined.

Following Theorem 2.8, if  $\mathbf{r}(x)$  is nonzero then  $\mathbf{r}(x + \underline{q})$  can be represented as a nonzero recognizable generalized series over  $\text{Mat}_d(K)$ . Moreover, using the first part of the proof of Proposition 5.1, to obtain a hitting set for  $\mathbf{r}(x)$ , it suffices to find a hitting set for generalized ABP  $B$  over  $\text{Mat}_d(K)$  of width  $r \leq 2s$  and degree  $\hat{d} \leq 2sd - 1$ , the degree- $\hat{d}$  truncated part of the generalized series  $\mathbf{r}(x + \underline{q})$ . We recall the substitution map  $\psi$  from Proposition 5.1 and consider  $\psi(B)$ . Each entry of  $\psi(B)$  is computable by a noncommutative ABP of width  $2sd$  and degree  $\hat{d}$  over  $Z = \{z_{i,j,k'}\}$  variables. Let  $\mathcal{H}_{FS} \subseteq \text{Mat}_{\hat{d}+1}^{nd^2}(K)$  be the hitting set for ABPs of width  $2sd$  and of degree  $\hat{d}$  over  $nd^2$  many variables obtained from Theorem 2.5. We now define  $\tilde{\mathcal{H}}_{FS} \in \text{Mat}_{d(\hat{d}+1)}^n(K)$  in the following way. For every matrix substitution in  $\mathcal{H}_{FS}$ , define a matrix substitution for each  $x_{k'}$  as a  $d(\hat{d}+1) \times d(\hat{d}+1)$  matrix which can be thought of as a  $d \times d$  block matrix whose  $(i, j)^{\text{th}}$  block is the matrix substituted for  $z_{i,j,k'}$  variable from  $\mathcal{H}_{FS}$ . It follows that  $\mathcal{H}_{FS}$  is a hitting set of  $B$  under the inclusion map  $a \mapsto a \otimes I_{\hat{d}+1}$ . To explain the purpose of the inclusion map, see the remark in the full version.

Inspecting the proof of Proposition 5.1, we can now find a subset  $T \subseteq \mathbb{Q}$  of size  $\text{poly}(n, s)$  and the final quasipolynomial-size hitting set is the following:

$$\mathcal{H}_2 = \{\alpha \underline{p} + \underline{q} \otimes I_{\hat{d}+1} : \underline{p} \in \tilde{\mathcal{H}}_{FS}, \underline{q} \in \widehat{\mathcal{H}}_1, \alpha \in T\}. \quad (2)$$

To get a hitting set over  $\mathbb{Q}$  itself, we add a discussion in subsection C.1 of the appendix.  $\blacktriangleleft$

---

### References

- 1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. doi:10.1137/140975103.
- 2 S.A Amitsur. Rational identities and applications to algebra and geometry. *Journal of Algebra*, 3(3):304–359, 1966. doi:10.1016/0021-8693(66)90004-4.
- 3 V. Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. A Special Case of Rational Identity Testing and the Brešar-Klep Theorem. In Javier Esparza and Daniel Král, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.10.

- 4 George M Bergman. Rational relations and rational identities in division rings. *Journal of Algebra*, 43(1):252–266, 1976. doi:10.1016/0021-8693(76)90159-9.
- 5 J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. URL: [https://books.google.co.in/books?id=LL8Nhn72I\\_8C](https://books.google.co.in/books?id=LL8Nhn72I_8C).
- 6 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.
- 7 Harm Derksen and Visu Makam. Polynomial degree bounds for matrix semi-invariants. *Advances in Mathematics*, 310:44–63, 2017. doi:10.1016/j.aim.2017.01.018.
- 8 Samuel Eilenberg. *Automata, Languages, and Machines (Vol A)*. Pure and Applied Mathematics. Academic Press, 1974.
- 9 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013. doi:10.1109/FOCS.2013.34.
- 10 Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 109–117. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.95.
- 11 Pavel Hrubeš and Avi Wigderson. Non-commutative arithmetic circuits with division. *Theory of Computing*, 11(14):357–393, 2015. doi:10.4086/toc.2015.v011a014.
- 12 Loo-Keng Hua. Some properties of a sfield. *Proceedings of the National Academy of Sciences of the United States of America*, 35(9):533–537, 1949. URL: <http://www.jstor.org/stable/88328>.
- 13 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *computational complexity*, 27(4):561–593, December 2018. doi:10.1007/s00037-018-0165-7.
- 14 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.
- 15 T.Y. Lam. *A First Course in Noncommutative Rings (Second Edition)*. Graduate Texts in Mathematics. Springer, 2001.
- 16 Louis Halle Rowen. *Polynomial identities in ring theory*. Pure and Applied Mathematics. Academic Press, 1980. URL: <http://gen.lib.rus.ec/book/index.php?md5=bde982110d09e6199643e04da0558459>.
- 17 Jacob T. Schwartz. Fast probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.
- 18 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. URL: <http://eudml.org/doc/151394>.
- 19 Jurij Volčič. Matrix coefficient realization theory of noncommutative rational functions. *Journal of Algebra*, 499:397–437, April 2018. doi:10.1016/j.jalgebra.2017.12.009.
- 20 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.

## A Missing Proof from Section 3

**Proof of Theorem 3.3.** Let  $\ell$  be the index of  $D$ . We set  $\ell = 2^L$ , where  $L$  is to be determined below. Thus,  $\omega = e^{\frac{2\pi}{2^L}}$  is a  $2^L$ -th primitive root of unity. Let  $F = \mathbb{Q}(z)$  and  $K = F(\omega, z)$  which gives the cyclic division algebra  $D = (K/F, \sigma, z)$  where we fix the  $K$ -automorphism  $\sigma$  as

$$\sigma(\omega) = \omega^{2^\kappa + 1},$$

and  $\kappa$  will be suitably chosen in the following analysis, fulfilling the constraints of Lemma 3.1 and some additional requirements.

Let  $\omega_i = \omega^{2^{a_i}}$  for  $a_1 > a_2 > \dots > a_d > 0$ , where  $a_i$  are positive integers to be chosen. Let  $K_i = F(\omega_i)$  be the cyclic Galois extension for  $1 \leq i \leq d$ . This gives a tower of extension fields

$$F \subset F(\omega_1) \subset F(\omega_2) \subset \dots \subset F(\omega_d) \subset F(\omega).$$

We require two properties of  $\omega_i, 1 \leq i \leq d$ .

1. For the hitting set generator  $\mathcal{G}_i$  we will choose the root of unity as  $\omega_i$  and the variable  $\alpha_i$  will take values only in the set  $W_i = \{\omega_i^j \mid 1 \leq j \leq 2^{L-a_i}\}$ .
2. We require that the  $K$ -automorphism  $\sigma$  has the property that for all  $1 \leq i \leq d$  the map  $\sigma^{2^i}$  fixes  $\omega_i$ . In fact we will ensure that  $\sigma^{2^i}$  has  $F(\omega_i)$  as its fixed field.

We take up the second property. As  $\sigma(\omega) = \omega^{2^\kappa+1}$ , we have  $\sigma(\omega_i) = \omega^{2^{a_i}(2^\kappa+1)}$ . Therefore

$$\sigma^{2^i}(\omega_i) = \omega^{2^{a_i}(2^\kappa+1)^{2^i}}.$$

Now,  $(2^\kappa + 1)^{2^i} = \sum_{j=0}^{2^i} \binom{2^i}{j} 2^{\kappa j}$ . Choosing  $\kappa = L/2$ , we have  $\omega^{2^{\kappa j}} = 1$  for  $j \geq 2$ . Therefore,

$$\sigma^{2^i}(\omega_i) = \omega^{2^{a_i}(2^{i+\kappa}+1)} = \omega_i \cdot \omega^{2^{a_i+i+\kappa}}.$$

We can set  $a_i + i + \kappa = L$  for  $1 \leq i \leq d$  to ensure that  $\sigma^{2^i}$  fixes  $\omega_i$ . Putting  $L = 2\kappa$ , we obtain

$$a_i = \kappa - i \text{ for } 1 \leq i \leq d. \tag{3}$$

It remains to choose  $\kappa$ . In the construction of our hitting set generator  $\mathcal{G}_i$ , the parameter  $\alpha_i$  will take values only in  $W_i$  defined above. We note that  $|W_i| = 2^{L-a_i} = 2^{\kappa+i}$ . By Lemma 3.1 there are at most  $(2^d nmr)^2$  many bad values of  $\alpha_i$  for any  $i$ . Thus, it suffices to choose  $\kappa$  such that  $2^\kappa > (2^d nmr)^2$ . It suffices to set

$$\kappa = 2d + \lceil 2 \log_2(nmr) \rceil + 1.$$

The choice of  $\kappa$  determines the value of parameter  $\mu$  in Lemma 3.1.

Coming back to the modified construction of  $\mathcal{G}_d$ , inductively, we can assume that the hitting set generator  $\mathcal{G}_{d-1} : (\alpha_1, \dots, \alpha_{d-1}, u) \mapsto (f_0(u), f_1(u), \dots, f_{2^{d-1}-1}(u))$  (where for  $0 \leq i \leq 2^{d-1}-1$ , the polynomial  $f_i(u) \in K_{d-1}[u]$ ) has that property. Namely, suppose  $f_{i+1}(u) = \sigma(f_i(u))$  holds for all  $i \leq 2^{d-1}-2$ . Now define  $\mathcal{G}_d$  using Lemma 3.1. Since  $p_{\ell'}(v)$  has only integer coefficients,  $\sigma(p_{\ell'}(v)) = p_{\ell'}(v)$ . Therefore, for  $0 \leq i \leq 2^{d-1}-2$  and for  $2^{d-1} \leq i \leq 2^d-2$  we have  $f'_{i+1}(v) = \sigma(f'_i(v))$ .

Now, consider  $i = 2^{d-1}-1$ . We need to ensure that  $\sigma(f'_{2^{d-1}-1}(v)) = f'_{2^{d-1}}(v)$ . Equivalently, we need to ensure that

$$\sigma \left( \sum_{\ell'=1}^{r^2} f_{2^{d-1}-1}(\omega_d^{\ell'} \alpha_d) p_{\ell'}(v) \right) = \sum_{\ell'=1}^{r^2} f_1((\omega_d^{\ell'} \alpha_d)^\mu) p_{\ell'}(v).$$

This is enforced by requiring that

$$\sigma^{2^{d-1}} \left( \sum_{\ell'=1}^{r^2} f_1(\omega_d^{\ell'} \alpha_d) p_{\ell'}(v) \right) = \sum_{\ell'=1}^{r^2} f_1((\omega_d^{\ell'} \alpha_d)^\mu) p_{\ell'}(v).$$



Since  $\alpha_d$  will be chosen from  $W_d$  (all powers of  $\omega_d$ ), we can write  $\omega_d^{\ell'} \alpha_d = \omega_d^j$  for some  $j$ . Now,  $\sigma^{2^{d-1}} f_1(\omega_d^j) = f_1(\sigma^{2^{d-1}}(\omega_d^j))$  as  $\sigma^{2^{d-1}}$  fixes all coefficients of  $f_1$  (because  $f_1(u) \in K_{d-1}[u]$ ). Now,

$$\sigma^{2^{d-1}}(\omega_d^j) = \omega_d^{j \cdot (2^\kappa + 1)^{2^{d-1}}} = \omega_d^{j(1+2^{d-1+\kappa})} = (\omega_d^\ell \alpha_d)^\mu,$$

which verifies the choice of  $\mu$  in Lemma 3.1 is  $1 + 2^{d-1+\kappa}$ .

As shown in [9], the parameter  $v$  (whose place holder is  $\alpha_{d+1}$  in the description of  $\mathcal{G}_d$ ) should vary over a set of size  $\text{poly}(2^d, n, m, r)$ . This way we ensure that  $f_{i+1} = \sigma(f_i)$  for  $0 \leq i \leq 2^d - 2$ . Now define  $f_{2^d+j} = \sigma(f_{2^d+j-1})$  for  $0 \leq j \leq \ell - 2^d - 1$ . The fact that  $\mathcal{G}_d$  is indeed a generator follows from the span preserving property and the proof is identical to the proof of [9, Lemma 3.19]. ◀

## B Missing proof from Section 4

**Proof of Theorem 4.4.** By Lemma 4.2, we know that there exists matrix tuple  $(p_1, \dots, p_n)$  in  $\text{Mat}_{d\ell}^n(K)$  of the following form

$$p_k = M(x_k) = \begin{bmatrix} 0 & p_{k1} & 0 & \cdots & 0 \\ 0 & 0 & p_{k2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p_{k(d-1)} \\ p_{kd} & 0 & \cdots & 0 & 0 \end{bmatrix},$$

where each  $p_{kl} \in D : 1 \leq k \leq n, 1 \leq l \leq d$  such that  $B(p_1, p_2, \dots, p_n)$  is an invertible matrix.

Write each  $p_{kl}$  as  $p_{kl} = \sum_{i,j \in [\ell]} y_{ijkl} C_{ij}$  for some commuting indeterminates  $Y = \{y_{ijkl}\}$  whose values we need to determine. On such a substitution,  $B$  evaluates to the following matrix:

$$\begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_d \end{bmatrix}.$$

where each  $B_l, 1 \leq l \leq d$  is nonzero by Lemma 4.2 (using the inclusion map  $\iota'$ ). We now observe the following.

▷ **Claim B.1.** For each  $l \in [d]$ ,  $B_l$  is a matrix of commutative set-multilinear ABPs each of width  $r\ell$ .

**Proof.** To see this, consider the matrix  $B_1$ . We can think of  $B_1$  as the matrix obtained by substituting  $p_{kl}$  for  $x_k$  in layer  $l$  of the input generalized ABP  $B$  over  $D$  of index  $\ell$ . This computation can also be thought of by making  $\ell$ -many copies of each node in  $B$ .

More precisely, each coefficient  $a \in D$  in  $B$  has a  $\ell \times \ell$  matrix representation over  $K$ . Now consider each edge  $\sum_{k=1}^n a_k x_k b_k$  between the layer  $l$  and  $l+1$ . Since  $x_k$  is replaced by  $p_{kl}$  and  $a_k, b_k \in D$ , this edge can be replaced by an  $\ell \times \ell$  bipartite graph such that for each  $i, j \in [\ell]$ , the edge connecting the  $i^{\text{th}}$  node (from left) to the  $j^{\text{th}}$  node (to right) is labeled by the  $(i, j)^{\text{th}}$  entry of the product of  $a_k p_{kl} b_k$ , a linear form over  $K[Y]$ . Clearly, it produces an  $\ell$ -input  $\ell$ -output setmultilinear ABP of width  $r\ell$ . Therefore, each entry in  $B_1$  is computed by a set-multilinear ABP of width  $r\ell$  and degree  $d$ . The situation for other  $B_l : 2 \leq l \leq d$  are similar. ◀



Therefore we can use a hitting set generator for commutative set-multilinear ABPs of width  $r\ell$  and degree  $d$  to obtain a point such that the image for each  $B_l$  is nonzero on that evaluation.

However, our goal is to obtain an invertible image for the image of  $B$ . In other words, we want a substitution of  $Y$  variables for which each  $B_l$  would be invertible. Notice that if for substitution of  $Y$  variables from  $F$  at least one entry of  $B_l$  is nonzero, then the matrix  $B_l$  is also invertible as the image of  $B_l$  is a nonzero element in  $D$ . Hence, to obtain a strong hitting set for the input generalized ABP over  $D$  (equivalently, to obtain a substitution on which the product of the matrices  $B_l, 1 \leq l \leq d$  is invertible), it suffices to obtain a hitting set for the product of set-multilinear ABPs (product of one of the nonzero entries of each  $B_l$ ).

We do this by first converting each set-multilinear ABP to an ROABP encoding each  $y_{ijkl}$  to  $v_l^{(\ell+1)^2 i + (\ell+1)j + k_2}$ . By construction each encoded  $B_l$  yields a known variable partition for the corresponding ROABP. More precisely, for each  $l$  the ROABP computed in the  $(l, l)^{th}$  diagonal block follows the variable partition:

$$v_l < v_{l+1} < \dots < v_d < v_1 < \dots < v_{l-1}.$$

Therefore, for each  $nd\ell^2$ -variate ROABP of degree  $d$  and of width  $\ell r$  computed in each diagonal block, we can use the hitting set generator of Theorem 2.4. Now, for a  $d$ -fold product of such ROABPs of different but known orders, the same hitting set generator will also work. This is because we can ensure that the hitting set generator of Theorem 2.4 has the property that more than  $1 - 1/d$  fraction of seeds for the generator works for a given ROABP with the above parameters. Then, by a standard union bound argument it follows that there is a choice of seed for the generator that will hit the product of these  $d$  many ROABPs. Notice that the ROABPs are all  $nd\ell^2$ -variate,  $d$ -degree, and of width  $\ell r$ . Thus by Theorem 2.4, the size of the hitting set for them is  $(nd\ell r)^{O(\log nd\ell)}$ . Hence, we can now find a substitution for the  $v_l$  variables such that each  $B_l$  is invertible, hence  $B$  is also invertible.

This gives us a hitting set  $\mathcal{H}$  under the inclusion map  $\iota' : \text{Mat}_\ell(K) \rightarrow \text{Mat}_{d\ell}(K)$  where  $\iota'(a) = I_d \otimes a$ . However for the purpose of Section 5, we find a hitting set  $\mathcal{H}'$  under the inclusion map  $\iota : \text{Mat}_\ell(K) \rightarrow \text{Mat}_{d\ell}(K)$  where  $\iota(a) = a \otimes I_d$ . Although it is technically possible to work with two inclusion maps thanks to Remark 2.9, we find it mathematically nicer to work with a single inclusion map. For this we explicitly find a permutation matrix  $q_0$  of dimension  $d\ell$  such that  $q_0(I_d \otimes a)q_0^{-1} = a \otimes I_d$  for all  $a \in \text{Mat}_\ell(K)$ . Once we find  $q_0$ , the final hitting set can be defined as  $\mathcal{H}' = \{(q_0 p_1 q_0^{-1}, \dots, q_0 p_n q_0^{-1}) \mid \underline{p} \in \mathcal{H}\}$ . To see this, let

$$B = \sum a_0 x_{k_1} a_1 \dots a_{d-1} x_{k_d} a_d.$$

Let  $M = B(q_1, \dots, q_n)$  is an invertible matrix for  $\underline{q} \in \mathcal{H}$ . We know that,

$$\sum (I_d \otimes a_0) q_{k_1} (I_d \otimes a_1) \dots (I_d \otimes a_{d-1}) q_{k_d} (I_d \otimes a_d) = M. \quad (4)$$

By conjugating  $M$  with  $q_0$ , obtain the following:

$$\sum q_0 (I_d \otimes a_0) q_0^{-1} q'_{k_1} q_0 (I_d \otimes a_1) q_0^{-1} \dots q_0 (I_d \otimes a_{d-1}) q_0^{-1} q'_{k_d} q_0 (I_d \otimes a_d) q_0^{-1} = q_0 M q_0^{-1}, \quad (5)$$

where  $q'_{k_j} = q_0 q_{k_j} q_0^{-1}$ . In other words  $B(q'_1, \dots, q'_n)$  is the invertible matrix  $M' = q_0 M q_0^{-1}$  under the inclusion map  $\iota$ . In the full version, we show that the permutation matrix  $q_0$  can be constructed explicitly.  $\blacktriangleleft$

---

<sup>2</sup> Note that by the choice,  $\ell$  is larger than  $n$  and  $d$ .

## C

 Missing Proofs from Section 5

► **Lemma C.1.** Let  $\tau \in \mathbb{F}\langle x \rangle$  be a rational formula of size  $s$ . Let  $\underline{p} = (p_1, \dots, p_n) \in \text{Mat}_m^n(\mathbb{F}(t_1, t_2))$  be an  $n$ -tuple of matrix of bivariate rational functions where the degrees of the numerator and denominator polynomials over  $t_1, t_2$  at each entry are at most  $d'$  and  $\tau$  is defined at  $\underline{p}$ . Then, evaluating  $\tau$  on  $\underline{p}$  outputs  $\tau(\underline{p}) \in \text{Mat}_m(\mathbb{F}(t_1, t_2))$  such that each entry of the output matrix is of form  $\frac{P(t_1, t_2)}{Q(t_1, t_2)}$  where  $P$  and  $Q$  are bivariate polynomials of degree at most  $O(smd')$ .

**Proof.** As already stated in Section 1 that  $\tau$  has a linear pencil  $L$  of size (at most)  $2s$  such that for any tuple  $\underline{p}$ ,  $\tau(\underline{p})$  is defined if and only if  $L(\underline{p})$  is invertible [11, Proposition 7.1]. Moreover,  $\tau(\underline{p}) = L_{i,j}^{-1}(\underline{p})$  for some  $(i, j)^{th}$  entry of  $L$  i.e.  $\tau(\underline{p})$  is the  $(i, j)^{th}$  block of  $L^{-1}(\underline{p})$  thinking of it as a  $2s \times 2s$  block matrix where each block is of size  $m$ . Notice that, if  $L = \sum_{i=1}^n A_i x_i$ , then  $L(\underline{p}) = \sum_{i=1}^n A_i \otimes p_i$ . Therefore,  $L(\underline{p})$  is a  $2sm \times 2sm$  matrix such that each entry is a polynomial over  $t_1, t_2$  of degree at most  $d'$ . From the standard computation of matrix inverse, it is immediate that each entry of  $L^{-1}(\underline{p})$  (therefore, each entry of  $\tau(\underline{p})$ ) is a commutative rational function such that the numerator and the denominator are bivariate polynomials over  $t_1, t_2$  with degree bound  $O(smd')$ . ◀

**Proof of Proposition 5.1.** By Theorem 2.8, we know that  $\tau(\underline{x})$  is zero if and only if  $\tau(\underline{x} + \underline{q})$  is zero. Let  $Z = \{z_{i,j,k'}\}_{1 \leq i,j \leq m, 1 \leq k' \leq n}$  be a set of noncommuting variables. Consider a substitution map  $\psi$  that substitutes each variable  $x_{k'}, 1 \leq k' \leq n$  of  $\tau(\underline{x} + \underline{q})$  by an  $m \times m$  matrix  $Z_{k'}$  consisting of fresh noncommutative variables  $\{z_{i,j,k'}\}_{1 \leq i,j \leq m}$ . Consider  $\tau(\psi(\underline{x}) + \underline{q})$  and observe that,  $\psi$  is an identity preserving and degree preserving substitution.

From the definition,  $\tau(\underline{x} + \underline{q}) = \mathbf{c}(I - M)^{-1}\mathbf{b}$  where  $M$  is of size at most  $2s$  by Theorem 2.8. Therefore,  $\tau(\psi(\underline{x}) + \underline{q}) = \mathbf{c}(I - \psi(M))^{-1}\mathbf{b}$ , where it is convenient to think of  $\mathbf{c}$  (respectively  $\mathbf{b}$ ) as an  $m \times 2ms$  (resp.  $2ms \times m$ ) rectangular matrix  $C$  (resp.  $B$ ), and  $\psi(M)$  as  $2ms \times 2ms$  matrix.

Observe that, for the matrix  $\tau(\psi(\underline{x}) + \underline{q})$ , the  $(i, j)^{th}$  entry is the following recognizable series which has linear representation of size at most  $2sm$ :

$$\mathbf{C}_i(I - \psi(M))^{-1}\mathbf{B}_j$$

where  $\mathbf{C}_i$  is the  $i^{th}$  row of  $C$  and  $\mathbf{B}_j$  is the  $j^{th}$  column of  $B$ . If  $\tau(\underline{x} + \underline{q})$  is nonzero, then some  $(i, j)^{th}$  entry of  $\tau(\psi(\underline{x}) + \underline{q})$  is also nonzero. Clearly, the degree- $d$  truncated part of the matrix  $\tau(\psi(\underline{x}) + \underline{q})$  is  $\psi(S^{\{d\}})$ . Moreover, for the matrix  $\psi(S^{\{d\}})$ , each entry is computed by a noncommutative ABP of width  $2sm$  and depth  $d$  over  $Z$  variables. By Theorem 2.6, there exists a minimum  $d \leq 2sm - 1$  such that  $\psi(S^{\{d\}})$  and thus  $S^{\{d\}}$  is nonzero. Clearly  $S^{\{d\}}$  is computable by a generalized ABP.

**Proof of Part (1).** Now, for some matrix tuple  $(p_1, \dots, p_n) \in \text{Mat}_{km}(\mathbb{F})$ , let  $d \leq 2sm - 1$  such that  $S^{\{d\}}$  is nonzero at  $\underline{p}$  under the inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{km}(\mathbb{F})$  given by  $\iota : a \rightarrow a \otimes I_k$ . Consider the evaluation of  $\tau$  at  $(tp_1 + q_1 \otimes I_k, \dots, tp_n + q_n \otimes I_k)$  where  $t$  is some commuting indeterminate. Let  $M(t) = \tau(tp_1 + q_1 \otimes I_k, \dots, tp_n + q_n \otimes I_k)$ . We now interpret  $M(t)$  in two ways. First, think of  $M(t)$  as the evaluation of the generalized series  $\tau(\underline{x} + \underline{q})$  at  $(tp_1, \dots, tp_n)$  under the inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{km}(\mathbb{F})$  given by  $\iota : a \rightarrow a \otimes I_k$ . We can write  $M(t) = t^d S^{\{d\}}(\underline{p}) + M'(t)$  where  $t$ -degree of each term of the matrix  $M'(t)$  is strictly more than  $d$ . Therefore,  $M(t)$  is nonzero.

Another way to interpret  $M(t)$  is to evaluate the rational formula  $\tau$  on  $(tp_1 + q_1 \otimes I_k, \dots, tp_n + q_n \otimes I_k)$ . Since  $\tau$  is a rational formula of size  $s$ , each entry of the matrix  $M(t)$  is an element of the function field  $\mathbb{F}(t)$ . Moreover by Lemma C.1, the  $t$ -degrees of the numerator

and denominator polynomials of each such commutative rational expression computed at all the nodes, are bounded by  $\hat{d} = \text{poly}(ksm)$ . Therefore, the final choice of the parameter  $t$  should be such that it avoids the zeros of the numerator and denominator polynomials involved in the computation of  $M(t)$ . This is clearly possible by varying  $t$  over a  $\text{poly}(ksm)$  size set  $T \subseteq \mathbb{F}$ .

**Proof of Part (2).** The proof of the second part is similar. For some matrix tuple  $(p_1, \dots, p_n) \in \text{Mat}_{km}(\mathbb{F})$ , let  $d \leq 2sm - 1$  such that  $S^{\{d\}}$  is invertible at  $\underline{p}$  under the inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{km}(\mathbb{F})$  given by  $\iota : a \rightarrow a \otimes I_k$ . Let  $M(t) = \mathbf{r}(tp_1 + q_1 \otimes I_k, \dots, tp_n + q_n \otimes I_k)$ . As before, consider two interpretations of  $M(t)$ . Think of  $M(t)$  as the evaluation of the generalized series  $\mathbf{r}(\underline{x} + \underline{q})$  at  $(tp_1, \dots, tp_n)$  again under the inclusion map  $\iota : \text{Mat}_m(\mathbb{F}) \rightarrow \text{Mat}_{km}(\mathbb{F})$  given by  $\iota : a \rightarrow a \otimes I_k$ . We write  $\det M(t) = t^{mkd} \det S^{\{d\}}(\underline{p}) + M''(t)$  where  $t$ -degree of each term of the matrix  $M''(t)$  is strictly more than  $mkd$ . Therefore,  $\det M(t)$  is nonzero.

Interpret  $M(t)$  as the evaluation of the rational formula  $\mathbf{r}$  on  $(tp_1 + q_1 \otimes I_k, \dots, tp_n + q_n \otimes I_k)$ . Since  $\mathbf{r}$  is a rational formula of size  $s$ , each entry of the matrix  $M(t)$  is an element of the function field  $\mathbb{F}(t)$ . Again by Lemma C.1, the  $t$ -degrees of each numerator and denominator polynomial involved in the computation of  $M(t)$  and  $\det M(t)$  is also bounded by  $\text{poly}(ksm)$ . Therefore, the final choice of the parameter  $t$  should be such that it avoids the zeros of all such the numerator and denominator polynomials involved in the computation of  $M(t)$  and  $\det(M(t))$ . This is clearly possible by varying  $t$  over any  $\text{poly}(ksm)$  size set  $T \subseteq \mathbb{F}$ .

Final substitution is of the following form in both the cases:

$$\{(\alpha p_1 + q_1 \otimes I_k, \dots, \alpha p_n + q_n \otimes I_k)\}, \quad (6)$$

for some suitably chosen  $\alpha \in T \subseteq \mathbb{F}$ . ◀

## C.1 Obtaining a hitting set over Rationals

Now we discuss how to obtain a hitting set over  $\mathbb{Q}$  itself. In the hitting set points suppose we replace  $\omega$  and  $z$  by commuting indeterminates  $t_1, t_2$  of degree bounded by  $\ell$ . Then, for any nonzero rational formula  $\mathbf{r}$  of size  $s$  there is a matrix tuple in the hitting set on which  $\mathbf{r}$  evaluates to a nonzero matrix  $M(t_1, t_2)$  of dimension  $\text{poly}(n, s)$  over the commutative function field  $\mathbb{Q}(t_1, t_2)$ . By Lemma C.1, each entry of  $M(t_1, t_2)$  is a rational expression of the form  $a/b$ , where  $a$  and  $b$  are polynomials in  $t_1$  and  $t_2$  and the degrees of both  $a$  and  $b$  are bounded by  $\text{poly}(n, s)$ . Hence by the argument sketched in Proposition 5.1, we can vary the parameters  $t_1, t_2$  over a sufficiently large set  $\tilde{T} \subseteq \mathbb{Q}$  of size  $\text{poly}(n, s)$  such that we avoid the roots of the numerator and denominator polynomials involved in the computation. This gives our final hitting set  $\tilde{\mathcal{H}}_2 = \{\underline{q}'(\alpha_1, \alpha_2) : \underline{q}'(\omega, z) \in \mathcal{H}_2, (\alpha_1, \alpha_2) \in \tilde{T} \times \tilde{T}\}$ .

# Sampling from Potts on Random Graphs of Unbounded Degree via Random-Cluster Dynamics

Antonio Blanca ✉

Department of CSE, Pennsylvania State University, University Park, PA, USA

Reza Gheissari ✉

Department of Statistics and EECS, University of California, Berkeley, CA, USA

---

## Abstract

We consider the problem of sampling from the ferromagnetic Potts and random-cluster models on a general family of random graphs via the Glauber dynamics for the random-cluster model. The random-cluster model is parametrized by an edge probability  $p \in (0, 1)$  and a cluster weight  $q > 0$ . We establish that for every  $q \geq 1$ , the random-cluster Glauber dynamics mixes in optimal  $\Theta(n \log n)$  steps on  $n$ -vertex random graphs having a prescribed degree sequence with bounded average branching  $\gamma$  throughout the full high-temperature uniqueness regime  $p < p_u(q, \gamma)$ .

The family of random graph models we consider includes the Erdős–Rényi random graph  $G(n, \gamma/n)$ , and so we provide the first polynomial-time sampling algorithm for the ferromagnetic Potts model on Erdős–Rényi random graphs for the full tree uniqueness regime. We accompany our results with mixing time lower bounds (exponential in the largest degree) for the Potts Glauber dynamics, in the same settings where our  $\Theta(n \log n)$  bounds for the random-cluster Glauber dynamics apply. This reveals a novel and significant computational advantage of random-cluster based algorithms for sampling from the Potts model at high temperatures.

**2012 ACM Subject Classification** Theory of computation → Random walks and Markov chains; Theory of computation → Generating random combinatorial structures; Theory of computation → Random network models; Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Markov processes

**Keywords and phrases** Potts model, random-cluster model, random graphs, Markov chains, mixing time, tree uniqueness

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.24

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.10246>

**Funding** *Reza Gheissari*: Thanks the Miller Institute for Basic Research for its support.

**Acknowledgements** The authors thank the anonymous referees for their helpful comments.

## 1 Introduction

The ferromagnetic Potts model is a classical spin system model in statistical physics and computer science. It is defined on a finite graph  $G = (V, E)$ , by a set of spins (or colors)  $[q] = \{1, \dots, q\}$  and an edge weight or inverse temperature parameter  $\beta > 0$ . A configuration  $\sigma \in \{1, \dots, q\}^V$  of the model is an assignment of spins to the vertices of  $V$ . The probability of  $\sigma$  is given by the Gibbs distribution:

$$\mu_{G, \beta, q}(\sigma) = \frac{1}{Z_{G, \beta, q}} \exp(-\beta D(\sigma)), \quad (1)$$

where  $D(\sigma) = |\{\{v, w\} \in E : \sigma(v) \neq \sigma(w)\}|$  is the number of edges whose endpoints have different spins in  $\sigma$ , and  $Z_{G, \beta, q}$  is a normalizing factor known as the partition function. The Ising model of ferromagnetism corresponds to the case where  $q = 2$ .



© Antonio Blanca and Reza Gheissari;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 24; pp. 24:1–24:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sampling from the Potts Gibbs distribution (1) is one of the most frequently encountered problems when running simulations in statistical physics or when solving a variety of inference tasks in computer science; see [33, 32, 54, 52, 25, 24, 49] and the references therein for a sample of these applications. There is a family of powerful sampling algorithms for the Potts model based on its *random-cluster representation*, defined subsequently. Such algorithms, which include the Glauber dynamics of the random-cluster model and the widely-used Swendsen–Wang dynamics [55], are an attractive option computationally since they are often efficient at “low-temperatures” (large  $\beta$ ), where standard Markov chains for the Potts model (including the canonical Glauber dynamics) often converge exponentially slowly; see, e.g., [13, 14, 18, 11, 38].

To be more precise, the *random-cluster model* on a finite graph  $G = (V, E)$ , is defined by an edge probability parameter  $p \in (0, 1)$  and a cluster weight  $q > 0$ . The set of configurations of the model is the set of all subsets of edges  $\omega \subseteq E$ . The probability of each configuration  $\omega$  is given by the Gibbs distribution:

$$\pi_{G,p,q}(\omega) = \frac{1}{Z_{G,p,q}} p^{|\omega|} (1-p)^{|E|-|\omega|} q^{c(\omega)}, \quad (2)$$

where  $c(\omega)$  is the number of connected components (also called clusters) in the subgraph  $(V, \omega)$ , and  $Z_{G,p,q}$  is the corresponding partition function. The random-cluster model was introduced by Fortuin and Kasteleyn [26] as a unifying framework for studying random graphs, spin systems, and electrical networks, and it is also known as the *FK-representation* of the Ising and Potts model.

For integer  $q \geq 2$ , a sample  $\omega \subseteq E$  from the random-cluster Gibbs distribution  $\pi_{G,p,q}$  can be easily transformed into one for the ferromagnetic  $q$ -state Potts model with inverse temperature  $\beta(p) = -\ln(1-p)$ , by independently assigning a random spin from  $\{1, \dots, q\}$  to (all vertices in) each connected component of  $(V, \omega)$  [26, 23, 37]. As such, any sampling algorithm for the random-cluster model yields one for the ferromagnetic Potts model with essentially no computational overhead. This has led to significantly improved sampling algorithms for the Potts model in various low-temperature settings [55, 29, 47, 12, 57, 42] and more generally, to a broad interest in dynamics for the random-cluster model [15, 38, 7, 6, 3, 8, 5].

In this paper, we focus on the Glauber dynamics of the random-cluster model, which for easy distinction we will henceforth call the *FK-dynamics*. From a configuration  $\omega_t \subseteq E$ , one step of this Markov chain transitions to a new configuration  $\omega_{t+1} \subseteq E$  as follows:

1. Choose an edge  $e_t \in E$  uniformly at random;
2. Set  $\omega_{t+1} = \omega_t \cup \{e_t\}$  with probability  $\begin{cases} \hat{p} := \frac{p}{q(1-p)+p} & \text{if } e_t \text{ is a “cut-edge” in } (V, \omega_t); \\ p & \text{otherwise;} \end{cases}$
3. Otherwise set  $\omega_{t+1} = \omega_t \setminus \{e_t\}$ .

Here, we say  $e$  is a *cut-edge* in  $(V, \omega_t)$  if changing the state of  $e_t$  changes the number of connected components  $c(\omega_t)$  in  $(V, \omega_t)$ . The probabilities in step (2) are exactly the conditional probabilities of  $e_t$  being in the configuration  $\omega_t$  given the remainder of  $\omega_t$ . As such, this Markov chain is reversible with respect to  $\pi_{G,p,q}$  and converges to it. We are interested in its *mixing time*  $t_{\text{MIX}}$ ; i.e., the number of steps until the dynamics is within total variation distance  $1/4$  of  $\pi_{G,p,q}$ , starting from the worst possible initial configuration.

As mentioned, the FK-dynamics is by now well-studied in its own right, though sharp analyses of its mixing time are only available on certain structured graphs like the complete graph [6, 36, 8], boxes in the infinite integer lattice graph  $\mathbb{Z}^d$  [7, 5, 34, 35, 40, 31, 14], and

trees [1]. Recently, in [3], the authors studied the FK-dynamics on random regular graphs and established an optimal  $\Theta(n \log n)$  mixing time bound throughout the entire high-temperature tree uniqueness regime.

Our aim in this paper is to study the FK-dynamics in settings in which the maximum degree of the underlying graph is much larger than its *average* degree. Such settings introduce hard technical challenges and have been well-studied for other models; see [20] for some early work. For instance, high-degree vertices are an obstruction to the fast convergence of the Ising/Potts Glauber dynamics. We later prove (see Section 1.2) that on a general class of random graphs on  $n$  vertices with maximum degree  $d_{\max}$ , the Ising/Potts Glauber dynamics requires  $n \cdot \exp(\Omega(d_{\max}))$  steps to converge at high temperatures.

We reveal here that, for the same general family of random graphs, random-cluster based algorithms are not affected by the presence of high-degree vertices; both their mixing times and fast mixing parameter regimes are determined instead by the *average degree* of the graph. This reveals a novel and significant computational advantage of random-cluster based algorithms for sampling from the ferromagnetic Potts model *at high temperatures*. Indeed, prior to this work, random-cluster based sampling algorithms were only found to be more efficient than Ising/Potts Glauber dynamics at low temperatures.

More precisely, we study the mixing time of the FK-dynamics on random graphs of average branching  $\gamma > 0$  in the full uniqueness (high-temperature) regime  $p < p_u(q, \gamma)$ . At integer  $\gamma$ , the threshold  $p_u(q, \gamma)$ , formally defined in (4), was identified in [39] as a uniqueness/non-uniqueness phase transition point of the random-cluster model on the *wired*  $\gamma$ -ary tree, i.e., where the leaves are externally wired to be in the same connected component. For us,  $p_u(q, \gamma)$  is the natural extension of that function to non-integer  $\gamma$ , which we prove corresponds to the high-temperature uniqueness threshold of the random-cluster model on general trees of average branching  $\gamma$  for all  $q \geq 1$ . (This result could be of independent interest.)

Before we describe our general results for random graph models with fixed degree sequence (which we define in the next subsection) we present a special case of our main result of particular interest concerning the FK-dynamics on sparse Erdős–Rényi random graphs.

► **Theorem 1.** *Fix  $q \geq 1$ ,  $\gamma > 0$  and  $p < p_u(q, \gamma)$ . If  $\mathcal{G}$  is an Erdős–Rényi random graph  $\mathcal{G} \sim G(n, \gamma/n)$ , then with probability  $1 - o(1)$ ,  $\mathcal{G}$  is such that the FK-dynamics on  $\mathcal{G}$  satisfies  $t_{\text{MIX}} = \Theta(n \log n)$ .*

This yields a sampler for the Potts distribution on Erdős–Rényi random graphs with near-optimal running time. Let  $\beta_u(q, \gamma) = -\ln(1 - p_u(q, \gamma))$  be the corresponding uniqueness point for the Potts model.

► **Corollary 2.** *Fix  $q \geq 2$ ,  $\gamma > 0$  and  $\beta < \beta_u(q, \gamma)$ . There is a sampling algorithm that, with probability  $1 - o(1)$  over the choice of an Erdős–Rényi random graph  $\mathcal{G} \sim G(n, \gamma/n)$ , outputs a configuration whose distribution is within total-variation distance  $\delta > 0$  of  $\mu_{\mathcal{G}, \beta, q}$  in time  $O(n(\log n)^3 \log(1/\delta))$ .*

Corollary 2 is a direct consequence of Theorem 1 and the aforementioned connection between the random-cluster model and the ferromagnetic Potts model. The extra  $O((\log n)^2)$  factor in the running time of the algorithm comes from the (amortized) cost of checking whether the chosen edge is a cut-edge in each step of the FK-dynamics (see [43, 56]).

To the best of our knowledge, this is the first polynomial-time sampling algorithm for the Potts model on Erdős–Rényi random graphs for  $q \geq 3$  and  $\beta = \Omega(1)$ . Even for the better understood  $q = 2$  case (i.e., the Ising model), Corollary 2 provides the fastest known



sampling algorithm for this parameter regime, improving upon the running time of samplers based on the Glauber dynamics for the Ising model which converges in  $n^{1+\Theta(\frac{1}{\log \log n})}$  steps for all  $\beta < \beta_u(2, \gamma)$  [51].

We mention that the thresholds  $p_u(q, \gamma)$  and  $\beta_u(q, \gamma)$  should be sharp, in the sense that the FK-dynamics is conjectured to undergo polynomial or exponential slowdowns (depending on  $q$ ) at the point  $p_u(q, \gamma)$  (and when  $q > 2$  in a whole critical window  $(p_u, p'_u)$ ). This is by analogy with the FK-dynamics on the complete graph [36] and on random regular graphs [17]; see also [30, 42, 19].

### 1.1 Results on random graphs with general degree sequences

We next provide our main results on random graph models with a fixed degree sequence. Let  $\mathbf{d}_n = (d_1, \dots, d_n)$  be the degree sequence giving the degree of each vertex  $v \in \{1, \dots, n\}$ . Our results will hold for uniform random graphs with degree sequence  $\mathbf{d}_n$  under certain mild conditions on this degree sequence. The first condition we make on  $\mathbf{d}_n$  is that the sequence is *graphical*: i.e., that there exists at least one simple graph having degree sequence  $\mathbf{d}_n$ .

Given a graphical sequence  $\mathbf{d}_n$ , we define  $\mathbb{P}_{\text{RG}(\mathbf{d}_n)}$  as the uniform distribution over all simple graphs on  $n$  vertices having degree sequence  $\mathbf{d}_n$ . The governing quantity in this degree sequence, in terms of the uniqueness thresholds for the Potts and random-cluster models on  $\mathcal{G} \sim \mathbb{P}_{\text{RG}(\mathbf{d}_n)}$ , will be what we call the *effective offspring distribution*  $\mathbb{P}_{\mathbf{d}_n}$ . In words, the distribution  $\mathbb{P}_{\mathbf{d}_n}$  will correspond to choosing  $d - 1$  with probability proportional to the total degree of vertices having degree  $d$ . This determines the offspring distribution corresponding to the random trees one obtains when looking at balls of small radius around a vertex of a random graph  $\mathcal{G} \sim \mathbb{P}_{\text{RG}(\mathbf{d}_n)}$ . Specifically, a vertex of degree  $d$  is selected to be the next vertex added to the random tree with probability proportional to the total degree of all such vertices, and once it is selected and connected to its parent, it has  $d - 1$  available edges to connect to other randomly chosen vertices. Formally,  $\mathbb{P}_{\mathbf{d}_n}$  is defined as the distribution over the set  $\mathcal{M}(\mathbf{d}_n) = \{d_v - 1 : v \in \{1, \dots, n\}\}$  where  $x \in \mathcal{M}(\mathbf{d}_n)$  is assigned probability:

$$\mathbb{P}_{\mathbf{d}_n}(x) = \frac{\sum_v (x+1) \mathbf{1}_{\{d_v=x+1\}}}{\sum_v d_v}. \quad (3)$$

Our results apply to graphical degree sequences where  $\mathbb{P}_{\mathbf{d}_n}$  has bounded finite moments, as we detail next.

► **Definition 3.** Let  $\mathcal{D}_{\gamma, \kappa}$  be the set of graphical degree sequences  $(\mathbf{d}_n)_n$  such that  $D \sim \mathbb{P}_{\mathbf{d}_n}$  has mean that is uniformly bounded away from  $\gamma$  and uniformly bounded  $\kappa$ -th moment. Formally,

1. There exists  $\varepsilon > 0$  such that  $\mathbb{E}_{\mathbf{d}_n}[D] < \gamma - \varepsilon$  for all sufficiently large  $n$ ; and
2. There exists finite  $K > 0$  such that  $\mathbb{E}_{\mathbf{d}_n}[D^\kappa] < K$  for all sufficiently large  $n$ .

This framework is fairly standard in the random graphs literature [10] and is similar to e.g., the setting of [28] for studying sampling from Potts on random graphs with fixed degree sequences at sufficiently low temperatures. While Definition 3 yields a fairly general family of random graphs, we draw attention to some well-studied examples which fall under its umbrella.

► **Example 4.**  *$\Delta$ -regular random graph.* In this case,  $\mathbf{d}_n = (\Delta, \dots, \Delta)$  and the effective offspring distribution simply assigns probability 1 to  $\Delta - 1$ ; thus  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$  for every  $\gamma > \Delta - 1$  and every  $\kappa$ .

► **Example 5.** *Erdős–Rényi random graph  $G(n, \lambda/n)$ .* It was shown in [45] that if  $\mathbf{d}_n$  is drawn as an i.i.d. sequence of Poisson random variables of mean  $\lambda$ , then  $\mathbb{P}_{\text{RG}(\mathbf{d}_n)}$  is contiguous with respect to  $G(n, \lambda/n)$ . (Two random graph models are *contiguous* when any sequence of



events that has a probability of  $1 - o(1)$  in one has a probability of  $1 - o(1)$  in the other model as well.) Hence, it suffices to prove the desired results with high probability over such  $\mathbf{d}_n$ . Standard concentration estimates for Poisson random variables then give that for every  $\gamma > \lambda$  and every  $\kappa$ , with high probability,  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ .

Our main result is an optimal mixing time bound for the FK-dynamics on  $\mathcal{G} \sim \mathbb{P}_{\text{RG}(\mathbf{d}_n)}$ , which applies to all the examples above and more generally to random graphs with degree sequences in  $\mathcal{D}_{\gamma, \kappa}$ .

► **Theorem 6.** *Fix  $q \geq 1$ ,  $\gamma > 0$ , and  $p < p_u(q, \gamma)$ . There exists  $\kappa$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , then with probability  $1 - o(1)$ ,  $\mathcal{G} \sim \mathbb{P}_{\text{RG}(\mathbf{d}_n)}$  is such that the FK-dynamics on  $\mathcal{G}$  satisfies  $t_{\text{MIX}} = \Theta(n \log n)$ .*

This parameter regime in Theorem 6 is tight as FK-dynamics have been very recently shown [17] to exponentially slow down as soon as  $p > p_u(q, \gamma)$  for random regular graphs (Example 4) at integer  $q > 2$ .

The proof of the upper bound in Theorem 6 is the main content of this paper. As mentioned, the special case of the  $\Delta$ -regular random graph (i.e.,  $\mathbf{d}_n = (\Delta, \dots, \Delta)$ ) was the content of an earlier paper [3]. However, as soon as the degree sequence is not homogeneous, substantial further obstacles arise.

First, the uniqueness threshold for the random-cluster model on wired heterogeneous trees (specifically, with offspring distribution  $\mathbb{P}_{\mathbf{d}_n}$ ) had not been established. In our proof of Theorem 6 we in fact require something much stronger; namely, an exponential decay of connectivities with the correct rate (see Lemma 17). The fact that  $p_u(q, \gamma)$  is the uniqueness threshold in the regular case goes back to the work of Häggström [39] (see also [44, 2]). The exponential decay rate was established in [3]. To establish analogous results for the heterogeneous case, we combine the approach of [48] (which considered the special case of the Ising model  $q = 2$ ) with ideas from [2], so as to recurse, not on the marginal of an edge of the tree, but rather on a functional of its probability of downwards connection to infinity.

The second technical obstacle concerns establishing that the FK-dynamics on  $\mathcal{G} \sim \mathbb{P}_{\text{RG}(\mathbf{d}_n)}$  *shatters*, i.e., that its components have size at most  $O(n^\epsilon)$  after  $O(n)$  steps of the dynamics. This is proved using a delicate revealing procedure for the random graph with the FK-dynamics configuration on top of it, a technique introduced in [3] for the case of random regular graphs. The heterogeneity of the degrees in the current setting, however, introduces extra correlations between the underlying graph and the FK-dynamics configuration, necessitating substantial modifications to the revealing procedure from [3].

The changes we make to deal with the above-described dependencies include: (i) modifications to the revealing process so that it is based on half-edges rather than vertices and the dynamics is run in continuous time, and (ii) a new criteria to truncate potentially unbounded increments in the revealing procedure. The more robust procedure yields a notable further improvement: we show that the shattering time is  $O(n)$  (as opposed to  $O(n \log n)$  in [3]). Though this improvement has no impact on the eventual mixing time bound, the more precise understanding of the shattering phase may be useful in other settings. A more detailed proof sketch of this theorem and the new complications that arise is provided in Section 2.

## 1.2 Slowdown for the Ising/Potts Glauber dynamics

Returning to the advantage of FK-dynamics in the presence of high-degree vertices, the following theorem shows that, in the same setting as Theorem 6, the mixing time of the Ising/Potts Glauber dynamics depends exponentially on the maximum degree.

► **Theorem 7.** Fix  $q \geq 2$ ,  $\gamma > 0$  and  $\beta < \beta_u(q, \gamma)$ . Then there exists  $\kappa$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , then with probability  $1 - o(1)$ ,  $\mathcal{G} \sim \mathbb{P}_{\text{RG}}(\mathbf{d}_n)$  is such that the Glauber dynamics for the ferromagnetic Potts model on  $\mathcal{G}$  has  $t_{\text{MIX}} = n \cdot \exp(\Omega(\|\mathbf{d}_n\|_\infty))$ .

Intuitively, the slowdown comes from the fact that the neighborhood of a vertex of degree  $\|\mathbf{d}_n\|_\infty$  is a star graph, where the Ising/Potts Glauber dynamics mixes slowly for  $\beta \gg 1/\|\mathbf{d}_n\|_\infty$ . In random graphs at high temperatures ( $\beta < \beta_u(q, \gamma)$ ) there is no interference with this effect from the rest of the graph. In contrast, the FK-dynamics in the star graph is fast mixing at all temperatures, so this obstruction is not present.

► **Remark 8.** We remark that under various decay of correlation conditions (see, e.g., [21, 41, 22, 16]) the mixing time of the Ising/Potts Glauber dynamics is known to be  $\text{poly}(n)$  when (roughly)  $\beta \leq 1/\|\mathbf{d}_n\|_\infty$ . This does not contradict Theorem 7, which holds when  $\beta = \Omega(1)$ . In fact, if one tracks the dependence on  $\beta$  in our proof, it gives  $t_{\text{MIX}} = n \cdot \exp(\Omega(\beta^2 \|\mathbf{d}_n\|_\infty))$ .

The known  $n^{1+\Omega(\frac{1}{\log \log n})}$  slowdown of the Ising/Potts Glauber dynamics on the Erdős–Rényi random graph [50, 51], where  $\|\mathbf{d}_n\|_\infty = \Theta(\frac{\log n}{\log \log n})$ , is a special case of Theorem 7. Below are a few examples where this slowdown is more dramatic, indeed stretched exponential in the total number of vertices.

► **Example 9.** *Power-law degree distributions.* Consider graphical sequences  $(\mathbf{d}_n)_n$  satisfying item (1) in Definition 3, and for which the fraction of degrees of size  $\ell$  is  $\Theta(\ell^{-\zeta})$ . For every  $\kappa$ , if  $\zeta > \kappa + 2$ , one would have  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ . In such situations,  $\|\mathbf{d}_n\|_\infty = \Theta(n^{1/\zeta})$  and  $t_{\text{MIX}} = \exp(\Omega(n^{1/\zeta}))$ .

► **Example 10.** *Planted high-degree vertices.* Consider a random  $\Delta$ -regular random graph and change the degree of one vertex to  $\Theta(n^\varepsilon)$ . If  $\varepsilon < 1/(\kappa + 1)$  and  $\gamma > \Delta - 1$ , then  $(\mathbf{d}_n) \in \mathcal{D}_{\gamma, \kappa}$  and  $t_{\text{MIX}} = \exp(\Omega(n^\varepsilon))$ .

In the above instances where the maximum degree is polynomial in  $n$  but the average degree is constant, there is an exponential vs. polynomial difference in the high-temperature mixing times of the Ising/Potts Glauber dynamics and of the FK-dynamics. At this level, the computational benefits of random-cluster based sampling methods also extend to the often implemented Swendsen–Wang dynamics [55]. In particular, using the comparison inequalities from [57] the upper bounds of Theorems 1 and 6 translate into  $O(n^2 \log n)$  upper bounds on the mixing time of the Swendsen–Wang dynamics in those settings.

## 2 Proof outline

In this section, we present our technical contributions about random graphs (Section 2.1), the exponential decay of correlations and uniqueness on heterogeneous trees (Section 2.2), and the shattering phenomenon of the FK-dynamics (Section 2.3), all of which we combine in Section 2.4 to yield our main result: the mixing time upper bound of Theorem 6. Throughout the paper, a subset  $\omega \subset E$  is naturally identified with an assignment of  $\{0, 1\}$ , or closed and open, to  $E$ , via  $\omega(e) = 1$  if and only if  $e \in \omega$ . All our results should also be understood to hold uniformly over all sufficiently large  $n$ .

### 2.1 Random graphs

We start by describing the locally treelike structure and exponential rate of volume growth of random graphs with fixed degree sequence  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ . It will be convenient to work with the *configuration model*, a useful and standard tool for studying random graphs with

fixed degree sequence. The configuration model  $\mathbb{P}_{\text{CM}(\mathbf{d}_n)}$  is a distribution over multigraphs on  $n$  vertices with degree sequence  $\mathbf{d}_n$ . It is defined by giving  $d_v$  half-edges to every vertex  $v$  and drawing a uniform random perfect matching on the  $\sum_v d_v$  many half-edges to form the  $\frac{1}{2} \sum_v d_v$  edges of the graph [9]. It is a standard fact that for any  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , and any sequence of sets  $A_n$  of simple graphs on  $n$  vertices, we have

$$\mathbb{P}_{\text{RG}(\mathbf{d}_n)}(\mathcal{G} \in A_n) = o(1) \quad \text{if and only if} \quad \mathbb{P}_{\text{CM}(\mathbf{d}_n)}(\mathcal{G} \in A_n) = o(1) :$$

see [9, 27]. It thus suffices to prove Theorems 6-7 for  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$ .

For a graph  $G = (V, E)$  and a vertex  $v \in V$ , we define the ball of radius  $R$  around  $v$  as:

$$B_R(v) := \{w \in V : d(w, v) \leq R\},$$

where  $d(\cdot, \cdot)$  is the graph distance. For a set  $B \subset V$  define  $E(B) = \{\{v, w\} \in E : v, w \in B\}$ .

► **Definition 11.** We say that a graph  $G = (V, E)$  is  $L$ -Treelike if there is a set  $H \subset E$  with  $|H| \leq L$  such that the graph  $(V, E \setminus H)$  is a tree. We say that  $G$  is  $(L, R)$ -Treelike if for every  $v \in V$  the subgraph  $(B_R(v), E(B_R(v)))$  is  $L$ -Treelike.

The following lemma says that small balls of the random graph  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$  are tree-like. Indeed,  $B_R(v)$  in  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$  is typically a random tree with offspring distribution approximately  $\mathbb{P}_{\mathbf{d}_n}$ , defined in (3).

► **Lemma 12.** There exists  $\kappa$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$  the following holds. For every  $\delta > 0$ , there exists  $L = L(\delta)$  such that if  $1 \leq R \leq (\frac{1}{2} - \delta) \log_\gamma n$ , we have  $\mathbb{P}_{\text{CM}(\mathbf{d}_n)}(\mathcal{G} \text{ is } (L, R)\text{-Treelike}) = 1 - o(n^{-10})$ .

Using standard concentration estimates for the volume of Galton–Watson trees, we also establish that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , then  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$  has average exponential rate  $\gamma$  of volume growth.

► **Definition 13.** A graph  $G = (V, E)$  on  $n$  vertices is said to have  $(\gamma, \varepsilon)$ -volume growth if for every  $v \in V$  and every integer  $r \in [\varepsilon \log_\gamma n, \frac{1}{2} \log_\gamma n]$  the graph has  $|B_r(v)| \leq \gamma^r$ .

► **Lemma 14.** Fix  $\varepsilon \in (0, \frac{1}{2})$ . There exists  $\kappa(\varepsilon)$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , then

$$\mathbb{P}_{\text{CM}(\mathbf{d}_n)}(\mathcal{G} \text{ has } (\gamma, \varepsilon)\text{-growth}) \geq 1 - o(n^{-10}).$$

## 2.2 Exponential decay and uniqueness on general trees and treelike graphs

Given the local tree structure of the random graphs from  $\mathbb{P}_{\text{CM}(\mathbf{d}_n)}$ , to control the decay rate of connectivities of the random-cluster model on  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$ , we need to first understand how these connectivities decay on heterogeneous (i.e., non-regular) trees. The relevant random-cluster measure on the tree requires the addition of *boundary conditions* mimicking the possible presence of open edges in the random graph outside of the treelike ball. Towards this, let us formally define boundary conditions.

► **Definition 15.** A random-cluster boundary condition  $\xi$  on  $G = (V, E)$  is a partition of  $V$ , such that the vertices in each element of the partition are identified with one another. The random-cluster measure with boundary conditions  $\xi$ , denoted  $\pi_{G, p, q}^\xi$ , is the same as in (2) except the number of connected components  $c(\omega) = c(\omega; \xi)$  would be counted with this vertex identification, i.e., if  $v, w$  are in the same element of  $\xi$ , they are always counted as being in the same connected component of  $\omega$  in (2). The boundary condition can alternatively be seen as external “wirings” of the vertices in the same element of  $\xi$ .

► **Remark 16.** The *free* boundary condition,  $\xi = 0$ , corresponds to the case of no external wirings; i.e., its partition is the one consisting of only of singletons. For a subset  $\partial V \subset V$ , the *wired* boundary condition on  $\partial V$ , denoted  $\xi = 1$ , is the one whose partition has all vertices of  $\partial V$  in the same element (and all vertices of  $V \setminus \partial V$  as singletons); i.e.,  $\xi = \{\partial V\} \cup \bigcup\{v : v \in V \setminus \partial V\}$ . For boundary conditions  $\xi, \xi'$  we say  $\xi \leq \xi'$  if  $\xi$  is a finer partition than  $\xi'$ . When  $q \geq 1$ , the random-cluster model has the following monotonicity property: for any two boundary conditions  $\xi \geq \xi'$ ,  $\pi_{G,p,q}^\xi \succcurlyeq \pi_{G,p,q}^{\xi'}$  where  $\succcurlyeq$  denotes stochastic domination [37].

Now define the threshold

$$p_u(q, \gamma) := 1 - \frac{1}{1 + \inf_{y>1} h(y)}, \quad \text{where} \quad h(y) := \frac{(y-1)(y^\gamma + q - 1)}{y^\gamma - y}. \quad (4)$$

The work [39] studied the random-cluster measure on homogeneous,  $d$ -ary trees, with wired boundary conditions and identified  $p_u(q, d)$  as the *uniqueness threshold* such that whenever  $p < p_u(q, d)$ , the probability that the root is connected to a distance  $h$  in the wired  $d$ -ary tree goes to zero as  $h \rightarrow \infty$ ; a different proof was given in [2]. In [3], it was shown that this decay is in fact exponential with rate  $\hat{p} = p/(p + q(1 - p))$ . However, the methods of those papers do not easily extend to the non-regular setting, where there may be vertices of unbounded degree, but one would expect the threshold for connectivity decay to only depend on the *average* branching rate. In [48], it was shown that the analogue  $\beta_u(2, \gamma)$  of (4) gives the correct uniqueness threshold in the case of the Ising model  $q = 2$ , for general (non-homogenous) trees of average branching  $\gamma$ . However, the argument there recursed over the single-site spin marginals, and relied on the fact that it was an Ising model whose interactions are nearest-neighbor. In the case of the random-cluster model, interactions between edge-marginals are non-local, and we therefore have to work with a more complicated functional encoding the probability of an edge being downward connected to the wired boundary. Combining ideas from [48] and [2], we are then able to establish uniqueness, and that connectivities decay exponentially with rate  $\hat{p}$  on general heterogenous trees of average branching factor  $\gamma$  for all  $q \geq 1$  and all  $p < p_u(q, \gamma)$ . When  $p < p_u(q, \gamma)$ , we have  $\hat{p} < 1/\gamma$  (see e.g., [39, Theorem 1.5]); this indicates by a union bound why there will typically be no connections to the boundary in a tree of average branching  $\gamma$ .

More formally, let  $\mathcal{T}_h = (V(\mathcal{T}_h), E(\mathcal{T}_h))$  be an arbitrary finite tree, rooted at  $\rho$ , and of height  $h$ . Let  $\partial\mathcal{T}_h \subset V(\mathcal{T}_h)$  be the set of vertices of  $\mathcal{T}_h$  at distance exactly  $h$  from  $\rho$ . For  $v \in V(\mathcal{T}_h)$ , let  $\mathcal{T}_v$  be the subtree of  $\mathcal{T}_h$  rooted at  $v$ , let  $h(v)$  denotes the height of  $\mathcal{T}_v$ , and let  $\partial\mathcal{T}_v = \partial\mathcal{T}_h \cap \mathcal{T}_v$ . For a random-cluster configuration  $\omega$  on  $\mathcal{T}_h$ , let  $\mathcal{C}_\rho(\omega)$  denote the connected component of  $\omega$  that contains the root  $\rho$  of  $\mathcal{T}_h$ . Finally, let  $(1, \odot)$  denote the boundary condition that wires all vertices of  $\partial\mathcal{T}_h$  together, and also wires them up to the root, and let  $\pi_{\mathcal{T}_h}^{(1, \odot)}$  be the random-cluster measure with this boundary condition.

► **Lemma 17.** Fix  $q \geq 1$ ,  $\gamma > 1$ ,  $p < p_u(q, \gamma)$ , and  $\varepsilon \in [0, 1)$ . Suppose that  $|\partial\mathcal{T}_v| \leq \gamma^{h(v)}$  for every  $v \in V(\mathcal{T}_h)$  with  $h(v) > \varepsilon h$ . Then, there exists a constant  $C = C(p, q, \gamma)$  such that for any  $u \in \partial\mathcal{T}_h$

$$\pi_{\mathcal{T}_h}^{(1, \odot)}(\omega : u \in \mathcal{C}_\rho(\omega)) \leq C \hat{p}^{(1-\varepsilon)h}.$$

We note that the condition that  $|\partial\mathcal{T}_v| \leq \gamma^{h(v)}$  for every  $v \in V(\mathcal{T}_h)$  with  $h(v) > \varepsilon h$  in the lemma holds with high probability for random trees with averaging branching  $\gamma$ . In addition, the exponential decay rate in Lemma 17 is essentially optimal, and together with Lemmas 12–14, allows us to derive precise estimates on the exponential decay of connectivities on the

treelike balls around each vertex of the random graph  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$ . We will actually need a sharp bound on the rate of influence decay between the boundary and the center of the ball  $B_R(v)$ ; we find that this is the square of the rate of connectivity decay on a corresponding tree of depth  $R$ . (Intuitively, this is because *two* disjoint paths are required to reach the center of the ball in order for the boundary to have any effect on it.) To be more precise, let  $G = (V, E)$  be a graph and for  $v \in V$ , let  $E_v \subseteq E$  denote the set of edges incident to  $v$ .

► **Definition 18.** A random-cluster boundary condition  $\xi$  on a graph  $H$  is said to be  $K$ -Sparse if the number of vertices in non-trivial (non-singleton) boundary components of  $\xi$  is at most  $K$ .

► **Theorem 19.** Fix  $\gamma > 0$ ,  $q \geq 1$ , and  $p < p_u(q, \gamma)$ . Suppose  $G$  is  $(L, R)$ -Treelike for some  $L$  and some  $R \leq \frac{1}{2} \log_\gamma n$ . Also suppose  $G$  has  $(\gamma, \varepsilon)$ -volume growth for some  $\varepsilon > 0$  sufficiently small. There exists a constant  $C > 0$  such that for every  $v \in G$ , and any two  $K$ -Sparse boundary conditions  $\xi$  and  $\tau$  on  $B_R(v)$ :

$$\|\pi_{B_R(v)}^\xi(\omega(E_v) \in \cdot) - \pi_{B_R(v)}^\tau(\omega(E_v) \in \cdot)\|_{\text{TV}} \leq C \hat{p}^{(2-CL\sqrt{\varepsilon})R}.$$

### 2.3 Shattering of the FK-dynamics

With Theorem 19 in hand, the core of our argument becomes establishing that the boundary conditions induced by the FK-dynamics chains on the small balls around each vertex are *shattered*. This is formalized as follows.

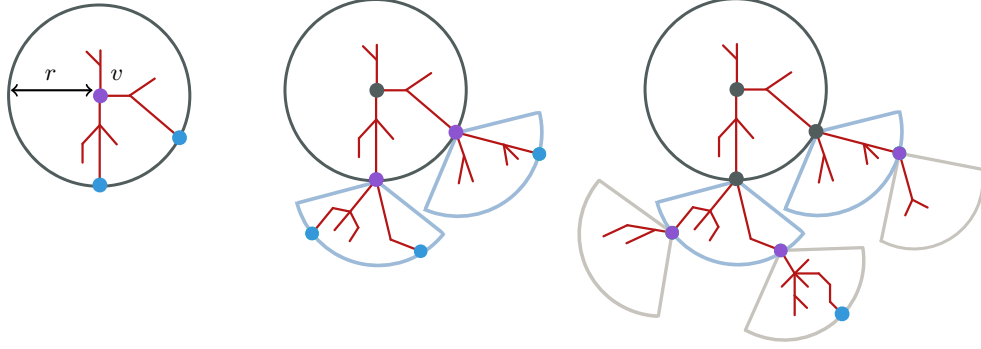
► **Definition 20.** A random-cluster configuration  $\omega$  on  $G = (V(G), E(G))$  is  $(K, R)$ -Sparse if, for every  $v \in V(G)$ , the boundary conditions induced on  $B_R(v)$  by  $\omega(E(G) \setminus E(B_R(v)))$  are  $K$ -Sparse.

► **Remark 21.** It will be technically convenient to prove our results in continuous time instead of discrete time. In the continuous-time FK-dynamics, each edge of the graph has a rate-1 Poisson clock and every time a clock rings, the corresponding edge is updated as in the discrete-time version of the FK-dynamics; that is, according to the conditional distribution given the configuration off of this edge. It is a standard fact (see e.g., [46, Theorem 20.3]) that the discrete-time mixing time is comparable to  $|E(\mathcal{G})|$  times the continuous-time mixing time. It therefore suffices for us to establish the mixing time bounds of Theorems 1 and 6 as  $\Theta(\log n)$  bounds for the continuous-time version of the FK-dynamics. From this point on, we let  $X_t^{x_0}$  denote the continuous-time FK-dynamics on  $\mathcal{G}$  initialized from the configuration  $x_0$ , and use the superscripts 1 and 0 to denote the full (all-open) and empty (all-closed) configurations, respectively.

The following theorem says that after  $O(1)$  continuous time, the configuration of the FK-dynamics from the all wired, and by monotonicity from any, initialization is  $(K, R)$ -Sparse, for  $R \leq \frac{1}{2} \log_\gamma n$  and constant  $K$ .

► **Theorem 22.** Fix  $q \geq 1$ ,  $\gamma > 0$  and  $p < p_u(q, \gamma)$ . For every  $\delta > 0$ , there exists  $\kappa$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$ , there exists  $T = T(p, q, \gamma)$  and  $K = K(p, q, \gamma, \delta)$  such that for any  $t \geq T$ , and every  $R \leq (\frac{1}{2} - \delta) \log_\gamma n$ , with probability  $1 - o(1)$ ,  $\mathcal{G} \sim \mathbb{P}_{\text{CM}(\mathbf{d}_n)}$  is such that  $\mathbb{P}(X_t^1 \text{ is } (K, R)\text{-Sparse}) \geq 1 - o(n^{-5})$ .

Our proof of Theorem 22 relies on a delicate simultaneous revealing procedure for the random graph, along with the connected component of a vertex  $v$  in  $X_t^1$ , showing that after a burn-in period, the configuration  $X_t^1$  is shattered. This technique was introduced for random regular graphs in [3]. The revealing scheme for the component of a vertex  $v$  in the FK-dynamics chain  $X_t^1$  roughly proceeds as follows (see Figure 1). First set the starting vertex  $v$  as “exposed”, and iteratively, for each exposed vertex  $u$  do the following:



■ **Figure 1** Three “generations” of the revealing procedure. In each figure, the purple vertices are the current generation of exposed vertices; the revealing procedure reveals the ball of radius  $r$  around such a vertex  $v$ , and a dominating localized FK-dynamics configuration  $\tilde{\omega}(B_r(v))$  on that ball. The next generation of exposed vertices (blue) consists of those on the boundary  $B_r(v)$  that are in the connected component of  $v$  in the configuration  $\tilde{\omega}(B_r(v))$ . Exposed vertices from previous generations are then colored black.

1. Reveal the ball  $B_r(u)$  in the random graph for a large  $r = O(1)$ ;
2. Reveal a configuration  $\tilde{\omega}(B_r(u))$  that dominates the configuration of the FK-dynamics at time  $t$  on  $B_r(u)$ . This configuration will come from simulating the FK-dynamics that ignores all updates outside of  $B_r(u)$  (effectively inducing the wired boundary condition on  $B_r(u)$ ) and thus can be obtained independently of the dynamics on the rest of the graph;
3. Add to the set of exposed vertices all vertices of  $\partial B_r(u)$  that get connected to  $u$  in  $\tilde{\omega}(B_r(u))$ .

The key point of the argument is then to stochastically dominate the exposed vertices by a branching process, which can be shown to be sub-critical (see Lemma 17). In our setting, the heterogeneity of the degrees causes substantial complications to the argument from [3], because in balls where the branching rate is locally larger than  $\gamma$ , the overlaid FK-dynamics configuration will actually be highly connected. The presence of high degrees also destroys the  $O(1)$  bounds on the maximum number of new vertices that could possibly get exposed in step (3) above; this complicates relevant concentration arguments, as our branching process martingale will no longer have bounded increments.

## 2.4 Proof of main result

We end this section by proving the mixing time upper bound of Theorem 6. The proofs of all other results are deferred to the full version of this paper [4]. It suffices to prove the result for  $\gamma > 1$  since  $\lim_{\gamma \downarrow 1} p_u(q, \gamma) = 1$ , and if  $\gamma \geq \gamma'$ , then  $\mathcal{D}_{\gamma', \kappa} \subset \mathcal{D}_{\gamma, \kappa}$ .

**Proof of Theorem 6: upper bound.** Fix  $q > 1$ ,  $\gamma > 1$  and  $p < p_u(q, \gamma)$ . Let  $R = (\frac{1}{2} - \delta) \log_\gamma n$ , where  $\delta > 0$  is a small constant we choose later. For  $K$  and  $L$  fixed positive constants,  $\varepsilon \in (0, 1/2)$  and  $t \geq 0$ , let  $\Gamma_t = \Gamma_t(L, K, \delta, \varepsilon, \gamma)$  be the subset of (multi)graphs on  $n$  vertices with degree sequence  $\mathbf{d}_n$  given by:

$$\Gamma_t = \{ \mathcal{G} : \mathcal{G} \text{ is } (L, R)\text{-Treelike, has } (\gamma, \varepsilon)\text{-volume growth,} \\ \text{and } \mathbb{P}(X_{\mathcal{G}, t}^1 \text{ is } (K, R)\text{-Sparse}) \geq 1 - n^{-5} \}.$$



By Lemmas 12 and 14, as well as Theorem 22, for every  $\delta \in (0, 1/2)$  and  $\varepsilon \in (0, 1/2)$ , there exist constants  $\kappa(p, q, \gamma, \delta)$ ,  $L(\delta)$ ,  $K(p, q, \gamma, \delta)$ , and  $T(p, q, \gamma)$  such that if  $(\mathbf{d}_n)_n \in \mathcal{D}_{\gamma, \kappa}$  then  $\mathbb{P}_{\text{cm}(\mathbf{d}_n)}(\Gamma_T^c) = o(1)$ . Hence, it suffices for us to prove that the mixing time of the FK-dynamics on any  $\mathcal{G} \in \Gamma_T$  is  $O(\log n)$ .

Fix any  $\mathcal{G} \in \Gamma_T$ . Let  $((X_t^{x_0})_{t \geq 0})_{x_0}$  be the family of FK-dynamics initialized from all possible configurations  $x_0$ , coupled via the standard grand coupling for the FK-dynamics; i.e., using the same clock rings and the same uniform random variables to make the edge updates while running the chain from different initializations. Recall that this coupling is monotone when  $q \geq 1$  so that for every  $t \geq 0$ , if  $X_t^{x_0} \leq X_t^{y_0}$ , then  $X_{t'}^{x_0} \leq X_{t'}^{y_0}$  for all  $t' \geq t$ . Using the standard fact that the coupling time provides a bound on the mixing time (see e.g., [46]), by a union bound over the edges, it suffices to show that under this grand coupling,

$$\mathbb{P}(X_T^1(e) \neq X_T^0(e)) = o(1/|E(\mathcal{G})|) \quad \text{for every } e \in E(\mathcal{G}). \quad (5)$$

Now fix any such  $e = \{u, v\}$  and for ease of notation, set  $B_v = E(B_R(v))$  and  $B_v^c = E(\mathcal{G}) \setminus B_v$ . Consider two auxiliary copies of the FK-dynamics  $Y_t^1$  and  $Y_t^0$  that censor (ignore) all updates on edges of  $B_v^c$  after time  $T$ . The censoring inequality from [53] applied to the FK-dynamics [35, Theorem 2.5] implies that  $Y_t^1 \succcurlyeq X_t^1$  and  $Y_t^0 \preccurlyeq X_t^0$  for all  $t \geq 0$  and thus

$$\mathbb{P}(X_t^1(e) \neq X_t^0(e)) = \mathbb{P}(X_t^1(e) = 1) - \mathbb{P}(X_t^0(e) = 1) \leq \mathbb{P}(Y_t^1(e) = 1) - \mathbb{P}(Y_t^0(e) = 1).$$

Let  $\mathcal{H}_v$  be the set of configurations on  $B_v^c$  such that the boundary conditions they induce on  $B_v$  are  $K$ -Sparse. (Here and throughout the paper, the boundary condition induced by a configuration  $\omega(B^c)$  on a set  $B$  wires two vertices  $w, w' \in V(B)$  if they are in the same connected component of  $\omega(B^c)$ .) By definition of  $\Gamma_T$  and monotonicity, we have for every  $\mathcal{G} \in \Gamma_T$ ,  $\mathbb{P}(Y_T^0(B_v^c) \notin \mathcal{H}_v) \leq \mathbb{P}(Y_T^1(B_v^c) \notin \mathcal{H}_v) \leq n^{-5}$ , because up to time  $T$  there is no censoring, and so  $Y_T^1, Y_T^0$  have the same distribution as  $X_T^1, X_T^0$ , respectively. Therefore,  $\mathbb{P}(Y_t^1(e) = 1) - \mathbb{P}(Y_t^0(e) = 1)$  is bounded by

$$\max_{\phi^1, \phi^0 \in \mathcal{H}_v} \left[ \mathbb{P}(Y_t^1(e) = 1 \mid Y_T^1(B_v^c) = \phi^1) - \mathbb{P}(Y_t^0(e) = 1 \mid Y_T^0(B_v^c) = \phi^0) \right] + 2n^{-5}.$$

Fix any  $\phi^1, \phi^0 \in \mathcal{H}_v$ , and let  $t = T + s$ . By the triangle inequality, the difference in the brackets is at most

$$\left| \mathbb{P}(Y_{T+s}^1(e) = 1 \mid Y_T^1(B_v^c) = \phi^1) - \pi_{\mathcal{G}}(\omega(e) = 1 \mid \omega(B_v^c) = \phi^1) \right| \quad (6)$$

$$+ \left| \pi_{\mathcal{G}}(\omega(e) = 1 \mid \omega(B_v^c) = \phi^1) - \pi_{\mathcal{G}}(\omega(e) = 1 \mid \omega(B_v^c) = \phi^0) \right| \quad (7)$$

$$+ \left| \mathbb{P}(Y_{T+s}^0(e) = 1 \mid Y_T^0(B_v^c) = \phi^0) - \pi_{\mathcal{G}}(\omega(e) = 1 \mid \omega(B_v^c) = \phi^0) \right|. \quad (8)$$

Observe that the chain  $(Y_{T+s}^1)_{s \geq 0}$  may be viewed as an FK-dynamics on  $B_v$  with the boundary condition induced by  $\phi^1$ , initialized from the (random) configuration  $Y_T^1(B_v)$  and with stationary distribution  $\pi_{\mathcal{G}}(\omega(B_v) \in \cdot \mid \omega(B_v^c) = \phi^1) = \pi_{B_v}^{\phi^1}$ ; the analogous statement is true for  $(Y_{T+s}^0)_{s \geq 0}$  and  $\pi_{B_v}^{\phi^0}$ .

In order to bound (6) and (8), we use as an input a bound on the rate of convergence of FK-dynamics on treelike graphs with sparse boundary conditions. This bound comes from a straightforward (Dirichlet form) comparison to the FK-dynamics on a tree with free boundary conditions, and its proof is deferred to the full version [4].

► **Lemma 23.** *Consider an  $L$ -Treelike graph  $G = (V, E)$  with a  $K$ -Sparse boundary condition  $\xi$ . Let  $\pi_{\min} = \min_{x \in \Omega} \pi_G^\xi(x)$ . For every  $p \in (0, 1)$  and  $q > 0$ , there exists  $\alpha_0 = \alpha_0(p, q, L, K) > 0$  such that*

$$\max_{x_0 \in \Omega} \|\mathbb{P}(X_t^{x_0} \in \cdot) - \pi_G^\xi\|_{\text{TV}} \leq e^{-\alpha_0 t} \sqrt{2 \log(1/\pi_{\min})}.$$



Setting  $\hat{T} = T + \hat{S}_n$  where  $\hat{S}_n = \hat{C} \log n$  for a constant  $\hat{C}(p, q, \gamma, L, K)$  sufficiently large, since  $B_v$  is  $L$ -Treelike and  $\phi^1$  is  $K$ -Sparse, by Lemma 23, we obtain the following, and its analogue for (8):

$$|\mathbb{P}(Y_{\hat{T}}^1(e) = 1 \mid Y_{\hat{T}}^1(B_v^c) = \phi^1) - \pi_{\mathcal{G}}(\omega(e) = 1 \mid \omega(B_v^c) = \phi^1)| \leq n^{-5}.$$

Finally, since both  $\phi^1$  and  $\phi^0$  induce  $K$ -Sparse boundary conditions on  $B_v$  and  $\mathcal{G}$  is  $(L, R)$ -Treelike with  $(\gamma, \varepsilon)$ -volume growth, by Theorem 19 there exists  $C = C(p, q, L, K, \gamma) > 0$  such that (7) is at most

$$\|\pi_{B_v}^{\phi^1}(\omega(E_v) \in \cdot) - \pi_{B_v}^{\phi^0}(\omega(E_v) \in \cdot)\|_{\text{TV}} \leq C\hat{p}^{2(1-C\sqrt{\varepsilon})R} \leq C\hat{p}^{(1-2\delta)(1-C\sqrt{\varepsilon})\log_{\gamma} n},$$

where  $E_v$  is the set of edges incident to  $v$ , and we used  $R = (\frac{1}{2} - \delta)\log_{\gamma} n$ . Setting  $\theta = (1 - 2\delta)(1 - C\sqrt{\varepsilon})$ ,

$$\|\pi_{B_v}^{\phi^1}(\omega(E_v) \in \cdot) - \pi_{B_v}^{\phi^0}(\omega(E_v) \in \cdot)\|_{\text{TV}} \leq C\hat{p}^{\theta \log_{\gamma} n} = Cn^{-\theta(1 - \frac{1}{\log_{\hat{p}\gamma} \gamma})}. \quad (9)$$

Since  $\hat{p} < 1/\gamma$ ,  $\log_{\hat{p}\gamma} \gamma < 0$ , there is some  $c_{p,\gamma} > 0$  such that the right-hand side is  $Cn^{-\theta(1+c_{p,\gamma})}$ . By taking  $\varepsilon, \delta$  sufficiently small,  $\theta$  can be made arbitrarily close to 1, so that (9) is  $o(1/n)$ .

Now notice that  $|E(\mathcal{G})| = O(n)$ . To see this, observe that by Jensen's inequality  $(\frac{1}{n} \sum_v d_v)^2 \leq \frac{1}{n} \sum_v d_v^2$ , and since  $(\mathbf{d}_n) \in \mathcal{D}_{\gamma,\kappa}$ , we also have  $\sum_v d_v^2 \leq (1 + \gamma) \sum_v d_v$ . Combining these two inequalities we find that  $|E(\mathcal{G})| \leq \frac{(1+\gamma)n}{2}$ . Therefore, each of (6)–(8) are  $o(1/|E(\mathcal{G})|)$ , implying (5) as desired.  $\blacktriangleleft$

---

## References

- 1 Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda. The Swendsen-Wang dynamics on trees. In *Proceedings of the 25th International Workshop on Randomization and Computation (RANDOM)*, 2021.
- 2 Antonio Blanca, Andreas Galanis, Leslie Ann Goldberg, Daniel Štefankovič, Eric Vigoda, and Kuan Yang. Sampling in uniqueness from the Potts and random-cluster models on random regular graphs. In *Proceedings of the 22nd International Workshop on Randomization and Computation (RANDOM)*, 2018.
- 3 Antonio Blanca and Reza Gheissari. Random-cluster dynamics on random regular graphs in tree uniqueness. *Communications in Mathematical Physics*, 2021. doi:10.1007/s00220-021-04093-z.
- 4 Antonio Blanca and Reza Gheissari. Sampling from Potts on random graphs of unbounded degree via random-cluster dynamics. *arXiv preprint*, 2021. arXiv:2107.10246.
- 5 Antonio Blanca, Reza Gheissari, and Eric Vigoda. Random-cluster dynamics in  $\mathbb{Z}^2$ : Rapid mixing with general boundary conditions. *Ann. Appl. Probab.*, 30(1):418–459, February 2020. doi:10.1214/19-AAP1505.
- 6 Antonio Blanca and Alistair Sinclair. Dynamics for the mean-field random-cluster model. In *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 528–543, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.528.
- 7 Antonio Blanca and Alistair Sinclair. Random-cluster dynamics in  $\mathbb{Z}^2$ . *Probab. Theory Related Fields*, 2016. Extended abstract appeared in Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), pp. 498–513. doi:10.1007/s00440-016-0725-1.
- 8 Antonio Blanca, Alistair Sinclair, and Xusheng Zhang. The critical mean-field Chayes-Machta dynamics. In *Proceedings of the 25th International Workshop on Randomization and Computation (RANDOM)*, 2021.
- 9 Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

- 10 Béla Bollobás. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2001. doi:10.1017/CB09780511814068.
- 11 Magnus Bordewich, Catherine Greenhill, and Viresh Patel. Mixing of the glauher dynamics for the ferromagnetic potts model. *Random Structures & Algorithms*, 48(1):21–52, 2016.
- 12 Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on  $Z^d$  at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 738–751, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384271.
- 13 Christian Borgs, Jennifer T. Chayes, Alan Frieze, Jeong H. Kim, Prasad Tetali, Eric Vigoda, and Van H. Vu. Torpid mixing of some Monte Carlo Markov chain algorithms in statistical physics. In *Proc. of the 40th Annual Symposium on Foundations of Computer Science (FOCS 1999)*, pages 218–229, 1999. doi:10.1109/SFFCS.1999.814594.
- 14 Christian Borgs, Jennifer T. Chayes, and Prasad Tetali. Tight bounds for mixing of the Swendsen-Wang algorithm at the Potts transition point. *Probab. Theory Related Fields*, 152(3-4):509–557, 2012. doi:10.1007/s00440-010-0329-0.
- 15 Lincoln Chayes and Jon Machta. Graphical representations and cluster algorithms I. Discrete spin systems. *Physica A: Statistical Mechanics and its Applications*, 239(4):542–601, 1997.
- 16 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Rapid mixing of Glauber dynamics via spectral independence for all degrees. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 137–148. IEEE, 2022.
- 17 Amin Coja-Oghlan, Andreas Galanis, Leslie Ann Goldberg, Jean Bernoulli Ravelomanana, Daniel Štefankovič, and Eric Vigoda. Metastability of the Potts Ferromagnet on Random Regular Graphs. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.45.
- 18 Paul Cuff, Jian Ding, Oren Louidor, Eyal Lubetzky, Yuval Peres, and Allan Sly. Glauber dynamics for the mean-field Potts model. *Journal of Statistical Physics*, 149(3):432–477, 2012.
- 19 Amir Dembo, Andrea Montanari, Allan Sly, and Nike Sun. The replica symmetric solution for Potts models on  $d$ -regular graphs. *Communications in Mathematical Physics*, 327(2):551–575, 2014. doi:10.1007/s00220-014-1956-6.
- 20 Martin Dyer, Abraham D Flaxman, Alan M Frieze, and Eric Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 29(4):450–465, 2006.
- 21 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Dobrushin conditions and systematic scan. *Combinatorics, Probability and Computing*, 17(6):761–779, 2008.
- 22 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Matrix norms and rapid mixing for spin systems. *The Annals of Applied Probability*, 19(1):71–107, 2009.
- 23 Robert G. Edwards and Alan D. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm. *Phys. Rev. D (3)*, 38(6):2009–2012, 1988. doi:10.1103/PhysRevD.38.2009.
- 24 G. Ellison. Learning, local interaction, and coordination. *Econometrica: Journal of the Econometric Society*, pages 1047–1071, 1993.
- 25 J. Felsenstein. *Inferring phylogenies*, volume 2. Sinauer Associates, Inc., Sunderland, MA, 2004.
- 26 Cornelius M. Fortuin and Pieter W. Kasteleyn. On the random-cluster model. I. Introduction and relation to other models. *Physica*, 57:536–564, 1972.
- 27 Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

- 28 Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast mixing via polymers for random graphs with unbounded degree. *Information and Computation*, page 104894, 2022.
- 29 Andreas Galanis, Daniel Štefanković, and Eric Vigoda. Swendsen-Wang Algorithm on the Mean-Field Potts Model. In *Proc. of the 19th International Workshop on Randomization and Computation (RANDOM 2015)*, pages 815–828, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.815.
- 30 Andreas Galanis, Daniel Štefanković, Eric Vigoda, and Linji Yang. Ferromagnetic Potts model: Refined #BIS-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016.
- 31 Shirshendu Ganguly and Insuk Seo. Information percolation and cutoff for the random-cluster model. *Random Structures & Algorithms*, 57(3):770–822, 2020. doi:10.1002/rsa.20931.
- 32 S. Geman and C. Graffigne. Markov random field image models and their applications to computer vision. In *Proceedings of the International Congress of Mathematicians*, volume 1, pages 1496–1517. Berkeley, CA, 1986.
- 33 H.-O. Georgii. *Gibbs measures and phase transitions*, volume 9. Walter de Gruyter, 2011.
- 34 Reza Gheissari and Eyal Lubetzky. Mixing times of critical two-dimensional Potts models. *Comm. Pure Appl. Math.*, 71(5):994–1046, 2018.
- 35 Reza Gheissari and Eyal Lubetzky. Quasi-polynomial mixing of critical two-dimensional random cluster models. *Random Structures and Algorithms*, 2019. doi:10.1002/rsa.20868.
- 36 Reza Gheissari, Eyal Lubetzky, and Yuval Peres. Exponentially slow mixing in the mean-field Swendsen-Wang dynamics. *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*, 56(1):68–86, 2020. doi:10.1214/18-AIHP955.
- 37 Geoffrey Grimmett. The random-cluster model. In *Probability on discrete structures*, volume 110 of *Encyclopaedia Math. Sci.*, pages 73–123. Springer, Berlin, 2004. doi:10.1007/978-3-662-09444-0\_2.
- 38 Heng Guo and Mark Jerrum. Random cluster dynamics for the Ising model is rapidly mixing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1818–1827, 2017.
- 39 Olle Häggström. The random-cluster model on a homogeneous tree. *Probability Theory and Related Fields*, 104(2):231–253, 1996. doi:10.1007/BF01247839.
- 40 Matan Harel and Yinon Spinka. Finitary codings for the random-cluster model and other infinite-range monotone models. *Electronic Journal of Probability*, 27:1–32, 2022.
- 41 Thomas P Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 39–46. IEEE, 2006.
- 42 Tyler Helmuth, Matthew Jenssen, and Will Perkins. Finite-size scaling, phase coexistence, and algorithms for the random cluster model on random graphs, 2020. arXiv:2006.11580.
- 43 Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001.
- 44 Johan Jonasson. The random cluster model on a general graph and a phase transition characterization of nonamenability. *Stochastic Processes and their Applications*, 79(2):335–354, 1999.
- 45 Jeong Han Kim. Poisson cloning model for random graphs. In *International Congress of Mathematicians (ICM)*, 2006.
- 46 David A. Levin and Yuval Peres. Markov chains and mixing times (second edition). *The Mathematical Intelligencer*, 41(1):90–91, 2019. doi:10.1007/s00283-018-9839-x.
- 47 Yun Long, Asaf Nachmias, Weiyang Ning, and Yuval Peres. A power law of order 1/4 for critical mean field Swendsen-Wang dynamics. *Mem. Amer. Math. Soc.*, 232(1092), 2014. doi:10.1090/memo/1092.
- 48 Russell Lyons. The Ising model and percolation on trees and tree-like graphs. *Communications in Mathematical Physics*, 125(2):337–353, 1989. doi:cmp/1104179469.

- 49 A. Montanari and A. Saberi. The spread of innovations in social networks. *Proceedings of the National Academy of Sciences*, 107(47):20196–20201, 2010.
- 50 Elchanan Mossel and Allan Sly. Rapid mixing of Gibbs sampling on graphs that are sparse on average. *Random Structures & Algorithms*, 35(2):250–270, 2009.
- 51 Elchanan Mossel and Allan Sly. Exact thresholds for Ising–Gibbs samplers on general graphs. *Ann. Probab.*, 41(1):294–328, January 2013. doi:10.1214/11-AOP737.
- 52 S. Osindero and G.E. Hinton. Modeling image patches with a directed hierarchy of Markov random fields. In *Advances in neural information processing systems*, pages 1121–1128, 2008.
- 53 Yuval Peres and Peter Winkler. Can extra updates delay mixing? *Communications in Mathematical Physics*, 323(3):1007–1016, 2013. doi:10.1007/s00220-013-1776-0.
- 54 S. Roth and M.J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867, 2005.
- 55 Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.*, 58:86–88, January 1987. doi:10.1103/PhysRevLett.58.86.
- 56 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Annual ACM symposium on Theory of computing (STOC)*, pages 343–350, 2000.
- 57 Mario Ullrich. Swendsen–Wang is faster than single-bond dynamics. *SIAM Journal on Discrete Mathematics*, 28(1):37–48, 2014. doi:10.1137/120864003.



# Improved Bounds for Randomly Colouring Simple Hypergraphs

Weiming Feng 

School of Informatics, University of Edinburgh, UK

Heng Guo 

School of Informatics, University of Edinburgh, UK

Jiaheng Wang 

School of Informatics, University of Edinburgh, UK

---

## Abstract

We study the problem of sampling almost uniform proper  $q$ -colourings in  $k$ -uniform simple hypergraphs with maximum degree  $\Delta$ . For any  $\delta > 0$ , if  $k \geq \frac{20(1+\delta)}{\delta}$  and  $q \geq 100\Delta^{\frac{2+\delta}{k-4/\delta-4}}$ , the running time of our algorithm is  $\tilde{O}(\text{poly}(\Delta k) \cdot n^{1.01})$ , where  $n$  is the number of vertices. Our result requires fewer colours than previous results for general hypergraphs (Jain, Pham, and Vuong, 2021; He, Sun, and Wu, 2021), and does not require  $\Omega(\log n)$  colours unlike the work of Frieze and Anastos (2017).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains

**Keywords and phrases** Approximate counting, Markov chain, Mixing time, Hypergraph colouring

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.25

**Category** RANDOM

**Related Version** Full Version: <https://arxiv.org/abs/2202.05554> [7]

**Funding** This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947778).

## 1 Introduction

The past few years have witnessed a bloom in techniques targeted at approximate counting and sampling problems, among which constraint satisfaction problems (CSPs) are probably the most studied. In fact, many problems can be cast as CSPs, e.g., Boolean satisfiability problems (SATs), proper colourings of graphs and hypergraphs, and independent sets, to name a few. In general, even deciding if a CSP instance can be satisfied or not is **NP**-hard. However, efficient algorithms become possible when the number of appearances of each variable (usually referred to as the degree) is not too high. For these instances, the Lovász Local Lemma [6] provides a fundamental criterion to guarantee the existence of a solution. Although the original local lemma does not provide an efficient algorithm, after two decades of effort [2, 1, 28, 5, 32, 29], the celebrated work of Moser and Tardos [30] provides an efficient algorithm matching the same conditions as the local lemma.

Unfortunately, the output distribution of the Moser–Tardos algorithm does not suit the need of approximate counting and sampling. This deficiency is fundamental, as it can be **NP**-hard to (uniformly or near-uniformly) sample satisfying assignments even when the criterion of the local lemma is satisfied and the corresponding searching problem lies in **P** [3, 14].<sup>1</sup> In other words, sampling problems are fundamentally more difficult than searching problems in the local lemma regime. Part of the difficulty comes from the possibility that

---

<sup>1</sup> As far as we are aware, all hardness results for sampling (including the ones mentioned in this paper) allow errors in total variation distances. It is not clear if stronger hardness results exist for perfect sampling (i.e. with no error).



© Weiming Feng, Heng Guo, and Jiaheng Wang;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 25; pp. 25:1–25:17



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the state space can be disconnected from local moves, but traditional algorithmic tools like Markov chain Monte Carlo rely on the connectivity. This barrier has been bypassed recently by some exciting developments [27, 16, 17, 23], and in particular the projected Markov chain approach [8, 9, 24, 19]. For searching problems, the local lemma is known to give a sharp computational transition threshold from **P** to **NP**-hard [30, 15] as the degree increases. Recent efforts aim to find and establish a similar threshold for sampling problems as well.

One very promising problem to establish such a threshold is (proper)  $q$ -colourings of hypergraphs, which is the original setting where the local lemma was developed [6], and has received considerable recent attention. A hypergraph  $H = (V, \mathcal{E})$  consists of a set of vertices  $V$  and a set of hyperedges  $\mathcal{E} \subseteq 2^V$ . We say  $H = (V, \mathcal{E})$  is  $k$ -uniform, if every hyperedge  $e \in \mathcal{E}$  satisfies  $|e| = k$ . A colouring of a hypergraph is *proper* if no hyperedge is monochromatic. An efficient (perfect) sampler exists when  $q \gtrsim \Delta^{3/(k-4)}$  (where  $\gtrsim$  or  $\lesssim$  hides some constant independent from  $q$ ,  $k$ , and  $\Delta$ ) for  $k$ -uniform hypergraphs with maximum degree  $\Delta$  [24, 19], while the sampling problem is **NP**-hard whenever  $q \lesssim \Delta^{2/k}$  for even  $q$  [14]. For comparison, the local lemma shows that a proper  $q$ -colouring exists if  $q \gtrsim \Delta^{1/(k-1)}$  (see also [35] for a recent alternative approach leading to a slightly better constant).

On the other hand, before the recent wave of local lemma inspired sampling algorithms, randomly sampling  $q$ -colourings in *simple*  $k$ -uniform hypergraphs<sup>2</sup> has already been studied [12, 11]. In particular, Frieze and Anastos [11] gave an efficient sampling algorithm when the number of colours satisfies  $q \geq \max\{C_k \log n, 500k^3 \Delta^{\frac{1}{k-1}}\}$ , where  $n$  is the number of vertices and  $C_k$  depends only on  $k$ . Their algorithm is the standard Glauber dynamics with a random initial (not necessarily proper) colouring. The logarithmic lower bound on the number of colours is crucial to their analysis, as it guarantees that there is a giant connected component in the state space so that connectivity is not an issue.

In this paper, we study the projected Markov chain for sampling  $q$ -colourings in simple hypergraphs. Our result improves the bound of [24, 19] for general hypergraphs, and does not require unbounded number of colours, unlike in [12, 11]. Let  $\mu$  denote the uniform distribution over all proper colourings. Our main result is stated as follows.

► **Theorem 1.** *For any  $\delta > 0$ , there is a sampling algorithm such that given any  $\epsilon \in (0, 1)$ , a  $k$ -uniform simple hypergraph  $H = (V, E)$  with maximum degree  $\Delta$ , where  $k \geq \frac{20(1+\delta)}{\delta}$ , and an integer  $q \geq 100\Delta^{\frac{2+\delta}{k-4/\delta-4}}$ , it returns a random  $q$ -colouring that is  $\epsilon$ -close to  $\mu$  in total variation distance in time  $\tilde{O}(k^5 \Delta^2 n (\frac{n\Delta}{\epsilon})^{0.01})$ , where  $n = |V|$  and  $\tilde{O}$  hides a polylog( $n, \Delta, q, 1/\epsilon$ ) factor.*

A few quick remarks are in order. First of all, the exponent of  $n$  in the running time can be made even closer to 1 if more colours are given. See Theorem 10 for the full technical statement. Secondly, our algorithm can be modified into a perfect sampler by applying the bounding chain method [21] based on coupling from the past (CFTP) [31], following the same lines of [19]. Moreover, using known reductions from approximate counting to sampling [25, 34, 22, 26] (see [8] for simpler arguments specialized to local lemma settings), one can efficiently and approximately count the number of proper colourings in simple hypergraphs under the same conditions in Theorem 1.

Our algorithm follows the recent projected Markov chain approach [8] with state compression [9]. Roughly speaking, instead of assigning colours to vertices, we split  $[q]$  into  $\sqrt{q}$  buckets of size  $\sqrt{q}$  each and assign buckets to vertices. We run a (systematic scan) Markov chain on these bucket assignments to generate a sample, and then conditional on this sample to draw a nearly uniform  $q$ -colouring. The benefit of this bucketing is that, under the conditions of

<sup>2</sup> A hypergraph is *simple* if any two hyperedges intersect in at most one vertex. Simple hypergraphs are also known as linear hypergraphs.



Theorem 1, conditional on the assignments of all but one vertices, the assignment of the remaining vertex is close to uniformly at random. This implies that any atomic event<sup>3</sup> is exponentially unlikely in the number of distinct vertices it depends on. In order to show that this approach works, we need to show two things: 1) the projected Markov chain is rapidly mixing; 2) each step of the Markov chain can be efficiently implemented. For general hypergraphs, the previous  $q \gtrsim \Delta^{3/(k-4)}$  bound comes from balancing the conditions so that the two claims are true simultaneously. However, there is no room left for relaxation on either claim. This means that, for our improvements in simple hypergraphs, new ingredients are required for both claims.

For rapid mixing, we take the information percolation approach [20, 24, 19], where the main effort is to trace discrepancies through a one-step greedy coupling, and to show that they are unlikely after a sufficient amount of time. In simple hypergraphs, an individual discrepancy path through time has more distinct updates of vertices than in the general case, and are thus more unlikely. This allows us to relax the condition. Our mixing time analysis is largely inspired by the work of Hermon, Sly, and Zhang [20], although we do need to handle some new complications, such as hyperedges whose vertices are consecutively updated in the discrepancy path.

For efficient implementation, we use rejection sampling. Here we want to sample the colour/bucket of a vertex conditional on the buckets of all other vertices. We can safely prune hyperedges containing vertices of different buckets. The remaining connected component containing the update vertex needs to have logarithmic size to guarantee efficiency of our rejection sampling. The standard approach to bound its size is to do a union bound over certain combinatorial structures with sufficiently many distinct vertices. Most previous analysis is based on enumerating so-called “2-trees”, a notion first introduced by Alon [1]. Unfortunately, under the conditions of Theorem 1, there are too many “2-trees” to our need. Instead, we introduce a new structure called “2-block-trees” (see Definition 15). Here each “block” is a collection of  $\theta$  connected hyperedges, and these blocks satisfy connectivity properties similar to a 2-tree. Since the hypergraph is simple, a block has at least  $\theta k - \binom{\theta}{2}$  distinct vertices. As long as  $\theta \ll k$ , we have a good lower bound on the number of distinct vertices, which in turn implies a good upper bound on the probability of these structures showing up. To finish off with the union bound, we give a new counting argument for the number of 2-block-trees, which is based on finding a good encoding of these structures.

The exponent (roughly  $2/k$ ) of  $\Delta$  in Theorem 1 is unlikely to be tight, although it appears to be the limit of current techniques. In fact, we conjecture that the computational transition for sampling  $q$ -colourings in simple hypergraphs happens around the same threshold of the local lemma (namely, the exponent should be roughly  $1/k$ ). This conjecture is supported by the hardness result of Galanis, Guo, and Wang [14] for general  $q$ , and by the algorithm of Frieze and Anastos [11] for  $q = \Omega(\log n)$ . Note that for a simple  $k$ -uniform hypergraph with maximum degree  $\Delta$ , Frieze and Mubayi [10] showed that the chromatic number  $\chi(H) \leq C_k \left( \frac{\Delta}{\log \Delta} \right)^{\frac{1}{k-1}}$  where  $C_k$  depends only on  $k$ . Their bound is asymptotically better than the bound given by the local lemma. Thus there may still be a gap between the searching threshold and the sampling threshold.

A final remark is that our method would still work as long as the overlap of hyperedges is much smaller than  $k$ . The condition on the parameters will deteriorate slightly but would still be better than those for general hypergraphs. In light of this, our method can potentially

<sup>3</sup> An event is *atomic* if each variable it depends on must take one particular value. In discrete spaces, any event can be decomposed into atomic ones.

be applied to improve the constant of [13] on sampling solutions to random CNF formulas, where the overlaps are at most 2 with high probability. On the other end of the spectrum, if any two intersecting hyperedges intersect at at least  $k/2$  vertices, the algorithm by Guo, Jerrum, and Liu [16] almost matches the hardness result [14]. It is an intriguing question how the size of overlaps affects the complexity of these sampling problems, or whether it is possible to improve sampling algorithms via a better use of the overlap information.

Due to space limitation, we omit many proofs and only give sketches for a few important results. Complete proofs can be found in the full version.

## 2 Preliminaries

In this section we gather some preliminary definitions and results for later use. We generally use the bold font to denote vectors, matrices, and/or random variables. The notation  $[q]$  stands for  $\{1, 2, \dots, q\}$ .

### 2.1 Graph theory

Throughout this paper, we use the following notations for a graph  $G = (V, E)$ :

- $G[A]$ : the induced subgraph of  $G$  on the vertex subset  $A \subseteq V$ .
- $\text{dist}_G(A, B)$ : the distance between two vertex sets  $A \subseteq V$  and  $B \subseteq V$  on  $G$ , which is defined by  $\text{dist}_G(A, B) := \min_{u \in A, v \in B} \text{dist}_G(u, v)$  and  $\text{dist}_G(u, v)$  is the length of the shortest path between  $u$  and  $v$  in  $G$ .
- $\Gamma_G^i(A)$ : the set of vertices  $u$  such that  $\text{dist}_G(A, u) = i$ . Specifically, when  $i = 1$ , this notation represents the neighbourhood of the given set  $A \subseteq V$ , and is also denoted by  $\Gamma_G(A)$ .

We sometimes do not distinguish  $u$  and the singleton set  $\{u\}$  in sub- or sup-scripts. For the sake of convenience, we may drop the subscript  $G$  when the underlying graph is clear from the context.

We need some more definitions for later use.

► **Definition 2** (Graph power). *Let  $G$  be an undirected graph. The  $i$ -th power of  $G$ , denoted by  $G^i$ , is another graph that has the same vertex set as  $G$ , and  $\{u, v\}$  is an edge in  $G^i$  iff  $1 \leq \text{dist}_G(u, v) \leq i$ .*

► **Definition 3** (Line graph). *Let  $H = (V, \mathcal{E})$  be a hypergraph. Its line graph  $\text{Lin}(H) = (V_L, E_L)$  is given by  $V_L = \mathcal{E}$ , and  $\{e, e'\} \in E_L$  iff  $e \cap e' \neq \emptyset$ .*

### 2.2 Coupling and Markov chains

Consider a discrete state space  $\Omega$  and two distributions  $\mu$  and  $\nu$  over it. The *total variation distance* between  $\mu$  and  $\nu$  is defined by

$$d_{\text{TV}}(\mu, \nu) := \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)|.$$

A *coupling* between  $\mu$  and  $\nu$  is a joint distribution  $(X, Y) \in \Omega^2$  such that its marginal distribution over  $X$  (resp.  $Y$ ) is  $\mu$  (resp.  $\nu$ ). The next lemma, usually referred to as the *coupling lemma*, bounds the total variation distance between  $\mu$  and  $\nu$  by any of their couplings.

► **Lemma 4** (Coupling lemma). *For any coupling  $(X, Y)$  between  $\mu$  and  $\nu$ ,  $d_{\text{TV}}(\mu, \nu) \leq \Pr[X \neq Y]$ . Moreover, there exists an optimal coupling reaching the equality.*

Given a finite state space  $\Omega$ , a discrete-time *Markov chain* is a sequence  $\{X_t\}_{t \geq 0}$  where the probability of each possible state of  $X_{t+1}$  only depends on the state of  $X_t$ . The transition of the chain is represented by the *transition matrix*  $\mathbf{P} : \Omega^2 \rightarrow \mathbb{R}_{[0,1]}$ , where  $\mathbf{P}(i, j) = \Pr[X_{t+1} = j \mid X_t = i]$ . When the state space  $\Omega$  is clear from context, we simply denote the chain by its transition matrix. A Markov chain  $\mathbf{P}$  is:

- *irreducible*, if for any  $X, Y \in \Omega$ , there exists  $t > 0$  such that  $\mathbf{P}^t(X, Y) > 0$ ;
- *aperiodic*, if for all  $X \in \Omega$ , it holds that  $\gcd\{t \mid \mathbf{P}^t(X, X) > 0\} = 1$ ; and
- *reversible* with respect to a distribution  $\pi$ , if  $\forall X, Y \in \Omega$ ,  $\pi(X)\mathbf{P}(X, Y) = \pi(Y)\mathbf{P}(Y, X)$ . This equation is usually known as the *detailed balance condition*.

A distribution  $\pi$  is *stationary* for  $\mathbf{P}$ , if  $\pi\mathbf{P} = \pi$  (regarding  $\pi$  as a row vector). The detailed balance condition actually implies that the corresponding distribution is stationary. Furthermore, if a Markov chain is both irreducible and aperiodic, then it converges to a unique stationary distribution  $\pi$ . The speed of convergence towards  $\pi$  is characterised by its *mixing time*, defined by

$$t_{\text{mix}}(\mathbf{P}, \epsilon) := \min \left\{ t \mid \max_{X \in \Omega} d_{\text{TV}}(\mathbf{P}^t(X, \cdot), \pi) < \epsilon \right\}.$$

The joint process  $(X_t, Y_t)_{t \geq 0}$  is a *coupling of Markov chain*  $\mathbf{P}$  if  $(X_t)_{t \geq 0}$  and  $(Y_t)_{t \geq 0}$  individually follow the transition rule of  $\mathbf{P}$ , and if  $X_i = Y_i$  then  $X_j = Y_j$  for all  $j \geq i$ . By the coupling lemma, for any coupling  $(X_t, Y_t)_{t \geq 0}$  of  $\mathbf{P}$ , it holds that  $d_{\text{TV}}(\mathbf{P}^t(X_0, \cdot), \mathbf{P}^t(Y_0, \cdot)) \leq \Pr[X_t \neq Y_t]$ . Hence, the mixing time of  $\mathbf{P}$  can be bounded by

$$t_{\text{mix}}(\mathbf{P}, \epsilon) \leq \max_{X_0, Y_0 \in \Omega} \min \{t \mid \Pr[X_t \neq Y_t] \leq \epsilon\}. \quad (1)$$

### 2.3 Lovász Local Lemma

Let  $\mathcal{R} = \{R_1, \dots, R_n\}$  be a set of mutually independent random variables. Given an event  $A$ , denote the set of variables that determines  $A$  by  $\text{vbl}(A) \subseteq \mathcal{R}$ . Let  $\mathcal{B} = \{B_1, \dots, B_n\}$  be a collection of “bad” events. For any event  $A$  (not necessarily in  $\mathcal{B}$ ), let  $\Gamma(A) := \{B \in \mathcal{B} \mid B \neq A, \text{vbl}(B) \cap \text{vbl}(A) \neq \emptyset\}$ . We will use the following version of *Lovász Local Lemma* from [18].

► **Theorem 5** ([6, 18]). *If there exists a function  $x : \mathcal{B} \rightarrow (0, 1)$  such that for any bad event  $B \in \mathcal{B}$ ,*

$$\Pr[B] \leq x(B) \prod_{B' \in \Gamma(B)} (1 - x(B')), \quad (2)$$

*then it holds that*

$$\Pr \left[ \bigwedge_{B \in \mathcal{B}} \bar{B} \right] \geq \prod_{B \in \mathcal{B}} (1 - x(B)) > 0.$$

*Moreover, for any event  $A$ ,*

$$\Pr \left[ A \mid \bigwedge_{B \in \mathcal{B}} \bar{B} \right] \leq \Pr[A] \prod_{B \in \Gamma(A)} (1 - x(B))^{-1}. \quad (3)$$

## 2.4 List hypergraph colouring and local uniformity

In our algorithm and analysis, we consider the general *list hypergraph colouring* problem. Let  $H = (V, \mathcal{E})$  be a  $k$ -uniform hypergraph with maximum degree  $\Delta$ . Let  $(Q_v)_{v \in V}$  be a set of colour lists. We say  $\mathbf{X} \in \otimes_{v \in V} Q_v$  is a proper list colouring if no hyperedge in  $H$  is monochromatic with respect to  $\mathbf{X}$ . Let  $\mu$  denote the uniform distribution of all proper list hypergraph colourings. The following local uniformity property holds for the distribution  $\mu$ . Its proof follows from the argument in [17].

► **Lemma 6** (local uniformity [17]). *Let  $q_0 = \min_{v \in V} |Q_v|$  and  $q_1 = \max_{v \in V} |Q_v|$ . For any  $r \geq k \geq 2$ , if  $q_0^k \geq eq_1 r \Delta$ , then for any  $v \in V$  and  $c \in Q_v$ ,*

$$\frac{1}{|Q_v|} \exp\left(-\frac{2}{r}\right) \leq \mu_v(c) \leq \frac{1}{|Q_v|} \exp\left(\frac{2}{r}\right),$$

where  $\mu_v$  is the marginal distribution on  $v$  induced by  $\mu$ .

## 3 Algorithm

Let  $H = (V, \mathcal{E})$  be a  $k$ -uniform hypergraph and  $[q]$  a set of colours. Let  $\mu$  denote the uniform distribution of proper hypergraph colourings. Our algorithm is a variant of the projected dynamics from [8], using a particular projection scheme from [9]. We first introduce some basic definitions and notations, and then describe the sampling algorithm.

### 3.1 Projection scheme, projected distribution and conditional distribution

Our sampling algorithm is based on the following *projection scheme* introduced in [9].

► **Definition 7** (projection scheme [9]). *Let  $1 \leq s \leq q$  be an integer. A (balanced) projection scheme with image size  $s$  is a function  $h : [q] \rightarrow [s]$  such that for any  $j \in [s]$ ,  $|h^{-1}(j)| = \lfloor \frac{q}{s} \rfloor$  or  $|h^{-1}(j)| = \lceil \frac{q}{s} \rceil$ .*

For any  $\mathbf{X} \in [q]^V$ , define the projection *image*  $\mathbf{Y} \in [s]^V$  of  $\mathbf{X}$  by

$$\forall v \in V, \quad Y_v = h(X_v).$$

For simplicity, we often denote  $\mathbf{Y} = h(\mathbf{X})$ , and for any subset  $\Lambda \subseteq V$ , we denote  $\mathbf{Y}_\Lambda = h(\mathbf{X}_\Lambda)$ .

Given a projection scheme, the following *projected distribution* can be naturally defined.

► **Definition 8** (projected distribution). *Given a projection scheme  $h$ , the projected distribution  $\nu$  is the distribution of  $\mathbf{Y} = h(\mathbf{X})$ , where  $\mathbf{X} \sim \mu$ .*

Given an image of the projection, we can define the following *conditional distribution* over  $[q]^V$ .

► **Definition 9** (conditional distribution). *Let  $\Lambda \subseteq V$  be a subset of vertices. Given a (partial) image  $\sigma_\Lambda \in [s]^\Lambda$ , the conditional distribution  $\mu^{\sigma_\Lambda}$  is the distribution of  $\mathbf{X} \sim \mu$  conditional on  $h(\mathbf{X}_\Lambda) = \sigma_\Lambda$ .*

By definition,  $\mu^{\sigma_\Lambda}$  is a distribution over  $[q]^V$ . We use  $\mu_S^{\sigma_\Lambda}$  to denote the marginal distribution on  $S \subseteq V$  projected from  $\mu^{\sigma_\Lambda}$ , and we simply denote  $\mu_{\{v\}}^{\sigma_\Lambda}$  by  $\mu_v^{\sigma_\Lambda}$ .

■ **Algorithm 1** Sampling algorithm for hypergraph colouring.

---

**Input:** A hypergraph  $H = (V, \mathcal{E})$ , a set of colours  $[q]$ , an error bound  $0 < \epsilon < 1$ , and a balanced projection scheme  $h : [q] \rightarrow [s]$ , where  $s = \lceil \sqrt{q} \rceil$

**Output:** A random colouring  $\mathbf{X} \in [q]^V$

- 1 sample  $\mathbf{Y} \in [s]^V$  uniformly at random;
- 2 **for**  $t$  from 1 to  $T = \lceil 50n \log \frac{2n\Delta}{\epsilon} \rceil$  **do**
- 3     let  $v$  be the vertex with label  $(t \bmod n)$ ;
- 4      $X'_v \leftarrow \text{Sample}(H, h, \{v\}, \mathbf{Y}_{V \setminus \{v\}}, \frac{\epsilon}{4T})$ ;
- 5     /\* The Sample subroutine is given in Algorithm 2. \*/
- 5      $Y_v \leftarrow h(X'_v)$ ;
- 6 **return**  $\mathbf{X} \leftarrow \text{Sample}(H, h, V, \mathbf{Y}, \frac{\epsilon}{4T})$ ;

---

### 3.2 The sampling algorithm

In this section and what follows, we always assume that all vertices in  $V$  are labeled by  $\{0, 1, \dots, n-1\}$ . We also fix the parameter  $s = \lceil \sqrt{q} \rceil$ . Given a projection scheme  $h$  with image size  $s$ , our sampling algorithm first samples  $\mathbf{Y} \in [s]^V$  from the projected distribution  $\nu$ , and then uses it to sample a random hypergraph colouring from the conditional distribution  $\mu^{\mathbf{Y}}$ . The pseudocode is given in Algorithm 1.

The main ingredient of Algorithm 1 is the part that samples  $\mathbf{Y}$  (Line 1 to Line 5). It is basically a *systematic scan* version of the Glauber dynamics for  $\nu$ . In order to update the state of a particular vertex, we invoke a subroutine **Sample**, given in Algorithm 2, to sample  $X'_v$  first from the distribution conditional on  $\mathbf{Y}_{V \setminus \{v\}}$ . Also, **Sample** is used to generate the random colouring conditional on  $\mathbf{Y}$  in Line 6. The subroutine **Sample** in fact returns an approximate sample with high probability. Here we have to settle with some small error because exactly calculating the conditional distribution is intractable. To implement **Sample**, we use standard rejection sampling, which is described in Algorithm 3. Showing the correctness and efficiency of Algorithm 2 and Algorithm 3 is one of our main contributions.

In the following we flesh out the outline above. Let  $\Lambda \subseteq V$  and  $\mathbf{Y}_\Lambda \in [s]^\Lambda$ . Note that during the execution of Algorithm 1,  $\mathbf{Y}_\Lambda$  is a random input to **Sample**. Let  $S \subseteq V$  and  $\zeta \in (0, 1)$ . The subroutine **Sample**( $H, h, S, \mathbf{Y}_\Lambda, \zeta$ ) in Algorithm 1 returns a random sample  $\mathbf{X}_S \in [q]^S$  such that with probability at least  $1 - \zeta$ , the total variation distance between  $\mathbf{X}_S$  and  $\mu_S^{\mathbf{Y}_\Lambda}$  is at most  $\zeta$ , where the probability is taken over the randomness of the input  $\mathbf{Y}_\Lambda$ .

In the  $t$ -th step of the systematic scan in Algorithm 1, we pick the vertex  $v$  with label  $(t \bmod n)$ , and use Line 4 and Line 5 to update the value of  $Y_v$ . Ideally, we want to resample the value of  $Y_v$  according to the conditional distribution  $\nu_v^{\mathbf{Y}_{V \setminus \{v\}}}$ , where  $\nu$  is the distribution projected from  $\mu$ . However, exactly computing the conditional distribution is not tractable, and we approximate it by projecting from the random sample  $X'_v \in [q]$  given by **Sample** in Line 4. It is straightforward to verify that  $Y_v$  approximately follows the law of  $\nu_v^{\mathbf{Y}_{V \setminus \{v\}}}$  as long as  $X'_v$  approximately follows the law of  $\mu_v^{\mathbf{Y}_{V \setminus \{v\}}}$ . In the last step, we use **Sample** to draw approximate samples from the conditional distribution  $\mu^{\mathbf{Y}}$ .

We explain the details of **Sample**( $H, h, S, \mathbf{Y}_\Lambda, \zeta$ ) next. First we need some notations. Given a partial image  $\mathbf{Y}_\Lambda$ , we say an hyperedge  $e \in \mathcal{E}$  is satisfied by  $\mathbf{Y}_\Lambda$  if there exists  $u, v \in e \cap \Lambda$  such that  $Y_u \neq Y_v$ . In other words, for all  $\mathbf{X} \in [q]^V$  such that  $\mathbf{Y}_\Lambda = h(\mathbf{X}_\Lambda)$ , the hyperedge  $e$  is not monochromatic with respect to  $\mathbf{X}$ , and thus  $e$  is always “satisfied” given

■ **Algorithm 2** Sample( $H, h, S, \mathbf{Y}_\Lambda, \zeta$ ).

---

**Input:** A hypergraph  $H = (V, \mathcal{E})$ , a projection scheme  $h : [q] \rightarrow [s]$ , a subset  $S \subseteq V$ , a (partial) image  $\mathbf{Y}_\Lambda \in [s]^\Lambda$  where  $\Lambda \subseteq V$ , and an error bound  $\zeta \in (0, 1)$

**Output:** A random (partial) colouring  $\mathbf{X}_S \in [q]^S$

- 1 remove all hyperedges in  $H$  that are satisfied by  $\mathbf{Y}_\Lambda$  to obtain  $H^{\mathbf{Y}_\Lambda} = (V, \mathcal{E}^{\mathbf{Y}_\Lambda})$ ;
- 2 let  $H_i = (V_i, \mathcal{E}_i^{\mathbf{Y}_\Lambda})$  for  $1 \leq i \leq \ell$  be the connected components such that  $V_i \cap S \neq \emptyset$ ;
- 3 **if**  $\exists 1 \leq i \leq \ell$  such that  $|\mathcal{E}_i^{\mathbf{Y}_\Lambda}| > 4\Delta k^3 \log\left(\frac{n\Delta}{\zeta}\right)$  **then**
- 4     **return**  $\mathbf{X}_S \in [q]^S$  uniformly at random;
- 5 **for**  $i$  from 1 to  $\ell$  **do**
- 6      $\mathbf{X}_i \leftarrow \text{RejectionSampling}(H_i, h, \mathbf{Y}_{\Lambda \cap V_i}, R)$ , where  $R = \left\lceil 10 \left(\frac{n\Delta}{\zeta}\right)^{\frac{1}{1000\eta}} \log \frac{n}{\zeta} \right\rceil$ ;
- 7     /\* The RejectionSampling subroutine is given in Algorithm 3. \*/
- 8     **if**  $\mathbf{X}_i = \perp$  **then**
- 9         **return**  $\mathbf{X}_S \in [q]^S$  uniformly at random ;
- 10 **return**  $\mathbf{X}_S$  where  $\mathbf{X} = \biguplus_{i=1}^\ell \mathbf{X}_i$ ;

---

$\mathbf{Y}_\Lambda$ . Let  $H^{\mathbf{Y}_\Lambda} = (V, \mathcal{E}^{\mathbf{Y}_\Lambda})$  be the hypergraph obtained from  $H$  by removing all hyperedges satisfied by  $\mathbf{Y}_\Lambda$ . Let  $H_1^{\mathbf{Y}_\Lambda}, H_2^{\mathbf{Y}_\Lambda}, \dots, H_m^{\mathbf{Y}_\Lambda}$  denote the connected components of  $H^{\mathbf{Y}_\Lambda}$ , where  $H_i^{\mathbf{Y}_\Lambda} = (V_i, \mathcal{E}_i^{\mathbf{Y}_\Lambda})$ . The following fact is straightforward to verify

$$\mu^{\mathbf{Y}_\Lambda} = \mu_1^{\mathbf{Y}_{\Lambda \cap V_1}} \times \mu_2^{\mathbf{Y}_{\Lambda \cap V_2}} \times \dots \times \mu_m^{\mathbf{Y}_{\Lambda \cap V_m}},$$

where  $\mu_i$  is the uniform distribution over proper  $q$ -colourings of the sub-hypergraph  $H_i^{\mathbf{Y}_\Lambda}$  (namely,  $\mu_i^{\mathbf{Y}_{\Lambda \cap V_i}}$  is the uniform distribution over list colourings of  $H_i^{\mathbf{Y}_\Lambda}$  conditional on  $\mathbf{Y}_{\Lambda \cap V_i}$ ). Without loss of generality, we assume  $S \cap V_j \neq \emptyset$  for  $1 \leq j \leq \ell$ . To draw a random sample from  $\mu^{\mathbf{Y}_\Lambda}$ , it suffices to draw a random sample from the product distribution  $\mu_1^{\mathbf{Y}_{\Lambda \cap V_1}} \times \mu_2^{\mathbf{Y}_{\Lambda \cap V_2}} \times \dots \times \mu_\ell^{\mathbf{Y}_{\Lambda \cap V_\ell}}$ , which we will do by drawing from each  $\mu_i^{\mathbf{Y}_{\Lambda \cap V_i}}$  individually using standard rejection sampling (given in Algorithm 3).

One final detail about Algorithm 2 and Algorithm 3 is about their efficiency. Basically we set some thresholds to guard against two unlikely bad events. We break out from the normal execution immediately and return an arbitrary random sample if one of the following two bad events occur:

- for some  $1 \leq i \leq \ell$ ,  $|\mathcal{E}_i^{\mathbf{Y}_\Lambda}| > 4\Delta k^3 \log\left(\frac{n\Delta}{\zeta}\right)$ ;
- for some  $1 \leq i \leq \ell$ , the rejection sampling for  $\mu_i^{\mathbf{Y}_{\Lambda \cap V_i}}$  fails after  $R$  trials, where

$$R := \left\lceil 10 \left(\frac{n\Delta}{\zeta}\right)^{\frac{1}{1000\eta}} \log \frac{n}{\zeta} \right\rceil \quad \text{and} \quad \eta := \frac{1}{\Delta} \left(\frac{q}{100}\right)^{\frac{k-3}{2}}. \quad (4)$$

In the analysis (see Lemma 13), we will show that both of the two bad events above occur with low probability, and thus with high probability the **Sample** subroutine returns an approximate sample with desired accuracy.

■ **Algorithm 3**  $\text{RejectionSampling}(H, h, Y_\Lambda, R)$ .

---

**Input:** A hypergraph  $H = (V, \mathcal{E})$ , a projection scheme  $h : [q] \rightarrow [s]$ , a (partial) image  $Y_\Lambda \in [s]^\Lambda$  where  $\Lambda \subseteq V$  and an integer  $R$

**Output:** A random colouring  $\mathbf{X} \in [q]^V$  or a special symbol  $\perp$

- 1 for each  $v \in V$ , let  $Q_v \leftarrow h^{-1}(Y_v)$  if  $v \in \Lambda$ , and  $Q_v \leftarrow [q]$  if  $v \notin \Lambda$ ;
- 2 **for**  $i$  from 1 to  $R$  **do**
- 3     sample  $X_v \in Q_v$  uniformly at random for all  $v \in V$  and let  $\mathbf{X} = (X_v)_{v \in V}$ ;
- 4     **if**  $\mathbf{X}$  is a proper hypergraph colouring of  $H$  **then**
- 5         **return**  $\mathbf{X}$ ;
- 6 **return**  $\perp$ ;

---

#### 4 Proof of the main theorem

Let  $H = (V, \mathcal{E})$  be a simple  $k$ -uniform hypergraph with maximum degree  $\Delta$ . Let  $[q]$  be a set of  $q$  colours. Recall  $s = \lceil \sqrt{q} \rceil$ , where  $s$  is the parameter of projection scheme  $h$  (Definition 7). To construct  $h$ , we partition  $[q]$  into  $s$  intervals, where the first  $(q \bmod s)$  of them contains  $\lceil q/s \rceil$  elements each while the rest contains  $\lfloor q/s \rfloor$  elements each. For each  $i \in [q]$ , set

$$h(i) = j \quad \text{where } i \text{ belongs to the } j\text{-th interval.} \quad (5)$$

Note that this  $h$  satisfies Definition 7. In our algorithm,  $h$  is implemented as an oracle, supporting the following two types of queries.

- Evaluation: given  $i$ , the oracle returns  $h(i)$ .
- Inversion: given  $j$ , the oracle returns a uniform element in  $h^{-1}(j)$ .

Obviously, each query can be answered in time  $O(\log q)$  because of the construction of  $h$ .

The next theorem is a stronger form of Theorem 1. It shows that our algorithm can run in time arbitrarily close to linear in  $n$ , the number of vertices, as long as sufficiently many colours are available.

► **Theorem 10.** *The following result holds for any  $\delta > 0$  and  $0 < \alpha \leq 1$ . Given any  $\epsilon \in (0, 1)$ , any  $q$ -colouring instance on  $k$ -uniform simple hypergraph  $H = (V, E)$  with maximum degree  $\Delta$ , and a balanced projection scheme, if  $k \geq \frac{20(1+\delta)}{\delta}$  and  $q \geq 100 \left(\frac{\Delta}{\alpha}\right)^{\frac{2+\delta}{k-4/\delta-4}}$ , Algorithm 1 returns a random colouring that is  $\epsilon$ -close to  $\mu$  in total variation distance in time  $O\left(\Delta^2 k^5 n \left(\frac{n\Delta}{\epsilon}\right)^{\alpha/100} \log^4 \left(\frac{n\Delta q}{\epsilon}\right)\right)$ .*

► **Remark 11.** The parameter  $\alpha$  captures the relation between the local lemma condition and the running time of the algorithm. If  $\alpha$  becomes smaller, the condition is more confined, and the running time is closer to linear. In particular, Theorem 1 is implied by setting  $\alpha = 1$ .

We need two lemmas to prove Theorem 10. The first lemma analyses the mixing time of the idealised systematic scan. Let  $\nu$  be the projected distribution. The idealised systematic scan chain  $\mathbf{P}_{scan}$  for  $\nu$  is defined as follows. Initially, let  $\mathbf{X}_0 \in [s]^V$  be an arbitrary initial configuration. In the  $t$ -th step, the systematic scan does the following update steps.

- Pick the vertex  $v \in V$  with label  $(t \bmod n)$  and let  $X_t(V \setminus \{v\}) \leftarrow X_{t-1}(V \setminus \{v\})$ .
- Sample  $X_t(v) \sim \nu_v^{\mathbf{X}_{t-1}(V \setminus \{v\})}$ .

► **Lemma 12.** *If  $q \geq 40\Delta^{\frac{2}{k-4}}$  and  $k \geq 20$ , the systematic scan chain  $\mathbf{P}_{scan}$  for  $\nu$  is irreducible, aperiodic and reversible with respect to  $\nu$ . Furthermore, the mixing time satisfies*

$$\forall 0 < \epsilon < 1, \quad T_{\text{mix}}(\mathbf{P}_{scan}, \epsilon) \leq \left\lceil 50n \log \frac{n\Delta}{\epsilon} \right\rceil.$$



Lemma 12 is shown in Section 6.

Our next lemma analyzes the **Sample** subroutine. Let  $(\mathbf{Y}_t)_{t=0}^T$  denote the sequence of random configurations in  $[s]^V$  generated by Algorithm 1, where  $\mathbf{Y}_0 \in [s]^V$  is the initial configuration and  $\mathbf{Y}_t$  is the configuration after the  $t$ -th iteration of the for-loop. For any  $1 \leq t \leq T+1$ , consider the  $t$ -th invocation of **Sample** and define the following two bad events:

- $\mathcal{B}_{\text{com}}(t)$ : in the  $t$ -th invocation,  $\mathbf{X}_S$  is returned by Line 4 in Algorithm 2;
- $\mathcal{B}_{\text{rej}}(t)$ : in the  $t$ -th invocation,  $\mathbf{X}_S$  is returned by Line 8 in Algorithm 2.

Note that the  $(T+1)$ -th invocation of the subroutine **Sample** is in Line 6 in Algorithm 1. Let  $H = (V, \mathcal{E})$  denote the input hypergraph of Algorithm 1.

► **Lemma 13.** *For  $1 \leq t \leq T+1$ , the  $t$ -th invocation of the subroutine **Sample**  $(H, h, S, \mathbf{Y}_\Lambda, \zeta)$ , where  $h$  is given by (5), satisfies*

1. *the running time of the subroutine is bounded by  $O\left(|S|\Delta^2 k^5 \left(\frac{n\Delta}{\zeta}\right)^{\frac{1}{1000\eta}} \log^3\left(\frac{n\Delta q}{\zeta}\right)\right)$ ;*
2. *conditional on neither  $\mathcal{B}_{\text{com}}(t)$  nor  $\mathcal{B}_{\text{rej}}(t)$  occurs, the subroutine returns a perfect sample from  $\mu_S^{\mathbf{Y}_\Lambda}$ ;*
3. *if  $q \geq 100\Delta^{\frac{2}{k-3}}$  and  $k \geq 20$ , then  $\Pr[\mathcal{B}_{\text{rej}}(t)] \leq \zeta$ ;*
4. *for any  $\delta > 0$ , if  $k \geq \frac{20(\delta+1)}{\delta}$ ,  $q \geq 100\Delta^{\frac{2+\delta}{k-4/\delta-3}}$ , and  $H$  is simple, then  $\Pr[\mathcal{B}_{\text{com}}(t)] \leq \zeta$ .*

Properties 1, 2 and 3 are very standard and hold in general hypergraphs. They can be established by mimicking the argument in [8, 9]. The challenge here is to prove Property 4, which is established in Section 5.

Given Lemma 12 and Lemma 13, it is straightforward to establish Theorem 10. Lemma 12 shows that the idealized systematic scan chain is rapidly mixing, and Lemma 13 shows that our implementation of the chain is efficient. Lemma 13 also shows that the exceptions and errors in our implementation have low probability to happen. Thus, by a coupling argument and the coupling lemma, Lemma 4, the output of our algorithm is within  $\epsilon$  total variation distance to the desired one.

## 5 Analysis of connected components

In this section, we provide a proof sketch for Property 4 in Lemma 13. Note that this property needs the input hypergraph  $H$  being simple. Fix  $1 \leq t \leq T+1$ . Consider the  $t$ -th invocation of the subroutine **Sample**. If  $1 \leq t \leq T$ , we use  $v_t$  to denote the vertex picked by the  $t$ -th step of the systematic scan, i.e.  $v_t$  is the vertex with label  $(t \bmod n)$ , and let  $\Lambda := V \setminus \{v_t\}$ . If  $t = T+1$ , let  $\Lambda := V$ . Recall that  $\mathbf{Y}_t \in [s]^V$  is the random configuration generated by Algorithm 1 after the  $t$ -th iteration of the for-loop. To simplify the notation, we define  $\mathbf{Y} = \mathbf{Y}_{t-1}(\Lambda)$ , so that the input partial configuration to **Sample** is  $\mathbf{Y}$  (see Algorithm 1). Hence, we consider the subroutine **Sample**  $(H, h, S, \mathbf{Y}, \zeta)$ . Note that  $\mathbf{Y} \in [s]^\Lambda$  is a random configuration, and therefore  $H^\mathbf{Y}$  is a random hypergraph, where  $H^\mathbf{Y}$  is obtained by removing all the hyperedges in  $H$  satisfied by  $\mathbf{Y}$ . Fix an arbitrary vertex  $v \in V$ . We use  $H_v^\mathbf{Y} = (V_v^\mathbf{Y}, \mathcal{E}_v^\mathbf{Y})$  to denote the connected component in  $H^\mathbf{Y}$  that contains the vertex  $v$ . Note that  $\mathcal{E}_v^\mathbf{Y}$  can be an empty set. A hyperedge  $e \in \mathcal{E}$  is *incident* to  $v$  in the hypergraph  $H$  if  $v \in e$ . We prove the following lemma, which implies Property 4 by a straightforward union bound.

► **Lemma 14.** *For any  $\delta > 0$ , if  $k \geq \frac{20(1+\delta)}{\delta}$ ,  $q \geq 100\Delta^{\frac{2+\delta}{k-4/\delta-3}}$ , and  $H$  is simple, then for any  $v \in V$ , any  $e$  incident to  $v$  in  $H$ , it holds that*

$$\Pr_{\mathbf{Y}} \left[ e \in \mathcal{E}_v^\mathbf{Y} \wedge |\mathcal{E}_v^\mathbf{Y}| \geq 4\Delta k^3 \log \left( \frac{n\Delta}{\zeta} \right) \right] \leq \frac{\zeta}{n\Delta}.$$

Denote by  $L_H = (V_L, E_L) = \text{Lin}(H)$  the line graph of  $H$  (recall Definition 3). Let  $e$  be the hyperedge in Lemma 14 and let  $u = u_e$  be the vertex in  $L_H$  corresponding to  $e$ . Let  $L_H^Y = (V_L^Y, E_L^Y)$  denote the line graph of  $H^Y$ . Note that  $L_H^Y$  is random, and the randomness of  $L_H^Y$  is determined by the randomness of  $Y$ .

Let  $\mathcal{C} \subseteq V_L$  denote the random set of all vertices in the connected component of  $L_H^Y$  that contains the vertex  $u$ . If  $u \notin V_L^Y$ , let  $\mathcal{C} = \emptyset$ . Define an integer parameter  $\theta := \lceil \frac{4}{\delta} \rceil$ . To prove Lemma 14, it suffices to show that

$$\forall M > \theta, \quad \Pr_Y[|\mathcal{C}| \geq M] \leq \left(\frac{1}{2}\right)^{\frac{M}{2\theta k^2 \Delta} - 1}. \quad (6)$$

This is because  $k \geq \frac{20(\delta+1)}{\delta} > \lceil \frac{4}{\delta} \rceil + 1 = \theta + 1$ , and setting  $M = 4\Delta k^3 \log\left(\frac{n\Delta}{\zeta}\right)$  proves Lemma 14.

Define the following collection of subsets

$$\text{Con}_u(M) := \{C \subseteq V_L \mid u \in C \wedge |C| = M \wedge L_H[C] \text{ is connected}\}.$$

It is straightforward to verify that

$$\Pr_Y[|\mathcal{C}| \geq M] \leq \Pr_Y[\exists C \in \text{Con}_u(M) \text{ s.t. } C \subseteq V_L^Y].$$

In our proof, we partition the set  $\text{Con}_u(M)$  into two disjoint subsets  $\text{Con}_u^{(1)}(M)$  and  $\text{Con}_u^{(2)}(M)$ , and bound their corresponding probabilities separately, by showing

$$\Pr_Y[\exists C \in \text{Con}_u^{(1)}(M) \text{ s.t. } C \subseteq V_L^Y] < \left(\frac{1}{2}\right)^{\frac{M}{2\theta k^2 \Delta}}; \quad (7)$$

$$\Pr_Y[\exists C \in \text{Con}_u^{(2)}(M) \text{ s.t. } C \subseteq V_L^Y] < \left(\frac{1}{2}\right)^M. \quad (8)$$

We use Algorithm 4 to partition the set  $\text{Con}_u(M)$ . Taking as an input any  $C \in \text{Con}_u(M)$ , Algorithm 4 outputs an integer  $\ell = \ell(C)$  and disjoint sets  $C_1, C_2, \dots, C_\ell \subseteq C$ . The set  $C$  falls into  $C \in \text{Con}_u^{(1)}(M)$  if  $\ell(C) \geq \frac{M}{2\theta k^2 \Delta}$ , and  $\text{Con}_u^{(2)}(M)$  otherwise. We remark that Algorithm 4 is only used for analysis, and we do not need to implement this algorithm.

To characterise the output of Algorithm 4, we introduce a notion called “2-block-tree”.

► **Definition 15** (2-block-tree). *Let  $\theta \geq 1$  be an integer. Let  $G = (V, E)$  be a graph. A set  $\{C_1, C_2, \dots, C_\ell\}$  is a 2-block-tree with block size  $\theta$  and tree size  $\ell$  in  $G$  if*

1. *for any  $1 \leq i \leq \ell$ ,  $C_i \subseteq V$ ,  $|C_i| = \theta$ , and the induced subgraph  $G[C_i]$  is connected;*
2. *for any distinct  $1 \leq i, j \leq \ell$ ,  $\text{dist}_G(C_i, C_j) \geq 2$ ;*
3.  *$\{C_1, \dots, C_\ell\}$  is connected on  $G^2$ . (Recall Definition 2 of graph powers.)*

One can easily observe that the notion of 2-block-trees is a generalisation of 2-trees in [1] by setting  $\theta = 1$ . According to the next proposition, the output of Algorithm 4 is a 2-block-tree in  $L_H$ . This explains the name “2-block-tree generator”.

► **Proposition 16.** *The output  $\{C_1, C_2, \dots, C_\ell\}$  of Algorithm 4 satisfies that*

1.  *$\{C_1, C_2, \dots, C_\ell\}$  is a 2-block-tree in  $L_H$  with block size  $\theta$  satisfying  $u \in C_1$  and  $\cup_{i=1}^\ell C_i \subseteq C$ ;*
2. *if all vertices in  $\Gamma_G(C_i)$  are removed from  $G$ , where  $G = L_H[C]$ , then the resulting graph  $G[C']$  is a collection of connected components whose sizes are at most  $\theta$ , where  $C' = C \setminus (\cup_{i=1}^\ell \Gamma_G(C_i))$ .*

■ **Algorithm 4** 2-block-tree generator.

---

**Input:** the parameter  $\delta \in (0, 1)$  in Lemma 14, the line graph  $L_H$ , an integer  $M > \theta$ ,  
a vertex  $u$  in  $L_H$ , and a subset  $C \in \text{Con}_u(M)$

**Output:** an integer  $\ell$  and connected subgraphs  $C_1, \dots, C_\ell \subseteq C$

```

1 let $G = L_H[C] = (C, E_C)$ be the subgraph of L_H induced by C ;
2 $\theta \leftarrow \lceil \frac{4}{\delta} \rceil$, $\ell \leftarrow 0$, $V \leftarrow C$;
3 while $|V| \geq \theta$ do
4 $\ell \leftarrow \ell + 1$;
5 if $\ell = 1$ then $u_\ell \leftarrow u$;
6 if $\ell > 1$ then let u_ℓ be an arbitrary vertex in $\Gamma_G(C \setminus V)$;
7 let $C_\ell \subseteq V$ be an arbitrary connected subgraph in G such that $|C_\ell| = \theta$ and
 $u_\ell \in C_\ell$;
8 $V \leftarrow V \setminus (C_\ell \cup \Gamma_G(C_\ell))$;
9 for each connected component $G' = (V', E')$ in $G[V]$ such that $|V'| < \theta$ do
10 $V \leftarrow V \setminus V'$;
11 return $\ell, \{C_1, C_2, \dots, C_\ell\}$;
```

---

To prove (7), it is not hard to see that for any  $C \in \text{Con}_u^{(1)}(M)$ , there is a 2-block-tree tree  $\{C_1, C_2, \dots, C_\ell\}$  in the line graph  $L_H$  with block size  $\theta$  and tree size  $\ell$  satisfying

$$u \in C_1 \cup C_2 \cup \dots \cup C_\ell \quad \text{and} \quad C_1 \cup C_2 \cup \dots \cup C_\ell \subseteq C \quad (9)$$

where  $\ell = \lceil \frac{M}{2\theta k^2 \Delta} \rceil$ . We denote a 2-block-tree tree with block size  $\theta$  and tree size  $\ell$  by  $(\theta, \ell)$ -2-block-tree. This implies that

$$\begin{aligned} & \Pr_{\mathbf{Y}} \left[ \exists C \in \text{Con}_u^{(1)}(M) \text{ s.t. } C \subseteq V_L^{\mathbf{Y}} \right] \\ & \leq \Pr_{\mathbf{Y}} \left[ \exists (\theta, \ell)\text{-2-block-tree } \{C_i\} \text{ in } L_H \text{ satisfying (9) s.t. } \forall 1 \leq i \leq \ell, C_i \subseteq V_L^{\mathbf{Y}} \right]. \end{aligned}$$

We apply a union bound on the RHS over all possible 2-block-trees. The probability of any 2-block-tree is bounded by

$$\Pr_{\mathbf{Y}} [\forall 1 \leq i \leq \ell, C_i \subseteq V_L^{\mathbf{Y}}] \leq \left( (es)^\theta \left( \frac{1}{s} + \frac{1}{q} \right)^{\theta(k-\theta)} \right)^\ell. \quad (10)$$

To bound the number of 2-block-trees, we use the following lemma.

► **Lemma 17.** *Let  $\theta \geq 1$  be an integer. Let  $G = (V, E)$  be a graph with maximum degree  $d$ . For any integer  $\ell \geq 1$ , any vertex  $v \in V$ , the number of 2-block-trees  $\{C_1, C_2, \dots, C_\ell\}$  with block size  $\theta$  and tree size  $\ell$  such that  $v \in \cup_{i=1}^\ell C_i$  is at most  $(\theta e^\theta d^{\theta+1})^\ell$ .*

One may attempt Lemma 17 using the count of connected components with a degree bound in [4] based on [33]. However, it is too loose to our need, and our refined estimation in Lemma 17 is shown by a more complicated encoding argument. Our encoding has  $\ell + 1$  components. The first one encodes how  $C_i$ 's are connected in  $G^2$ , and the rest encodes each individual  $C_i$ . To encode, we need to carefully perform a depth-first-search (DFS) in  $G$  and generate an encoding for  $C_i$  whenever we meet one. The DFS generates a tree encoding how  $C_i$ 's are connected in  $G^2$ . This way, we can uniquely recover the original 2-block-tree and map each component to some subtree of a suitable infinite tree to bound their numbers.

The inequality (7) follows directly from (10) and Lemma 17.

To prove (8), we take a union bound directly over connected components of size  $M$ . Using Algorithm 4, we can get a good lower bound on the number of distinct vertices in the original hypergraph for a component. This implies that each component in  $\text{Con}_u^{(2)}(M)$  happens with probability much smaller than the total number of such components, and the union bound succeeds.

## 6 Mixing of systematic scan

In this section, we give the proof sketch of the mixing lemma for the projected systematic scan Markov chain of hypergraph colourings (Lemma 12). It is straightforward to verify that the systematic scan is aperiodic and reversible with respect to  $\nu$ . Irreducibility follows from the local lemma, Theorem 5. More precisely, Theorem 5 implies that for any  $\tau \in [s]^V$ ,  $\nu(\tau) > 0$  if  $q \geq 40\Delta^{\frac{2}{k-4}}$  and  $k \geq 20$ . Hence, the systematic scan has the unique stationary distribution  $\nu$ .

For the mixing time, the analysis is based on an information percolation argument. Define a coupling  $\mathcal{C}$  of the systematic scan  $(\mathbf{X}_t, \mathbf{Y}_t)_{t \geq 0}$ . Let  $\mathbf{X}_0, \mathbf{Y}_0 \in [s]^V$  be two arbitrary initial configurations. In the  $t$ -th transition step,

- let  $v \in V$  be the vertex with label  $(t \bmod n)$  and set  $(X_t(u), Y_t(u)) \leftarrow (X_{t-1}(u), Y_{t-1}(u))$  for all other vertices  $u \in V \setminus \{v\}$ ;
- sample  $(X_t(v), Y_t(v))$  from the optimal coupling between  $\nu_v^{X_{t-1}(V \setminus \{v\})}$  and  $\nu_v^{Y_{t-1}(V \setminus \{v\})}$ .

We prove the following lemma in this section.

► **Lemma 18.** *Suppose  $k \geq 20$  and  $q \geq 40\Delta^{\frac{2}{k-4}}$ . For any initial configurations  $\mathbf{X}_0, \mathbf{Y}_0 \in [s]^V$ , any  $\epsilon \in (0, 1)$ , let  $T = \lceil 50n \log \frac{n\Delta}{\epsilon} \rceil$ , it holds that*

$$\forall v \in V, \quad \Pr_{\mathcal{C}} [X_T(v) \neq Y_T(v)] \leq \frac{\epsilon}{n}.$$

By Lemma 18, a union bound over all vertices and the coupling lemma (Lemma 4), it holds that  $\max_{\mathbf{X}_0, \mathbf{Y}_0 \in [s]^V} d_{\text{TV}}(\mathbf{X}_T, \mathbf{Y}_T) \leq \Pr_{\mathcal{C}} [\mathbf{X}_T \neq \mathbf{Y}_T] \leq \epsilon$ , which proves the mixing time part of Lemma 12 via (1). In the rest of this section, we use the information percolation technique to analyse the coupling  $\mathcal{C}$  and prove Lemma 18.

Consider the coupling procedure  $(\mathbf{X}_t, \mathbf{Y}_t)_{t \geq 0}$ . For each  $t \geq 1$ , let  $v_t$  denote the vertex picked in the  $t$ -th step of systematic scan, namely,  $v_t$  is the vertex with label  $(t \bmod n)$ . Consider the  $t$ -th transition step, where  $t > 0$ . Define the set of agreement vertices when updating  $v_t$  at time  $t$  by  $A_t := \{v \in V \setminus \{v_t\} \mid X_{t-1}(v) = Y_{t-1}(v)\}$ . We say a hyperedge  $e \in \mathcal{E}$  is satisfied by  $A_t$  if there exist two distinct vertices  $u, v \in e \cap A_t$  such that  $X_{t-1}(u) \neq X_{t-1}(v)$  (and hence  $Y_{t-1}(u) \neq Y_{t-1}(v)$ ). We remove all the hyperedges  $e \in \mathcal{E}$  satisfied by  $A_t$  to obtain a sub-hypergraph  $H_t$ . Let  $H_t^v$  denote the connected component in  $H_t$  containing  $v$ .

► **Lemma 19.** *If  $X_t(v_t) \neq Y_t(v_t)$  for some  $t \geq 1$ , then there exists  $u \neq v_t$  in  $H_t^{v_t}$  such that  $X_{t-1}(u) \neq Y_{t-1}(u)$ .*

Lemma 19 can be proved by contradiction. Note that  $X_t(v_t)$  (resp.  $Y_t(v_t)$ ) depends only on the configuration of  $X_{t-1}$  (resp.  $Y_{t-1}(v_t)$ ) restricted on the vertices in  $H_t^{v_t}$ . If  $X_{t-1}$  and  $Y_{t-1}$  are the same on the vertices in  $H_t^{v_t}$ , then  $X_t(v_t)$  and  $Y_t(v_t)$  must be coupled perfectly.

We say that a hyperedge sequence  $e_1, e_2, \dots, e_\ell$  is a path in a hypergraph if for each  $1 < i \leq \ell$ ,  $e_{i-1} \cap e_i \neq \emptyset$  and  $e_{i-1} \neq e_i$ . The following result is a straightforward corollary of Lemma 19.

► **Corollary 20.** *Let  $t \geq 1$ . If  $X_t(v_t) \neq Y_t(v_t)$ , then there exists a vertex  $u \neq v_t$  satisfying  $X_{t-1}(u) \neq Y_{t-1}(u)$  and a path  $e_1, e_2, \dots, e_\ell$  in hypergraph  $H$  such that*

- $v \in e_1$  and  $u \in e_\ell$ ;
- for any hyperedge  $e_i$  in the path, there exists  $c \in [s]$  such that for all vertex  $w \in e_i$  and  $w \neq v_t$ , either  $X_{t-1}(w) = Y_{t-1}(w) = c$  or  $X_{t-1}(w) \neq Y_{t-1}(w)$ .

Corollary 20 is a key result for the information percolation analysis. For any time  $0 \leq t \leq T$ , any vertex  $v \in V$ , define the set of previous update times by  $S(t, v) := \{1 \leq i \leq t \mid v_i = v\}$ , where  $v_i$  is the vertex picked in the  $i$ -th transition step. Define the last update time for  $v$  up to  $t$  by

$$\text{time}_{\text{ud}}(t, v) := \begin{cases} \max_{i \in S(t, v)} i & \text{if } S(t, v) \neq \emptyset; \\ 0 & \text{otherwise.} \end{cases}$$

By Corollary 20, if the coupling on vertex  $v$  failed at time  $t$ , then there must exist a vertex  $u$  such that the coupling on  $u$  failed at time  $t' = \text{time}_{\text{ud}}(t, u)$ . We apply Corollary 20 recursively until we find a vertex  $w$  such that  $X_0(w) \neq Y_0(w)$ . This gives us an update time sequence  $t = t_1 > t_2 > \dots > t_\ell = 0$  such that the coupling of each  $t_i$ -th transition fails, together with a set of paths satisfying the properties in Corollary 20. We will show that such an update time sequence and the set of paths occur with small probability, which bounds the probability of  $X_t(v_t) \neq Y_t(v_t)$ . For this analysis, we will use the notions of extended hyperedges and extended hypergraphs introduced by He, Sun, and Wu [19].

Fix an integer  $T \geq 1$  to be the total number of transitions of the systematic scan. Define the set of extended vertex  $V^{\text{ext}}$  by

$$V^{\text{ext}} = \{(t, v_t) \mid 1 \leq t \leq T\} \cup \{(0, v) \mid v \in V\},$$

where  $v_t$  is the vertex with label  $(t \bmod n)$ . Each vertex  $(t, u) \in V^{\text{ext}}$  represents an update, i.e.  $u$  is updated at the  $t$ -th transition step. We regard all vertices “updated” at the initial step ( $t = 0$ ). Consider the systematic scan process  $(\mathbf{X}_t)_{t \geq 0}$ . For any hyperedge  $e \in \mathcal{E}$ , the configuration  $X_t(e)$  of  $e$  at time  $t$  satisfies for all  $u \in e$ ,  $X_t(u) = X_{t'}(u)$ , where  $t' = \text{time}_{\text{ud}}(t, u)$ , namely, the value of  $u$  at time  $t$  is the same as the value of  $u$  at time  $t' = \text{time}_{\text{ud}}(t, u)$ . Besides, the configuration of hyperedge  $e$  remains unchanged until some vertex in  $e$  is updated. This motivates the following definition.

► **Definition 21.** *The set  $\mathcal{E}^{\text{ext}}$  of extended hyperedges is defined by  $\mathcal{E}^{\text{ext}} := \cup_{t=0}^T \mathcal{E}_t^{\text{ext}}$ , where*

$$\begin{aligned} \mathcal{E}_0^{\text{ext}} &:= \bigcup_{e \in \mathcal{E}} \{(0, v) \mid v \in e\}, \\ \forall 1 \leq t \leq T, \quad \mathcal{E}_t^{\text{ext}} &:= \bigcup_{e: v_t \in e} \{(t', v) \mid v \in e \wedge t' = \text{time}_{\text{ud}}(t, v)\}. \end{aligned}$$

The extended hypergraph is  $H^{\text{ext}} = (V^{\text{ext}}, \mathcal{E}^{\text{ext}})$ .

At the beginning, each hyperedge  $e \in \mathcal{E}$  takes its initial value, and thus we add all the extended hyperedges with  $t = 0$  to  $\mathcal{E}_0^{\text{ext}}$ . For each update at time  $1 \leq t \leq T$ , only the value of  $v_t$  is updated. Thus the configurations of only the hyperedges containing  $v_t$  are updated, and we add only those to  $\mathcal{E}_t^{\text{ext}}$ .

Corollary 20 shows that for any  $t \geq 1$ , if the coupling in the  $t$ -th transition step fails (i.e.  $X_t(v_t) \neq Y_t(v_t)$ ), then we can find a specific path in the hypergraph  $H$ . Our next lemma lifts such a path to  $H^{\text{ext}}$ . Note that there is a slight difference regarding  $v_t$  comparing to Corollary 20.

► **Lemma 22.** *Let  $1 \leq t \leq T$  be an integer. Suppose  $X_t(v_t) \neq Y_t(v_t)$ . There exist a vertex  $(t', u) \in V^{\text{ext}}$  satisfying  $t' < t$  and  $X_{t'}(u) \neq Y_{t'}(u)$ , together with a path  $e_1^{\text{ext}}, e_2^{\text{ext}}, \dots, e_\ell^{\text{ext}}$  in  $H^{\text{ext}}$  such that*

- $(t, v_t) \in e_1^{\text{ext}}$  and  $(t', u) \in e_\ell^{\text{ext}}$ ;
- for any hyperedge  $e_i^{\text{ext}}$  in the path, there exists  $c \in [s]$  such that for all  $(j, w) \in e_i^{\text{ext}}$ , either  $X_j(w) = Y_j(w) = c$  or  $X_j(w) \neq Y_j(w)$ .

We may repeatedly apply Lemma 22 to trace a discrepancy from some time  $t$  to time 0. By pruning “cycles” (in some hypergraph sense) from such a path from  $t$  to 0 in  $H^{\text{ext}}$ , we can find a path satisfying the properties in the following lemma.

► **Lemma 23.** *Let  $1 \leq t \leq T$  be an integer. Suppose  $X_t(v_t) \neq Y_t(v_t)$ . There exists a path  $e_1^{\text{ext}}, e_2^{\text{ext}}, \dots, e_\ell^{\text{ext}}$  in the extended hypergraph  $H^{\text{ext}}$  such that*

- $(t, v_t) \in e_1^{\text{ext}}$ ,  $\min\{j \mid (j, w) \in e_i^{\text{ext}}\} > 0$  for all  $i < \ell$  and  $\min\{j \mid (j, w) \in e_\ell^{\text{ext}}\} = 0$ ;
- for any  $1 \leq i, i' \leq \ell$  satisfying  $|i - i'| \geq 2$ ,  $e_i^{\text{ext}} \cap e_{i'}^{\text{ext}} = \emptyset$ ;
- for any hyperedge  $e_i^{\text{ext}}$  in the path, there exists  $c \in [s]$  such that for all  $(j, w) \in e_i^{\text{ext}}$ , either  $X_j(w) = Y_j(w) = c$  or  $X_j(w) \neq Y_j(w)$ .

Finally, we give the proof sketch of Lemma 18.

**Proof sketch of Lemma 18.** Let  $t = \text{time}_{\text{ud}}(T, v) \geq \lceil 40n \log \frac{n}{\epsilon} \rceil$ . We only need to bound the probability of  $X_t(v) \neq Y_t(v)$ . Fix a path  $\mathcal{P} = e_1^{\text{ext}}, e_2^{\text{ext}}, \dots, e_\ell^{\text{ext}}$  satisfying the first two properties in Lemma 23. Call  $\mathcal{P}$  bad if  $\mathcal{P}$  satisfies the third property in Lemma 23. We bound the probability of  $\mathcal{P}$  being bad, and then take a union bound over all possible paths.

To bound the probability, we truncate the last extended hyperedge  $e_\ell^{\text{ext}}$  in  $\mathcal{P}$  to obtain a new path  $\mathcal{P}'$  of length  $\ell - 1$ . By using the local lemma, we can show that

$$\Pr[\mathcal{P} \text{ is bad}] \leq \left( \frac{1.16}{\sqrt{q}} \left( 1 + \frac{5}{k} \right) \right)^{N(\mathcal{P}')},$$

where  $N(\mathcal{P}')$  denotes the number of distinct extended vertices in  $\mathcal{P}'$  and the constant 1.16 comes from comparing  $s = \lceil \sqrt{q} \rceil$  and  $\sqrt{q}$ .

In our analysis, we need to use some detailed structure of the extended hypergraph. Each  $e^{\text{ext}} \in \mathcal{E}^{\text{ext}}$  corresponds to a unique hyperedge  $\text{edge}(e^{\text{ext}}) \in \mathcal{E}$  in the input hypergraph, or more formally,  $\text{edge}(e^{\text{ext}}) := \{v \mid (t, v) \in e^{\text{ext}}\}$ . For two adjacent extended hyperedges  $e^{\text{ext}}, f^{\text{ext}} \in \mathcal{E}^{\text{ext}}$ ,  $e^{\text{ext}}$  is an out-neighbor of  $f^{\text{ext}}$  if  $\text{edge}(e^{\text{ext}}) \neq \text{edge}(f^{\text{ext}})$ , and  $e^{\text{ext}}$  is a self-neighbor of  $f^{\text{ext}}$  if  $\text{edge}(e^{\text{ext}}) = \text{edge}(f^{\text{ext}})$ . For each  $e^{\text{ext}}$ , the number of out-neighbours is at most  $\Delta k^2$ , whereas the number of self-neighbours is at most  $2k$ .

Back to the path  $\mathcal{P}'$ . Consider two adjacent extended hyperedges  $e_i^{\text{ext}}$  and  $e_{i+1}^{\text{ext}}$  in  $\mathcal{P}'$ .

- If  $e_i^{\text{ext}}$  is an out-neighbour of  $e_{i+1}^{\text{ext}}$ , then  $e_i^{\text{ext}}$  and  $e_{i+1}^{\text{ext}}$  share at most one extended vertex because the input hypergraph is simple. In this case,  $\mathcal{P}'$  has many distinct extended vertices. Hence, we can control the probability that  $\mathcal{P}$  is bad.
- If  $e_i^{\text{ext}}$  is a self-neighbour of  $e_{i+1}^{\text{ext}}$ , then  $e_i^{\text{ext}}$  and  $e_{i+1}^{\text{ext}}$  may share a lot of extended vertices. In this case we have to choose non-consecutive hyperedges to bound the number of distinct extended vertices. However, thankfully, given  $e_i^{\text{ext}}$ , there are at most  $2k$  choices for  $e_{i+1}^{\text{ext}}$ , and there are not very many paths with many self-neighbours inside.

With these two cases analysed, to apply the union bound and finish the proof, we parametrise the number of self-neighbours in a path and show that the number of all paths does not outweigh the probability of them being bad. ◀

## References

- 1 Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991.
- 2 József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Struct. Algorithms*, 2(4):343–366, 1991.
- 3 Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Daniel Štefankovič. Approximation via correlation decay when strong spatial mixing fails. *SIAM J. Comput.*, 48(2):279–349, 2019.
- 4 Christian Borgs, Jennifer Chayes, Jeff Kahn, and László Lovász. Left and right convergence of graphs with bounded degree. *Random Struct. Algorithms*, 42(1):1–28, 2013.
- 5 Artur Czumaj and Christian Scheideler. Coloring nonuniform hypergraphs: A new algorithmic approach to the general Lovász local lemma. *Random Struct. Algorithms*, 17(3-4):213–237, 2000.
- 6 P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday)*, Vol. II, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, 1975.
- 7 Weiming Feng, Heng Guo, and Jiaheng Wang. Improved bounds for randomly colouring simple hypergraphs. *CoRR*, abs/2202.05554, 2022. [arXiv:2202.05554](#).
- 8 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting  $k$ -SAT solutions in the local lemma regime. *J. ACM*, 68(6):40:1–40:42, 2021.
- 9 Weiming Feng, Kun He, and Yitong Yin. Sampling constraint satisfaction solutions in the local lemma regime. *arXiv*, abs/2011.03915, 2020. [arXiv:2011.03915](#).
- 10 Alan Frieze and Dhruv Mubayi. Coloring simple hypergraphs. *J. Combin. Theory Ser. B*, 103(6):767–794, 2013.
- 11 Alan M. Frieze and Michael Anastos. Randomly coloring simple hypergraphs with fewer colors. *Inf. Process. Lett.*, 126:39–42, 2017. doi:10.1016/j.ipl.2017.06.005.
- 12 Alan M. Frieze and Páll Melsted. Randomly coloring simple hypergraphs. *Inf. Process. Lett.*, 111(17):848–853, 2011. doi:10.1016/j.ipl.2011.06.001.
- 13 Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Kuan Yang. Counting solutions to random CNF formulas. *SIAM J. Comput.*, 50(6):1701–1738, 2021.
- 14 Andreas Galanis, Heng Guo, and Jiaheng Wang. Inapproximability of counting hypergraph colourings. *arXiv preprint*, 2021. [arXiv:2107.05486](#).
- 15 Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is asymptotically tight for SAT. *J. ACM*, 63(5):43:1–43:32, 2016.
- 16 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):18:1–18:31, 2019.
- 17 Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Counting hypergraph colorings in the local lemma regime. *SIAM J. Comput.*, 48(4):1397–1424, 2019.
- 18 Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):28, 2011.
- 19 Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) Lovász local lemma. *arXiv*, abs/2107.03932, 2021. [arXiv:2107.03932](#).
- 20 Jonathan Hermon, Allan Sly, and Yumeng Zhang. Rapid mixing of hypergraph independent sets. *Random Struct. Algorithms*, 54(4):730–767, 2019.
- 21 Mark Huber. Exact sampling and approximate counting techniques. In *STOC*, pages 31–40. ACM, 1998.
- 22 Mark Huber. Approximation algorithms for the normalizing constant of Gibbs distributions. *Ann. Appl. Probab.*, 25(2):974–985, 2015.
- 23 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Towards the sampling Lovász local lemma. *arXiv*, abs/2011.12196, 2020. [arXiv:2011.12196](#).



- 24 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. On the sampling Lovász local lemma for atomic constraint satisfaction problems. *arXiv*, abs/2102.08342, 2021. [arXiv:2102.08342](#).
- 25 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43:169–188, 1986.
- 26 Vladimir Kolmogorov. A faster approximation algorithm for the Gibbs partition function. In *COLT*, pages 228–249. PMLR, 2018. URL: <http://proceedings.mlr.press/v75/kolmogorov18a.html>.
- 27 Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):10:1–10:25, 2019.
- 28 Michael Molloy and Bruce A. Reed. Further algorithmic aspects of the local lemma. In *STOC*, pages 524–529. ACM, 1998.
- 29 Robin A. Moser. A constructive proof of the Lovász local lemma. In *STOC*, pages 343–350. ACM, 2009.
- 30 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11, 2010.
- 31 James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms*, 9(1-2):223–252, 1996.
- 32 Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In *SODA*, pages 611–620. SIAM, 2008.
- 33 Richard P. Stanley. *Enumerative combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1999.
- 34 Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *J. ACM*, 56(3):18, 2009.
- 35 Ian M Wanless and David R Wood. A general framework for hypergraph colouring. *arXiv preprint*, 2020. [arXiv:2008.00775](#).



# Lifting with Inner Functions of Polynomial Discrepancy

Yahel Manor

Department of Computer Science, University of Haifa, Israel

Or Meir   

Department of Computer Science, University of Haifa, Israel

---

## Abstract

Lifting theorems are theorems that bound the communication complexity of a composed function  $f \circ g^n$  in terms of the query complexity of  $f$  and the communication complexity of  $g$ . Such theorems constitute a powerful generalization of direct-sum theorems for  $g$ , and have seen numerous applications in recent years.

We prove a new lifting theorem that works for every two functions  $f, g$  such that the discrepancy of  $g$  is at most inverse polynomial in the input length of  $f$ . Our result is a significant generalization of the known direct-sum theorem for discrepancy, and extends the range of inner functions  $g$  for which lifting theorems hold.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity; Theory of computation  $\rightarrow$  Oracles and decision trees

**Keywords and phrases** Lifting, communication complexity, query complexity, discrepancy

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.26

**Category** RANDOM

**Funding** *Yahel Manor*: Supported by the Israel Science Foundation (grant No. 716/20).

*Or Meir*: Partially supported by the Israel Science Foundation (grant No. 716/20).

## 1 Introduction

The direct-sum question is a fundamental question in complexity theory, which asks whether computing a function  $g$  on  $n$  independent inputs is  $n$  times harder than computing it on a single input. A related type of result, which is sometimes referred to as an “XOR lemma”, says that computing the XOR of the outputs of  $g$  on  $n$  independent inputs is about  $n$  times harder than computing  $g$  on a single coordinate. Both questions received much attention in the communication complexity literature, see, e.g., [24, 13, 23, 7, 31, 21, 3, 22, 25, 2, 20, 33, 5, 4].

A lifting theorem is a powerful generalization of both direct-sum theorems and XOR lemmas. Let  $f: \{0, 1\}^n \rightarrow \mathcal{O}$  and  $g: \{0, 1\}^b \times \{0, 1\}^b \rightarrow \{0, 1\}$  be functions (where  $\mathcal{O}$  is some arbitrary set). The block-composed function  $f \circ g^n$  is the function that corresponds to the following task: Alice gets  $x_1, \dots, x_n \in \{0, 1\}^b$ , Bob gets  $y_1, \dots, y_n \in \{0, 1\}^b$ , and they wish to compute the output of  $f$  on the  $n$ -bit string whose  $i$ -th bit is  $g(x_i, y_i)$ . Lifting theorems say that the “natural way” for computing  $f \circ g^n$  is more-or-less the best way. In particular, direct-sum theorems and XOR lemmas can be viewed as lifting theorems for the special cases where  $f$  is the identity function and the parity function respectively.

A bit more formally, observe that there is an obvious protocol for computing  $f \circ g^n$ : Alice and Bob jointly simulate a decision tree of optimal height for solving  $f$ . Any time the tree queries the  $i$ -th bit, they compute  $g$  on  $(x_i, y_i)$  by invoking the best possible communication protocol for  $g$ . A (query-to-communication) *lifting theorem* is a theorem that says that this



© Yahel Manor and Or Meir;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 26; pp. 26:1–26:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

protocol is roughly optimal. Specifically, let  $D^{\text{dt}}(f)$  and  $D^{\text{cc}}(g)$  denote the deterministic query complexity of  $f$  and communication complexity of  $g$  respectively, and let  $R^{\text{dt}}(f)$  and  $R^{\text{cc}}(g)$  denote the corresponding randomized complexities. Then, a lifting theorem says that

$$\begin{aligned} D^{\text{cc}}(f \circ g^n) &= \Omega(D^{\text{dt}}(f) \cdot D^{\text{cc}}(g)) && \text{(in the deterministic setting)} \\ R^{\text{cc}}(f \circ g^n) &= \Omega(R^{\text{dt}}(f) \cdot R^{\text{cc}}(g)) && \text{(in the randomized setting).} \end{aligned} \quad (1)$$

In other words, a lifting theorem says that the communication complexity of  $f \circ g^n$  is close to the upper bound that is obtained by the natural protocol.

In recent years, lifting theorems found numerous applications, such as proving lower bounds on monotone circuit complexity and proof complexity (e.g. [28, 16, 30, 26, 14, 27, 11, 12]), the separation of partition number and deterministic communication complexity [17], proving lower bounds on data structures [10], and an application to the famous log-rank conjecture [19], to name a few.

For most applications, it is sufficient to prove a lifting theorem that holds for every outer function  $f$ , but only for one particular choice of the inner function  $g$ . Moreover, it is desirable that the inner function  $g$  would be as simple as possible, and that its input length  $b$  would be as small as possible in terms of the input length  $n$  of the outer function  $f$ . For these reasons, the function  $g$  is often referred to as the “gadget”.

On the other hand, if we view lifting theorems as a generalization of direct-sum theorems, then it is an important research goal to prove lifting theorems for as many inner functions  $g$  as possible, including “complicated” ones. This goal is not only interesting in its own right, but might also lead to additional applications. Indeed, this goal is a natural extension of the long line of research that attempts to prove direct-sum theorems for as many functions as possible. This is the perspective we take in this work, following Chattopadhyay et. al. [9, 8]. In particular, we intentionally avoid the term “gadget”, since we now view the function  $g$  as the main object of study.

## Previous work

The first lifting theorem, due to Raz and McKenzie [29], holds only when the inner function  $g$  is the index function. For a long time, this was the only inner function for which lifting theorems were known to hold for every outer function  $f$ . Then, the works of Chattopadhyay et. al. [9] and Wu et. al. [35] proved a lifting theorem for the case where  $g$  is the inner product function. The work of [9] went further than that, and showed that their lifting theorem holds for any inner function  $g$  that satisfies a certain hitting property. This includes, for example, the gap-Hamming-distance problem.

All the above results are stated only for the deterministic setting. In the randomized setting, Göös, Pitassi, and Watson [18] proved a lifting theorem with the inner function  $g$  being the index function. In addition, Göös et. al. [15] proved a lifting theorem in the non-deterministic setting (as well as several related settings) with  $g$  being the inner product function.

More recently, Chattopadhyay et. al. [8] proved a lifting theorem that holds for every inner function  $g$  that has logarithmic input length and exponentially small discrepancy. This theorem holds in both the deterministic and randomized setting, and includes the cases where  $g$  is the inner product function or a random function. Since our work builds on the lifting theorem of [8], we discuss this result in more detail. The *discrepancy* of  $g$ , denoted  $\text{disc}(g)$ , is a natural and widely-studied property of functions, and is equal to the maximum bias of  $g$  in any combinatorial rectangle. Formally, it is defined as follows:

► **Definition 1.** Let  $g : \{0, 1\}^b \times \{0, 1\}^b \rightarrow \{0, 1\}$  be a function, and let  $U, V$  be independent random variables that are uniformly distributed over  $\{0, 1\}^b$ . Given a combinatorial rectangle  $R \subseteq \{0, 1\}^b \times \{0, 1\}^b$ , the discrepancy of  $g$  with respect to  $R$ , denoted  $\text{disc}_R(g)$ , is defined as follows:

$$\text{disc}_R(g) = |\Pr[g(U, V) = 0 \text{ and } (U, V) \in R] - \Pr[g(U, V) = 1 \text{ and } (U, V) \in R]|.$$

The discrepancy of  $g$ , denoted  $\text{disc}(g)$ , is defined as the maximum of  $\text{disc}_R(g)$  over all combinatorial rectangles  $R \subseteq \{0, 1\}^b \times \{0, 1\}^b$ .

Informally, the main theorem of [8] says that if  $\text{disc}(g) = 2^{-\Omega(b)}$  and  $b \geq c \cdot \log n$  for some constant  $c$ , then

$$D^{\text{cc}}(f \circ g^n) = \Omega(D^{\text{dt}}(f) \cdot b) \quad \text{and} \quad R_{1/3}^{\text{cc}}(f \circ g^n) = \Omega(R_{1/3}^{\text{dt}}(f) \cdot b).$$

We note that when  $\text{disc}(g) = 2^{-\Omega(b)}$ , it holds that  $D^{\text{cc}}(g) \geq R^{\text{cc}}(g) \geq \Omega(b)$ , and therefore the latter result is equivalent to Equation (1).

### The research agenda of [8]

As discussed above, we would like to prove a lifting theorem that holds for as many inner functions  $g$  as possible. Inspired by the literature on direct-sum theorems, [8] conjectured that lifting theorems should hold for every inner function  $g$  that has a sufficiently large information cost  $\text{IC}(g)$ .

► **Conjecture 2** (special case of [8, Conj. 1.4]). *There exists a constant  $c > 0$  such that the following holds. Let  $f : \{0, 1\}^n \rightarrow \mathcal{O}$  and  $g : \{0, 1\}^b \times \{0, 1\}^b \rightarrow \{0, 1\}$  be an arbitrary function such that  $\text{IC}(g) \geq c \cdot \log n$ . Then*

$$R^{\text{cc}}(f \circ g^n) = \Omega(R^{\text{dt}}(f) \cdot \text{IC}(g)).$$

Proving this conjecture is a fairly ambitious goal. As an intermediate goal, [8] suggested to prove this conjecture for complexity measures that are simpler than  $\text{IC}(g)$ . In light of their result, it is natural to start with discrepancy. It has long been known that the quantity  $\Delta(g) \stackrel{\text{def}}{=} \log \frac{1}{\text{disc}(g)}$  is a lower bound on  $R^{\text{cc}}(g)$  up to a constant factor. More recently, it has even been shown that  $\Delta(g)$  is a lower bound on  $\text{IC}(g)$  up to a constant factor [6]. Motivated by this consideration, [8] suggested the following natural conjecture: for every function  $g$  such that  $\Delta(g) \geq c \cdot \log n$ , it holds that  $R^{\text{cc}}(f \circ g^n) = \Omega(R^{\text{dt}}(f) \cdot \Delta(g))$  (see Conjecture 1.5 there). The lifting theorem of [8] proves this conjecture for the special case where  $\Delta(g) = \Omega(b)$ .

### Our result

In this work, we prove the latter conjecture of [8] in full, by waiving the limitation of  $\Delta(g) = \Omega(b)$  from their result. We note that a full proof can be found in the full version that will be published later. As in previous works, our result holds even if  $f$  is replaced with a general search problem  $\mathcal{S}$ . In what follows, we denote by  $R_\beta^{\text{dt}}(\mathcal{S})$  and  $R_\beta^{\text{cc}}(\mathcal{S} \circ g^n)$  the randomized query complexity of  $\mathcal{S}$  with error  $\beta$  and the randomized communication complexity of  $\mathcal{S} \circ g^n$  with error  $\beta$  respectively. We now state our result formally.

► **Theorem 3** (Main theorem). *There exists a universal constant  $c$  such that the following holds: Let  $\mathcal{S}$  be a search problem that takes inputs from  $\{0, 1\}^n$ , and let  $g : \{0, 1\}^b \times \{0, 1\}^b \rightarrow \{0, 1\}$  be an arbitrary function such that  $\Delta(g) \geq c \cdot \log n$ . Then*

$$D^{\text{cc}}(\mathcal{S} \circ g^n) = \Omega(D^{\text{dt}}(\mathcal{S}) \cdot \Delta(g)),$$

and for every  $\beta > 0$  it holds that

$$R_{\beta}^{cc}(\mathcal{S} \circ g^n) = \Omega\left((R_{\beta'}^{dt}(\mathcal{S}) - O(1)) \cdot \Delta(g)\right),$$

where  $\beta' = \beta + 2^{-\Delta(g)/50}$ .

► **Remark 4.** It is interesting to note that one of the first direct-sum results in the randomized setting went along these lines. In particular, the work of Shaltiel [31] implies that for every function  $g$  such that  $\Delta(g) \geq c$  for some universal constant  $c$ , it holds that  $R^{cc}(g^n) = \Omega(n \cdot \Delta(g))$ . Our main theorem can be viewed as a generalization of that result.

► **Remark 5.** A natural question is whether the requirement that  $\Delta(g) \geq c \cdot \log n$  is necessary. In principle, it is possible that this requirement could be relaxed. Any such relaxation, however, would imply a lifting theorem that allows gadgets of smaller input length than is currently known which would be considered a significant breakthrough.

► **Remark 6.** In order to facilitate the presentation, we restricted our discussion on the previous work to lifting theorems that hold for every outer function  $f$  (and indeed, every search problem  $\mathcal{S}$ ). If one is willing to make certain assumptions on the outer function  $f$ , it is possible to prove stronger lifting theorems that in particular allow for a wider variety of inner functions (see, e.g., [32, 34, 16, 19, 12, 1]).

► **Remark 7.** We note that Definition 1 is in fact a special case of the common definition of discrepancy. The general definition refers to an arbitrary distribution  $\mu$  over  $\{0, 1\}^b \times \{0, 1\}^b$ . The *discrepancy of  $g$  over  $\mu$*  is defined similarly to Definition 1 except that the random variables  $U, V$  are distributed according to  $\mu$  rather than the uniform distribution.

## 1.1 Our Techniques

Following the previous works, we use a “simulation argument”: We show that given a protocol that computes  $f \circ g^n$  with communication complexity  $C$ , we can construct a decision tree that computes  $f$  with query complexity  $O(\frac{C}{\Delta(g)})$ . In particular, we follow the simulation argument of [8] and extend their main technical lemma. We now describe this argument in more detail, focusing on the main lemma of [8] and our extension of that lemma. For simplicity, we focus on the deterministic setting, but the proof in the randomized setting follows similar ideas. In this paper, due to space constraints, the simulation argument is omitted. Only the proof of the main lemma is presented in the paper.

### The simulation argument

We assume that we have a protocol  $\Pi$  that computes  $f \circ g^n$ , and would like to construct a decision tree  $T$  that computes  $f$ . The basic idea is that given an input  $z \in \{0, 1\}^n$ , the tree  $T$  uses the protocol  $\Pi$  to find a pair of inputs  $(x, y) \in (\{0, 1\}^b)^n \times (\{0, 1\}^b)^n$  such that  $(f \circ g^n)(x, y) = f(z)$ , and then returns the output of  $\Pi$  on  $(x, y)$ .

In order to find the pair  $(x, y)$ , the tree  $T$  maintains a pair of random variables  $(X, Y)$ . Initially, the variables  $(X, Y)$  are uniformly distributed over  $(\{0, 1\}^b)^n \times (\{0, 1\}^b)^n$ . Then, the tree gradually changes the distribution of  $(X, Y)$  until they satisfy  $(f \circ g^n)(X, Y) = f(z)$  with probability 1, at which point the tree chooses  $(x, y)$  to be an arbitrary pair in the support of  $(X, Y)$ . This manipulation of the distribution of  $(X, Y)$  is guided by a simulation of the protocol  $\Pi$  on  $(X, Y)$  (hence the name “simulation argument”). Throughout this process, the decision tree maintains the following structure of  $(X, Y)$ :

- There is a set of coordinates, denoted  $F \subseteq [n]$ , such that for every  $i \in F$  it holds that  $g(X_i, Y_i) = z_i$  with probability 1.
- $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  are *dense* in the following sense: for every  $J \subseteq [n] \setminus F$ , the variables  $X_J$  and  $Y_J$  have high min-entropy.

Intuitively, the set  $F$  is the set of coordinates  $i$  for which the simulation of  $\Pi$  has already computed  $g(X_i, Y_i)$ , while for the coordinates  $i \in [n] \setminus F$  the value  $g(X_i, Y_i)$  is unknown. Initially, the set  $F$  is empty, and then it is gradually expanded until it holds that  $(f \circ g^n)(X, Y) = f(z)$ .

### The main lemma of [8]

Suppose now that as part of the process described above, we would like to expand the set  $F$  by adding a new set of coordinates  $I \subseteq [n] \setminus F$ . This means that we should condition the distribution of  $(X, Y)$  on the event that  $g^I(X_I, Y_I) = z_I$ . This conditioning, however, decreases the min-entropy of  $(X, Y)$ , which might cause  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  to lose their density.

In order to resolve this issue, [8] defined a notion of “sparsifying values” of  $X$  and  $Y$ . Informally, a value  $x$  in the support of  $X$  is called *sparsifying* if after conditioning  $Y$  on the event  $g^I(x_I, Y_I) = z_I$ , the variable  $Y_{[n] \setminus (F \cup I)}$  ceases to be dense. A sparsifying value of  $Y$  is defined similarly. It is not hard to see that if  $X$  and  $Y$  do not have any sparsifying values in their supports, then the density of  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  is maintained after the conditioning on  $g^I(X_I, Y_I) = z_I$ . Therefore, [8] design their decision tree such that before every conditioning on the event  $g^I(x_I, Y_I) = z_I$ , the tree first removes the sparsifying values from the supports of  $X$  and  $Y$ .

The removal of sparsifying values, however, raises another issue: when we remove values from the supports of  $X$  and  $Y$ , we decrease the min-entropy of  $X$  and  $Y$ . In particular, the removal of the sparsifying values might cause  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  to lose their density. This issue is resolved by the main technical lemma of [8]. Informally, this lemma says that if  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  are dense, then the sparsifying values are very rare. This means that the removal of these values barely changes the min-entropy of  $X$  and  $Y$ , and in particular, does not violate the density property.

### Our contribution

Recall that the lifting theorem of [8] requires that  $\Delta(g) = \Omega(b)$ , and that our goal is to waive that requirement. Unfortunately, it turns out that main lemma of [8] fails when  $\Delta(g)$  is very small relatively to  $b$ . In fact, the full version provide an example in which *all* the values in the support of  $X$  are sparsifying. In such a case, it is simply impossible to remove the sparsifying values.

In short, unlike [8], we cannot afford to remove the sparsifying values before conditioning on the event  $g^I(X_I, Y_I) = z_I$ . Therefore, in our simulation  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  sometimes lose their density after the conditioning. Nevertheless, we observe that even if the density property breaks in this way, it can often be restored by removing some more values from the supports of  $X$  and  $Y$ . We formalize this intuition by defining a notion of “recoverable values”. Informally, a value  $x$  in the support of  $X$  is called *recoverable* if after conditioning  $Y$  on the event  $g^I(x_I, Y_I) = z_I$ , the density of  $Y_{[n] \setminus (F \cup I)}$  can be restored by discarding some values from its support.

Our main lemma says, informally, that if  $X_{[n] \setminus F}$  and  $Y_{[n] \setminus F}$  are dense, then almost all the values of  $X$  and  $Y$  are recoverable. In particular, we can afford to remove the unrecoverable values of  $X$  and  $Y$  without violating their density. Given our lemma, it is easy to fix



the simulation argument of [8]: whenever our decision tree is about to condition on an event  $g^I(x_I, Y_I) = z_I$ , it first discards the unrecoverable values of  $X$  and  $Y$ ; then, after the conditioning, the decision tree restores the density property by discarding some additional values. The rest of our argument proceeds exactly as in [8].

### The proof of our main lemma

The definition of a sparsifying value of  $X$  can be stated as follows: the value  $x$  is sparsifying if there exists a value  $y_J$  such that the probability

$$\Pr[Y_J = y_J \mid g(x_I, Y_I) = z_I] \quad (2)$$

is too high. On the other hand, it can be showed that a value  $x$  is *unrecoverable* if there are *many* such corresponding values  $y_J$ . Indeed, if there are only few such values  $y_J$ , then we can recover the density of  $Y_{[n] \setminus (F \cup I)}$  by discarding them.

Very roughly, the main lemma of [8] is proved by showing that for every  $y_J$ , there is only a very small number of corresponding  $x$ 's for which the latter probability is too high. Then, by taking union bound over all possible choices of  $y_J$ , it follows that there are only few values  $x$  for which there exists some corresponding  $y_J$ . In other words, there are only few sparsifying values.

This argument works in the setting of [8] because they can prove a very strong upper bound on the number of values  $x$  for a single  $y_J$  — indeed, the bound is sufficiently strong to survive the union bound. In our setting, on the other hand, the fact that we assume a smaller value of  $\Delta(g)$  translates to a weaker bound on the number of values  $x$  for a single  $y_J$ . In particular, we cannot afford to use the union bound. Instead, we take a different approach: we observe that, since for every  $y_J$  there is only a small number of corresponding  $x$ 's, it follows by an averaging argument that there can only be a small number of  $x$ 's that have *many* corresponding  $y_J$ 's. In other words, it follows from the averaging argument that there can only be a small number of *unrecoverable*  $x$ 's.

Implementing this idea is more difficult than it might seem at a first glance. The key difficulty is that when we say “values  $x$  that have many corresponding  $y_J$ 's” we do not refer to the absolute number of  $y_J$ 's but rather to their probability mass. Specifically, the probability distribution according to which the  $y_J$ 's should be counted is the probability distribution of Equation (2). Unfortunately, this means that for every value  $x$ , we count the  $y_J$ 's according to a different distribution, which renders a simple averaging argument impossible. We overcome this difficulty by proving a finer upper bound on the number of  $x$ 's for each  $y_J$  and using a careful bucketing scheme for the averaging argument.

## 2 The Main Lemma

In this section, we state and prove our main lemma. As discussed in the introduction, our simulation argument maintains a pair of random variables  $X, Y \in \Lambda^n$ . A crucial part of the simulation consists of removing certain “dangerous” values from the supports of these variables. Our main lemma says that almost all values are safe.

There are two types of “dangerous” values: non recoverable values are values that might lead to a violation of the structure of  $X, Y$  (as per density, defined in [8]); non almost uniform values are values for which  $g^I(x_I, Y_I)$  is not close enough to uniform and therefore might cause the simulation to leak too much information about  $X$  and  $Y$ . Additionally, the assumption that  $g^I(x_I, Y_I)$  is close to uniform allow the simulation to assume that  $X \mid g^I(X_I, Y_I) = z_I$  is close to  $X$  even when  $X, Y$  are not structured. We first define those notions formally and then compare between those notions and the notions from [8].

► **Definition 8** (dangerous values). Let  $\varepsilon, \alpha \geq 0$ . Let  $Y$  be a random variable and  $\rho$  a restriction. Let  $x \in \Lambda^n$ , and let  $\sigma_Y > 0$  be such that  $Y_{\text{free}(\rho)}$  is  $\sigma_Y$ -sparse. We say that  $x$  is almost uniform if for any set  $I \subseteq \text{free}(\rho)$  and an assignment  $z_I \in \{0, 1\}^I$  it holds that

$$\Pr [g^I(x_I, Y_I) = z_I] \in 2^{-|I|} \left(1 \pm 2^{-\frac{\alpha}{10}}\right).$$

We say that  $x$  is  $(\varepsilon, \alpha)$ -recoverable if for all  $I \subseteq \text{free}(\rho)$  and  $z_I$  the following holds: exist event  $\mathcal{E}$  such that  $\Pr [\mathcal{E} \mid g^I(x_I, Y_I) = z_I] \geq 1 - 2^{-\alpha\Delta}$  and the random variable

$$Y_{\text{free}(\rho)-I} \mid \mathcal{E} \text{ and } g^I(x_I, Y_I) = z_I$$

is  $(\sigma_Y + \varepsilon)$ -sparse. We say that  $x$  is  $(\varepsilon, \alpha)$ -safe if it is both almost uniform and  $(\varepsilon, \alpha)$ -recoverable. Almost uniform, recoverable, and safe values of  $Y$  are defined analogously.

The notion of “dangerous” (Alternatively, not safe) in this paper is closely connected to the definition presented in [8]. We will now discuss the differences and the reasons for the changes. The first type of “dangerous” values, that is non recoverable values, are connected to notion of sparsifying from [8]. Any non recoverable value is sparsifying, but the converse is false. Both definitions regard the sparsity of the random variable  $Y_{[n]-I} \mid g^I(x_I, Y_I) = z_I$ , if this variable is not dense then it sparsifying. We suggest to “recover”  $Y_{[n]-I} \mid g^I(x_I, Y_I) = z_I$  by conditioning it on high-probability event that make this random variable sparse enough. If such option is viable we say that  $x$  is recoverable. As show in the full version, using the original definition of sparsifying in the setting of  $\Delta \ll b$  can lead to the marking all values  $x$  as dangerous, and therefore the weakening is required. Regarding the second type of “dangerous” values, the definition of almost uniform is strictly stronger than the definition of non leaking. Both definition regard the values of  $\Pr [g^I(x_I, Y_I) = z_I]$ , almost uniform bound the value tightly both from above and bellow while non leaking bound only from bellow. The definition of almost uniform allow us to get tight connection between  $\Pr [\mathcal{E}]$  and  $\Pr [\mathcal{E} \mid g^I(x_I, Y_I) = z_I]$  as can be seen in proof of correctness of the randomized theorem in the full version, where leaking is not sufficient for the analysis of the recovering process.

We turn to state our main lemma.

► **Proposition 9** (Main Lemma). Let  $\varepsilon \geq \frac{5}{c}$ ,  $\alpha > \frac{1}{c}$ ,  $\gamma > 0$ , and let  $X$  and  $Y$  be independent  $(\rho, \tau)$ -structured random variables. Let  $\sigma_X, \sigma_Y > 0$  be such that  $X_{\text{free}(\rho)}$  is  $\sigma_X$ -sparse, and  $Y_{\text{free}(\rho)}$  is  $\sigma_Y$ -sparse. If  $\sigma_X + 2\sigma_Y \leq \frac{9}{10} - \frac{22}{c} - \gamma - \alpha$ . Then

$$\Pr_{x \sim X} [x \text{ is not } (\varepsilon, \alpha)\text{-safe}] \leq 2^{-\gamma\Delta}.$$

In the rest of this section, we prove the main lemma. Let  $\varepsilon, \alpha, \sigma_X, \sigma_Y$  be as in the lemma. Additionally, we let  $X, Y$  to be independent  $(\rho, \tau)$ -structured random variables such that  $X_{\text{free}(\rho)}$  is  $\sigma_X$ -sparse, and  $Y_{\text{free}(\rho)}$  is  $\sigma_Y$ -sparse and let  $\tau \stackrel{\text{def}}{=} \sigma_X + \sigma_Y$ . We note that we do not assume that  $\sigma_X + 2\sigma_Y \leq \frac{9}{10} - \frac{22}{c} - \gamma - \alpha$  in the following lemmas, and some other requirements are used instead. For simplicity, we assume that  $\text{fix}(\rho) = \emptyset$  and  $\text{free}(\rho) = [n]$  (otherwise, we can restrict our attention to the coordinates in  $\text{free}(\rho)$ ). The first step of the proof is to upper bound the probability that  $X$  takes a non almost uniform value.

► **Proposition 10.** Let  $\gamma > 0$ . Additionally assume that  $\tau \leq \frac{9}{10} - \frac{11}{c} - \gamma$ . The probability that  $X$  takes a value that is not almost uniform is at most  $2^{-\gamma\Delta}$ .

Proposition 10 is proved in Section 2.1 below. We now introduce the definition sparsifying, informally, a value  $x$  is  $(\varepsilon, t)$ -sparsifying with respect to  $y_J$  if in the distribution  $Y_J \mid g^I(x_I, Y_I) = z_I$  the value  $y_J$  violets the  $(\sigma_Y + \varepsilon)$ -sparsity of  $Y$ . While all sparsifying values

violate the  $(\sigma_Y + \varepsilon)$ -sparsity of  $Y$ , some of them violate it more strongly than others, and this is measured by the additional parameter  $t$ . The next steps of the proof use the following notion.

► **Definition 11.** Let  $x \in \Lambda^n$ ,  $J \subseteq [n]$ , and  $y_J \in \Lambda^J$ . We say that  $x$  is  $(\varepsilon, t)$ -sparsifying for  $y_J$  if there exist  $I \subseteq [n]$  and  $z_I \in \Lambda^I$  such that

$$\Pr [Y_J = y_J \mid g^I(x_I, Y_I) = z_I] > 2^{(\sigma_Y + \varepsilon) \cdot \Delta \cdot |J| + t - b \cdot |J|}.$$

Informally, a value  $x$  is not recoverable if it is sparsifying for many  $y_J$ 's, whereas a value  $x$  is sparsifying according to the terminology of [8] if it is sparsifying for some  $y_J$ .

The following proposition upper bounds the probability that  $X$  takes a sparsifying value for specific value  $y_J$ , and is proved in Section 2.2.

► **Proposition 12.** Let  $\gamma > \frac{2}{c}$ . Additionally assume that  $\varepsilon \geq \frac{5}{c}$  and  $\tau \leq 1 - \frac{14}{c} - \gamma$ . Then, for every  $J \subseteq [n]$  and for every  $y_J \in \Lambda^J$ , the probability that  $X$  takes an almost uniform value  $x$  that is  $(\varepsilon, t)$ -sparsifying for  $y_J$  is at most  $2^{-\gamma \cdot \Delta \cdot \frac{c-\varepsilon}{2} \cdot |J| - 2t}$ .

Proposition 12 is essentially a more refined version of the analysis in [8]. An important point about this proposition is that it gives a stronger bound for larger values of  $t$ . In contrast, the analysis [8] does not consider the parameter  $t$  and gives the same upper bound for all values  $x$ . In the final part of the proof, which is described in Section 2.3, we derive the main lemma from Propositions 10 and 12.

To prove the propositions in this section we will use the following lemma from [8]

► **Lemma 13** (see, e.g., [8, Cor. 2.13]). Let  $\gamma, \lambda > 0$  and let  $S \subseteq [n]$ . If it holds for  $X, Y$  that

$$D_\infty(X_S) + D_\infty(Y_S) \leq (\Delta(g) - 7 - \gamma - \lambda) \cdot |S|.$$

Then the probability that  $X$  takes a value  $x \in \Lambda^n$  such that

$$\text{bias}(g^{\oplus S}(x_S, Y_S)) > 2^{-\lambda|S|}$$

is less than  $2^{-\gamma|S|}$ .

## 2.1 Proof of Proposition 10

In this section we prove Proposition 10, following the ideas of [8]. Essentially, the proof uses the fact that  $X$  and  $Y$  have low sparsity together with the discrepancy of  $g$  to argue that with high probability the random variable  $X_S$  takes a value  $x_S$  such that all parities  $g^{\oplus S}(x_S, Y_S)$  are relatively unbiased. Then, the proof uses the latter claim together with the Vazirani lemma to conclude that the random strings  $g^I(x_I, Y_I)$  are almost uniform.

► **Proposition 10.** Let  $\gamma > 0$ . Additionally assume that  $\tau \leq \frac{9}{10} - \frac{11}{c} - \gamma$ . The probability that  $X$  takes a value that is not almost uniform is at most  $2^{-\gamma \cdot \Delta}$ .

**Proof.** We start by observing that for every  $x \in \Lambda^n$ , if it holds that  $\text{bias}(g^{\oplus S}(x_S, Y_S)) \leq 2^{-\frac{\Delta}{10}} \cdot (2n)^{-|S|}$  for every non-empty set  $S \subseteq [n]$ , then  $x$  is almost uniform. Indeed, let  $x \in \Lambda^n$  be a value that satisfies the above condition, and let  $I \subseteq [n]$ . Then, by applying the first variant of Vazirani's lemma to the random variable  $g^I(x_I, Y_I)$ , it holds that

$$\Pr [g^I(x_I, Y_I) = z_I] \in \left(1 \pm 2^{-\frac{\Delta}{10}}\right) \cdot 2^{-|I|}$$

for every  $z_I \in \{0, 1\}^I$ . It follows that  $x$  is almost uniform.

It remains to show that with probability at least  $1 - 2^{-\gamma \cdot \Delta}$  the random variable  $X$  takes a value  $x$  that satisfies the latter condition on the biases. We start by lower bounding the probability that  $\text{bias}(g^{\oplus S}(x_S, Y_S)) \leq 2^{-\frac{\Delta}{10}} \cdot (2n)^{-|S|}$  for a specific set  $S \subseteq [n]$ . Fix a non-empty set  $S \subseteq [n]$ . By assumption, it holds that

$$\begin{aligned} D_\infty(X_S) + D_\infty(Y_S) &\leq \left(1 - \frac{11}{c} - \gamma - \frac{1}{10}\right) \cdot \Delta \cdot |S| \\ &= \left(\Delta - \frac{7\Delta}{c} - \gamma\Delta - \frac{\Delta}{10} - \frac{4\Delta}{c}\right) \cdot |S| \\ &\leq \left(\Delta - 7 - \gamma\Delta - \frac{\Delta}{10} - 2\log n - 2\right) \cdot |S|. \end{aligned}$$

By applying Lemma 13 with  $\gamma = \gamma\Delta + \log n + 1$  and  $\lambda = \log n + 1 + \frac{\Delta}{10}$  it follows that with probability at least  $1 - 2^{-\gamma\Delta-1} \cdot \frac{1}{n^{|S|}}$ , the random variable  $X$  takes a value  $x$  such that

$$\text{bias}(g^{\oplus S}(x_S, Y_S)) \leq (2^{-\frac{\Delta}{10}} \cdot 2n)^{-|S|} \leq 2^{-\frac{\Delta}{10}} \cdot (2n)^{-|S|}.$$

Next, by taking the union bound over all non-empty sets  $S \subseteq [n]$ , it follows that the probability that there exists some non-empty set  $S$  with  $\text{bias}(g^{\oplus S}(x_S, Y_S)) > 2^{-\frac{\Delta}{10}} \cdot (2n)^{-|S|}$  is at most

$$\begin{aligned} &\sum_{S \subseteq [n]: S \neq \emptyset} 2^{-\gamma\Delta-1} \cdot \frac{1}{n^{|S|}} && \text{(binomial like bound)} \\ &< 2^{-\gamma\Delta-1} \cdot 2 \\ &= 2^{-\gamma\Delta}. \end{aligned}$$

It follows that with probability at least  $1 - 2^{-\gamma\Delta}$ , the random variable  $X$  takes a value  $x$  such that  $\text{bias}(g^{\oplus S}(x_S, Y_S)) \leq 2^{-\frac{\Delta}{10}} \cdot (2n)^{-|S|}$  for all non-empty sets  $S \subseteq [n]$ , as required.  $\blacktriangleleft$

## 2.2 Proof of Proposition 12

In this section, we prove Proposition 12 using a refined version of the analysis of [8]. The proof consists of three main steps: first, we use Bayes' formula to reduce the task of upper bounding the probability of sparsifying values into the task of upper bounding the probability of a related type of values, called *skewing* values; then, we use Vazirani's lemma to reduce the latter task to the task of the upper bounding the biases of  $g(x_I, Y_I)$ . Finally, we upper bound the biases of  $g(x_I, Y_I)$  using the low deficiency of  $X$  and  $Y$  and the discrepancy of  $g$ . We start by formally defining skewing values, and then prove their connection to sparsifying values.

► **Definition 14.** Let  $J \subseteq [n]$  and let  $y_J \in \Lambda^J$ . Let  $e(y_J)$  be the real number such that

$$\Pr[Y_J = y_J] = 2^{\sigma_Y \cdot \Delta \cdot |J| - b \cdot |J| - e(y_J)}$$

We note that this number is non-negative as we assume  $Y$  is  $\sigma_Y$ -sparse. We say that  $x$  is  $(\varepsilon, t)$ -skewing for  $y_J$  if there exist  $I \subseteq [n] - J$  such that

$$D_\infty(g^I(x_I, Y_I) \mid Y_J = y_J) > \varepsilon \cdot \Delta \cdot |J| + e(y_J) + t - 1$$

► **Proposition 15.** Let  $x \in \Lambda^n$ ,  $J \subseteq [n]$ , and  $y_J \in \Lambda^J$ . If  $x$  is  $(\varepsilon, t)$ -sparsifying for  $y_J$  and is almost uniform then  $x$  is  $(\varepsilon, t)$ -skewing for  $y_J$ .

## 26:10 Lifting with Functions of Polynomial Discrepancy

**Proof.** The proof is straightforward and been omitted due to space constraints. The proof can be found in the full version of this paper. ◀

We now formally define biasing values and connect them to skewing values via the usage of Vazirani lemma, thus allowing us to focus on the biases. Informally, biasing values are values  $x$  such that when conditioning on  $Y_J = y_J$ , the bias of  $g^{\oplus S}(x_S, Y_S)$  is too high.

► **Definition 16.** Let  $J \subseteq [n]$  and let  $y_J \in \Lambda^J$ . We say that  $x$  is  $(\varepsilon, t)$ -biasing for  $y_J$  if there exists a set  $S \subseteq [n] - J$  such that  $|S| \geq c \cdot \varepsilon \cdot |J| + \frac{t+e(y_J)-2}{\log n}$  and

$$\text{bias}(g^{\oplus S}(x_S, Y_S) \mid Y_J = y_J) > (2n)^{-|S|}.$$

► **Proposition 17.** Let  $x \in \Lambda^n$ , let  $J \subseteq [n]$ , and let  $y_J \in \Lambda^J$ . If  $x$  is not  $(\varepsilon, t)$ -biasing for  $y_J$  then  $x$  is not  $(\varepsilon, t)$ -skewing for  $y_J$ .

**Proof.** The proof is omitted due to space constraints and can be found in the full version of this paper. ◀

We finally prove Proposition 12, restated next.

► **Proposition 12.** Let  $\gamma > \frac{2}{c}$ . Additionally assume that  $\varepsilon \geq \frac{5}{c}$  and  $\tau \leq 1 - \frac{14}{c} - \gamma$ . Then, for every  $J \subseteq [n]$  and for every  $y_J \in \Lambda^J$ , the probability that  $X$  takes an almost uniform value  $x$  that is  $(\varepsilon, t)$ -sparsifying for  $y_J$  is at most  $2^{-\gamma \cdot \Delta \cdot \frac{c-\varepsilon}{2} \cdot |J| - 2t}$ .

**Proof.** Let  $J \subseteq [n]$  and let  $y_J \in \Lambda^J$ . We first observe that it suffices to prove that with probability at least  $1 - 2^{-\gamma \cdot \Delta \cdot |J| - 2t}$ , the random variable  $X$  takes a value  $x$  that is not  $(\varepsilon, t)$ -biasing for  $y_J$ . Indeed, if  $x$  is a value that is not  $(\varepsilon, t)$ -biasing for  $y_J$ , then by Proposition 17 it is not  $(\varepsilon, t)$ -skewing for  $y_J$ , and then by Proposition 15 it cannot be both  $(\varepsilon, t)$ -sparsifying for  $y_J$  and almost uniform. It remains to upper bound the probability that  $x$  is  $(\varepsilon, t)$ -biasing for  $y_J$ .

We start by upper bounding the probability that  $X$  takes a value  $x$  such that

$$\text{bias}(g^{\oplus S}(x_S, Y_S) \mid Y_J = y_J) > (2n)^{-|S|}$$

for some non-empty fixed set  $S \subseteq [n] - J$  such that  $|S| \geq c \cdot \varepsilon \cdot |J| + \frac{t+e(y_J)-2}{\log n}$ . Let  $S$  be such a set. In order to upper bound the latter probability, we use Lemma 13, which in turn requires us to upper bound the deficiencies  $D_\infty(X_S)$  and  $D_\infty(Y_S \mid Y_J = y_J)$ . By assumption, we know that  $D_\infty(X_S) \leq \sigma_X \cdot \Delta \cdot |S|$ . We turn to upper bound  $D_\infty(Y_S \mid Y_J = y_J)$ . For every  $y_S \in \Lambda^S$ , it holds that

$$\begin{aligned} \Pr[Y_S = y_S \mid Y_J = y_J] &= \frac{\Pr[Y_{S \cup J} = y_{S \cup J}]}{\Pr[Y_J = y_J]} \\ &= \frac{\Pr[Y_{S \cup J} = y_{S \cup J}]}{2^{\sigma_Y \cdot \Delta \cdot |J| - b \cdot |J| - e(y_J)}} && \text{(Definition of } e(y_J)) \\ &\leq \frac{2^{\sigma_Y \cdot \Delta \cdot (|S| + |J|) - b \cdot (|S| + |J|)}}{2^{\sigma_Y \cdot \Delta \cdot |J| - b \cdot |J| - e(y_J)}} && (Y \text{ is } \sigma_Y\text{-sparse}) \\ &= 2^{\sigma_Y \cdot \Delta \cdot |S| + e(y_J) - b \cdot |S|}. \end{aligned}$$

It follows that

$$D_\infty(Y_S \mid Y_J = y_J) \leq \sigma_Y \cdot \Delta \cdot |S| + e(y_J).$$

By our assumption on the size of  $S$ , it follows that

$$e(y_J) \leq \log n \cdot |S| + 2 \leq 3 \cdot \log n \cdot |S| \leq \frac{3}{c} \cdot \Delta \cdot |S|.$$

It follows that

$$\begin{aligned} & D(X_S) + D_\infty(Y_S \mid Y_J = y_J) \\ & \leq (\sigma_X + \sigma_Y + \frac{3}{c}) \cdot \Delta \cdot |S| \\ & \leq (1 - \frac{11}{c} - \gamma) \cdot \Delta \cdot |S| \quad (\sigma_X + \sigma_Y \leq 1 - \frac{14}{c} - \gamma) \\ & = \left( \Delta - \frac{7\Delta}{c} - \gamma\Delta - \frac{4\Delta}{c} \right) \cdot |S| \\ & \leq (\Delta - 7 - \gamma\Delta - 3 \log n - 1) \cdot |S|. \end{aligned}$$

Now, by applying Lemma 13 with  $\gamma = \gamma\Delta + 2 \log n$  and  $\lambda = \log n + 1$ , it follows that the probability that  $X$  takes a value  $x$  such that

$$\text{bias}(g^{\oplus S}(x_S, Y_S) \mid Y_J = y_J) > (2n)^{-|S|}$$

is at most

$$2^{-\gamma \cdot \Delta \cdot |S|} \cdot \frac{1}{n^{2|S|}} \leq 2^{-\gamma \cdot \Delta \cdot |S|} \cdot \frac{1}{n^{|S|+1}},$$

where the inequality holds since  $S$  is assumed to be non-empty. By taking union bound over all relevant sets  $S$ , it follows that the probability that  $X$  takes a value  $x$  that is  $(t, \varepsilon)$ -biasing for  $y_J$  is at most

$$\begin{aligned} & \sum_{S \subseteq [n]: |S| \geq c \cdot \varepsilon \cdot |J| + \frac{t + e(y_J) - 2}{\log n}} 2^{-\gamma \cdot \Delta \cdot |S|} \cdot \frac{1}{n^{|S|+1}} \\ & \leq \sum_{S \subseteq [n]: |S| \geq \left(\frac{c \cdot \varepsilon}{2} + 2\right) |J| + \frac{t-2}{\log n}} 2^{-\gamma \cdot \Delta \cdot |S|} \cdot \frac{1}{n^{|S|+1}} \quad (e(y_J) \geq 0, \varepsilon \geq \frac{5}{c}) \\ & \leq 2 \cdot 2^{-\gamma \cdot \Delta \cdot \left(\left(\frac{c \cdot \varepsilon}{2} + 2\right) |J| + \frac{t-2}{\log n}\right)} \cdot \frac{1}{n} \quad (\text{binomial like bound}) \\ & \leq 2^{-\gamma \cdot \Delta \cdot \left(\frac{c \cdot \varepsilon}{2} + 2\right) |J|} \cdot 2^{-2t + \left(\frac{\gamma \cdot \Delta \cdot 2}{\log n}\right)} \quad (\gamma \geq \frac{2}{c} \geq \frac{2 \log n}{\Delta}, n \geq 2) \\ & \leq 2^{-\gamma \cdot \Delta \cdot \frac{c \cdot \varepsilon}{2} \cdot |J| - 2t} \cdot 2^{\gamma \cdot \Delta \cdot \left(\frac{2}{\log n} - 2\right)} \quad (|J| \geq 1) \\ & \leq 2^{-\gamma \cdot \Delta \cdot \frac{c \cdot \varepsilon}{2} \cdot |J| - 2t} \quad (n \geq 2) \end{aligned}$$

as required. ◀

### 2.3 Proof of the Main Lemma from Propositions 10 and 12

In this section, we derive the main lemma from the previous propositions. The difficult part is to prove an upper bound on the probability of non-recoverable values  $x$ , which is essentially equivalent to proving the following statement:

■ There are very few values  $x$  that are sparsifying for many values  $y_J$ .

Proposition 12 essentially tells us the following statement:

■ For every  $y_J$ , there are very few values  $x$  that are sparsifying for it.

## 26:12 Lifting with Functions of Polynomial Discrepancy

It is tempting to try to deduce the first statement from the second statement via an averaging argument. However, there is a significant obstacle here: in the first statement, when we say “for many values  $y_J$ ”, we count the values  $y_J$  with respect to a distribution that depends on  $x$ . This complication renders a naive averaging argument impossible. In order to overcome this obstacle, we consider all the pairs  $(x, y_J)$  such that  $x$  is  $(\varepsilon, t)$ -sparsifying for  $y_J$ , and place them into buckets according to the value of  $t$ . Then, we bound the weight of each bucket separately, while making use of the fact that Proposition 12 provides a stronger upper bound for larger values of  $t$ . Using this bucketing scheme turns out to be sufficient for the averaging argument to go through.

We start by defining the notion of a “light” value of  $X$ , which is a value  $x$  that is *not* sparsifying for many values  $y_J$  for a particular set  $J \subseteq [n]$ . The term “light” is motivated by the intuitive idea that the relevant values  $y_J$  are “heavy” in terms of their probability mass, so a “light” value  $x$  is one that does not make many values  $y_J$  “heavy”. We show that “light” values of  $x$  are recoverable, intuitively this is true as one can remove all the relevant values  $y_J$  that cause the sparsity by condition on high the probability event of not choosing any of them. We then consider  $x$  that are not light with respect to some specific  $J$ . We proceed by bounding the probability of  $x$  that are not light with respect to single  $J$ , and complete the proof by taking union bound over all  $J$ .

► **Definition 18.** Let  $x \in \Lambda^n$  and let  $J \subseteq [n]$ . For every set  $I \subseteq [n] - J$  and a value  $z_I \in \{0, 1\}^I$ , we denote by

$$\mathcal{H}_{x,J,I,z_I} \stackrel{\text{def}}{=} \left\{ y_J \in \Lambda^J : \Pr[Y_J = y_J \mid g^I(x_I, Y_I) = z_I] > 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|} \right\}$$

the set of “heavy” values  $y_J$ . We say that a value  $x$  is  $(\varepsilon, \alpha)$ -light if for every disjoint  $I, J$  and  $z_I \in \{0, 1\}^I$  it holds that

$$\Pr[Y_J \in \mathcal{H}_{x,J,I,z_I} \mid g^I(x_I, Y_I) = z_I] \leq 2^{-\alpha \Delta} \cdot \left(\frac{1}{2n}\right)^{|J|}.$$

► **Proposition 19.** Let  $\alpha \geq \frac{1}{\Delta}$ . If  $x \in \Lambda^n$  is  $(\varepsilon, \alpha)$ -light with respect to every  $J \subseteq [n]$  then it is  $(\varepsilon, \alpha)$ -recoverable.

**Proof.** Let  $\alpha \geq \frac{1}{\Delta}$  and let  $x \in \Lambda^n$  be  $(\varepsilon, \alpha)$ -light with respect to every  $J \subseteq [n]$ . We show that  $x$  is  $(\varepsilon, \alpha)$ -recoverable by showing that for every  $I \subseteq [n]$  and  $z_I \in \Lambda^I$  there exists an event  $\mathcal{E}$  such that the random variable

$$Y_{[n]-I} \mid \mathcal{E} \text{ and } g^I(x_I, Y_I) = z_I$$

is  $(\sigma_Y + \varepsilon)$ -sparse. We choose  $\mathcal{E}$  to be the event that  $Y_J \notin \mathcal{H}_{x,J,I,z_I}$  for any non-empty set  $J \subseteq [n] - I$ . We first prove that  $\Pr[\neg \mathcal{E} \mid g(x_I, Y_I) = z_I] < 2^{-\alpha \Delta}$ . By the union bound, it holds that

$$\begin{aligned} & \Pr[\neg \mathcal{E} \mid g(x_I, Y_I) = z_I] \\ &= \Pr \left[ \bigvee_{\emptyset \neq J \subseteq [n]-I} Y_J \in \mathcal{H}_{x,J,I,z_I} \mid g(x_I, Y_I) = z_I \right] \\ &\leq \sum_{\emptyset \neq J \subseteq [n]-I} \Pr[Y_J \in \mathcal{H}_{x,J,I,z_I}] \\ &\leq \sum_{\emptyset \neq J \subseteq [n]-I} 2^{-\alpha \Delta} \cdot \left(\frac{1}{2n}\right)^{|J|} \\ &\leq 2^{-\alpha \Delta} \end{aligned} \quad (\text{binomial like bound})$$



It remains to prove that the random variable

$$Y_{[n]-I} \mid \mathcal{E} \text{ and } g^I(x_I, Y_I) = z_I$$

is  $(\sigma_Y + \varepsilon)$ -sparse. For every  $J \subseteq [n] - I$ , it holds that

$$\begin{aligned} & \Pr[Y_J = y_J \mid \mathcal{E} \text{ and } g^I(x_I, Y_I) = z_I] \\ &= \frac{\Pr[Y_J = y_J \text{ and } \mathcal{E} \mid g(x_I, Y_I) = z_I]}{\Pr[\mathcal{E} \mid g(x_I, Y_I) = z_I]} \\ &\leq \frac{\Pr[Y_J = y_J \mid g(x_I, Y_I) = z_I]}{\Pr[\mathcal{E} \mid g(x_I, Y_I) = z_I]} \\ &\leq \frac{2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|}}{1 - 2^{-\alpha \Delta}} \\ &\leq \frac{2^{(\sigma_Y + \varepsilon) \cdot \Delta \cdot |J| - b \cdot |J| - 1}}{1 - \frac{1}{2}} \quad (\text{since } \alpha \geq \frac{1}{\Delta}) \\ &\leq 2^{(\sigma_Y + \varepsilon) \cdot \Delta \cdot |J| - b \cdot |J|}, \end{aligned}$$

and therefore the above random variable is  $(\sigma_Y + \varepsilon)$ -sparse, as required.  $\blacktriangleleft$

► **Definition 20.** Let  $x \in \Lambda^n$ ,  $J \subseteq [n]$ ,  $I \subseteq [n] - J$  and  $z_I \in \{0, 1\}^I$ , recall that

$$\mathcal{H}_{x, J, I, z_I} \stackrel{\text{def}}{=} \left\{ y_J \in \Lambda^J : \Pr[Y_J = y_J \mid g^I(x_I, Y_I) = z_I] > 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|} \right\}.$$

We say that a value  $x$  is  $(\varepsilon, \alpha)$ -light with respect to  $J$  if for every  $I \subseteq [n] - J$  and  $z_I \in \{0, 1\}^I$  it holds that

$$\Pr[Y_J \in \mathcal{H}_{x, J, I, z_I} \mid g^I(x_I, Y_I) = z_I] \leq 2^{-\alpha \Delta} \cdot \left( \frac{1}{2n} \right)^{|J|}.$$

It is easy to see that by definition a value  $x$  is  $(\varepsilon, \alpha)$ -light if it is  $(\varepsilon, \alpha)$ -light with respect to every  $J \subseteq [n]$ . We now use this notion to bound the probability of  $x$  been not light for every  $J$  and later get bound on the probability that  $X$  is  $(\varepsilon, \alpha)$ -light by binomial like bound.

► **Proposition 21.** Assume that  $\sigma_X + 2 \cdot \sigma_Y \leq 1 - \frac{19}{c} - \gamma - \alpha$ . For every  $J \subseteq [n]$ , the probability that  $X$  takes an almost uniform value  $x$  that is not  $(\varepsilon, \alpha)$ -light for  $J$  is at most  $2^{-\gamma \cdot \Delta \cdot |J|}$ .

**Proof.** Let  $J \subseteq [n]$ . Let  $\mathcal{X}$  and  $\mathcal{Y}_J$  denote the supports of  $X$  and  $Y_J$  respectively. For every  $x \in \mathcal{X}$  and  $y_J \in \mathcal{Y}_J$ , let  $t_{x, y_J}$  denote the maximal value  $t$  such that  $x$  is  $(\varepsilon - \frac{1}{\Delta}, t)$ -sparsifying for  $y_J$ . Next, consider a two dimensional table whose rows and columns are indexed by  $\mathcal{X}$  and  $\mathcal{Y}_J$  respectively. For every row  $x \in \mathcal{X}$  and column  $y_J \in \mathcal{Y}_J$ , we set the corresponding entry to be

$$\text{ent}(x, y_J) \stackrel{\text{def}}{=} \begin{cases} 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J| + t_{x, y_J}} & t_{x, y_J} > 0 \text{ and } x \text{ is almost uniform} \\ 0 & \text{otherwise.} \end{cases}$$

Now we use bucketing argument to bound the probabilities of high values of  $\text{ent}(x, y_J)$  to occur. Let  $\gamma' = \gamma + \sigma_Y + \frac{2}{c} + \alpha + \frac{3}{c}$ . By applying Proposition 12 with  $\gamma = \gamma'$ , we get that for every  $y_J$  and every  $t \in \mathbb{Z}_{>0}$  it holds that

$$\Pr[[t_{X, y_J} = t \text{ and } X \text{ is almost uniform}] \leq 2^{-\gamma' \cdot \Delta \cdot \frac{c \cdot \varepsilon}{2} \cdot |J| - 2(t-1)}.$$

## 26:14 Lifting with Functions of Polynomial Discrepancy

Therefore, for every  $y_J \in \mathcal{Y}_J$ , the expected random entry in the  $y_J$ -th column (over the random choice of  $X$ ) is

$$\begin{aligned}
\mathbb{E}[\text{ent}(X, y_J)] &\leq 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|} \cdot \sum_{t=1}^{\infty} \Pr[\lceil t_{X, y_J} \rceil = t \text{ and } X \text{ is not leaking}] \cdot 2^t \\
&\leq 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|} \cdot \sum_{t=1}^{\infty} 2^{-\gamma' \cdot \Delta \cdot \frac{c \cdot \varepsilon}{2} \cdot |J| - 2(t-1)} \cdot 2^t \\
&= 2^{-\gamma' \cdot \Delta \cdot \frac{c \cdot \varepsilon}{2} \cdot |J| + 2} \cdot 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J|} \cdot \sum_{t=1}^{\infty} 2^{-t} \\
&\leq 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta} - \gamma' \cdot \frac{c \cdot \varepsilon}{2} + \frac{2}{c}) \cdot \Delta \cdot |J| - b \cdot |J|}.
\end{aligned}$$

It follows that the expected sum of a random row of the table (over the random choice of  $X$ ) is

$$\begin{aligned}
&\mathbb{E} \left[ \sum_{y_J \in \mathcal{Y}_J} \text{ent}(X, y_J) \right] \\
&= \sum_{y_J \in \mathcal{Y}_J} \mathbb{E}[\text{ent}(X, y_J)] \\
&\leq \sum_{y_J \in \mathcal{Y}_J} 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta} - \gamma' \cdot \frac{c \cdot \varepsilon}{2} + \frac{2}{c}) \cdot \Delta \cdot |J| - b \cdot |J|} \\
&= 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta} - \gamma' \cdot \frac{c \cdot \varepsilon}{2} + \frac{2}{c}) \cdot \Delta \cdot |J|} \\
&= 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta} - \gamma - \sigma_Y - \frac{2}{c} \cdot \frac{c \cdot \varepsilon}{2} - \alpha - \frac{3}{c} + \frac{2}{c}) \cdot \Delta \cdot |J|} \quad (\text{definition of } \gamma', \frac{c \cdot \varepsilon}{2} \geq 1) \\
&\leq 2^{-(\gamma + \alpha) \cdot \Delta \cdot |J|}. \quad (\Delta \geq c)
\end{aligned}$$

By Markov's inequality, the probability that  $X$  is almost uniform and the sum of the  $X$ -th row is more than  $2^{-\alpha \cdot \Delta \cdot |J|}$  is upper bounded by  $2^{-\gamma \cdot \Delta \cdot |J|}$ . We now prove that if a value  $x \in \mathcal{X}$  is almost uniform and the sum in the  $x$ -th row is at most  $2^{-\alpha \cdot \Delta \cdot |J|}$ , then  $x$  is  $(\varepsilon, \alpha)$ -light with respect to  $J$ , and this will finish the proof of the proposition.

Let  $x \in \mathcal{X}$  be such a value. We prove that  $x$  is  $(\varepsilon, \alpha)$ -light with respect to  $J$ . Let  $I \subseteq [n] - J$  and let  $z_I \in \{0, 1\}^I$ . We would like to prove that

$$\Pr[Y_J \in \mathcal{H}_{x, J, I, z_I} \mid g^I(x_I, Y_I) = z_I] \leq 2^{-\alpha \Delta} \cdot \left( \frac{1}{2n} \right)^{|J|}.$$

Observe that for every value  $y_J \in \mathcal{H}_{x, J, I, z_I}$ , it holds that  $x$  is  $(\varepsilon - \frac{1}{\Delta}, t')$ -sparsifying for  $y_J$  with some  $t' \geq 0$ . Therefore, for every such  $y_J$  it holds that  $t_{x, y_J} > 0$  and in particular  $\text{ent}(x, y_J) = 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J| + t_{x, y_J}}$ . Furthermore, recall that by the definition of  $t_{x, y_J}$  it holds that

$$\Pr[Y_J = y_J \mid g^I(x_I, Y_I) = z_I] \leq 2^{(\sigma_Y + \varepsilon - \frac{1}{\Delta}) \cdot \Delta \cdot |J| - b \cdot |J| + t_{x, y_J}}.$$

It follows that

$$\begin{aligned}
&\Pr[Y_J \in \mathcal{H}_{x, J, I, z_I} \mid g^I(x_I, Y_I) = z_I] \\
&\leq \sum_{y_J \in \mathcal{H}_{x, J, I, z_I}} \Pr[Y_J = y_J \mid g^I(x_I, Y_I) = z_I] \\
&\leq \sum_{y_J \in \mathcal{H}_{x, J, I, z_I}} \text{ent}(x, y_J). \\
&\leq 2^{-\alpha \Delta \cdot |J|}
\end{aligned}$$

as required. ◀

► **Remark.** In the preceded proof we bound the probability that

$$\Pr[t_{X,y_J} = t \text{ and } X \text{ is almost uniform}]$$

. At a first glance the usage of ceiling may be unclear. The ceiling in this argument is merely the implantation of the bucketing argument that used in the proof. Furthermore, the bucketing argument is needed as our tools such as Proposition 12 bound the probability of  $t$  to pass some threshold, if we not additionally give upper bound on  $t$  then the increment of the contribution for the exception become unlimited and those we need some for of bucketing. On the other hand one suggest creating “zero sized” buckets around every value of  $t$ , and thus removing the need for ceiling but that way the sum can be infinite.

We conclude the following bound by taking union bound of the probability for  $x$  to be not light over all  $J$ , yielding a bound for the probability that  $x$  is not light.

► **Corollary 22.** *Assume that  $\sigma_X + 2 \cdot \sigma_Y \leq 1 - \frac{21}{c} - \gamma - \alpha$ . Then, the probability that  $X$  takes an almost uniform value  $x$  that is not  $(\varepsilon, \alpha)$ -recoverable is at most  $2^{-\gamma \Delta}$ .*

**Proof.** By applying Proposition 21 with  $\gamma = \gamma + \frac{2}{c}$ , we obtain that for every set  $J \subseteq [n]$ , the probability that  $X$  takes an almost uniform value  $x$  that is not  $(\varepsilon, \alpha)$ -light for  $J$  is at most  $2^{-\gamma \Delta} \cdot \frac{1}{(2n)^{|J|}}$ . By binomial like bound, we obtain that with probability at least  $1 - 2^{-\gamma \Delta}$ , the random variable  $X$  takes a value  $x$  that is  $(\varepsilon, \alpha)$ -light for  $J \subseteq [n]$ . Such a value  $x$  is  $(\varepsilon, \alpha)$ -recoverable by Proposition 19, so the required result follows. ◀

We finally complete the proof of Proposition 9, restated next.

► **Proposition 9 (Main Lemma).** *Let  $\varepsilon \geq \frac{5}{c}$ ,  $\alpha > \frac{1}{c}$ ,  $\gamma > 0$ , and let  $X$  and  $Y$  be independent  $(\rho, \tau)$ -structured random variables. Let  $\sigma_X, \sigma_Y > 0$  be such that  $X_{\text{free}(\rho)}$  is  $\sigma_X$ -sparse, and  $Y_{\text{free}(\rho)}$  is  $\sigma_Y$ -sparse. If  $\sigma_X + 2\sigma_Y \leq \frac{9}{10} - \frac{22}{c} - \gamma - \alpha$ . Then*

$$\Pr_{x \sim X} [x \text{ is not } (\varepsilon, \alpha)\text{-safe}] \leq 2^{-\gamma \Delta}.$$

**Proof.** Any value that is not  $(\varepsilon, \alpha)$ -safe must be not almost uniform or almost uniform but not  $(\varepsilon, \alpha)$ -recoverable. By applying Proposition 10 with  $\gamma = \gamma + \frac{1}{c}$ , it follows that the probability that  $X$  takes a non almost uniform value is at most  $2^{-(\gamma + \frac{1}{c}) \cdot \Delta} \leq 2^{-\gamma \Delta - 1}$ . By applying Corollary 22 with  $\gamma = \gamma + \frac{1}{c}$ ,  $\alpha = \alpha$ , and  $\varepsilon = \varepsilon$ , it follows that the probability that  $X$  takes an almost uniform and not  $(\varepsilon, \alpha)$ -recoverable value is at most  $2^{-(\gamma + \frac{1}{c}) \cdot \Delta} \leq 2^{-\gamma \Delta - 1}$ . Therefore, the probability that  $X$  takes that is not  $(\varepsilon, \alpha)$ -safe value is at most  $2^{-\gamma \Delta - 1} + 2^{-\gamma \Delta - 1} = 2^{-\gamma \Delta}$ . ◀

## References

- 1 Anurag Anshu, Shalev Ben-David, and Srijita Kundu. On query-to-communication lifting for adversary bounds. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPICs*, pages 30:1–30:39. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 2 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. In *STOC*, pages 67–76, 2010.
- 3 Paul Beame, Toniann Pitassi, Nathan Segerlind, and Avi Wigderson. A direct sum theorem for corruption and the multiparty NOF communication complexity of set disjointness. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 52–66. IEEE Computer Society, 2005.
- 4 Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.

- 5 Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011.
- 6 Mark Braverman and Omri Weinstein. A discrepancy lower bound for information complexity. In *APPROX-RANDOM*, pages 459–470, 2012.
- 7 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS*, pages 270–278, 2001.
- 8 Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:103, 2019.
- 9 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudorandom properties. *CoRR*, abs/1704.06807, 2017.
- 10 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation beats richness: new data-structure lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1013–1020, 2018.
- 11 Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, and Robert Robere. KRW composition theorems via lifting. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 43–49. IEEE, 2020.
- 12 Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS '20)*, November 2020. Also available as ECCC TR19-186.
- 13 Tomás Feder, Eyal Kushilevitz, Moni Naor, and Noam Nisan. Amortized communication complexity. *SIAM J. Comput.*, 24(4):736–750, 1995.
- 14 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 902–911, 2018.
- 15 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 257–266. ACM, 2015.
- 16 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 847–856. ACM, 2014.
- 17 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '17)*, pages 1077–1088, 2015.
- 18 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143, 2017.
- 19 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of protocols for XOR functions. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 282–288. IEEE Computer Society, 2016.
- 20 Rahul Jain. New strong direct product results in communication complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:24, 2011.
- 21 Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A direct sum theorem in communication complexity via message compression. In *ICALP*, pages 300–315, 2003.
- 22 Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. Prior entanglement, message compression and privacy in quantum communication. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 285–296. IEEE Computer Society, 2005.

- 23 Mauricio Karchmer, Eyal Kushilevitz, and Noam Nisan. Fractional covers and communication complexity. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference, Boston, Massachusetts, USA, June 22-25, 1992*, pages 262–274. IEEE Computer Society, 1992.
- 24 Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via direct sum in communication complexity. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 – July 3, 1991*, pages 299–304. IEEE Computer Society, 1991.
- 25 Nikos Leonardos and Michael E. Saks. Lower bounds on the randomized communication complexity of read-once functions. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 341–350. IEEE Computer Society, 2009.
- 26 Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC '17)*, pages 1246–1255, 2017.
- 27 Toniann Pitassi and Robert Robere. Lifting Nullstellensatz to monotone span programs over any field. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18)*, pages 1207–1219. ACM, 2018.
- 28 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 234–243, 1997.
- 29 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.
- 30 Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '16)*, pages 406–415, 2016.
- 31 Ronen Shaltiel. Towards proving strong direct product theorems. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 107–117. IEEE Computer Society, 2001.
- 32 Alexander A. Sherstov. The pattern matrix method (journal version). *CoRR*, abs/0906.4291, 2009. [arXiv:0906.4291](https://arxiv.org/abs/0906.4291).
- 33 Alexander A. Sherstov. Strong direct product theorems for quantum communication and query complexity. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 41–50. ACM, 2011.
- 34 Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Information & Computation*, 9(5):444–460, 2009. URL: <http://www.rintonpress.com/xxqic9/qic-9-56/0444-0460.pdf>.
- 35 Xiaodi Wu, Penghui Yao, and Henry S. Yuen. Raz-McKenzie simulation with the inner product gadget. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:10, 2017.



# Exploring the Gap Between Tolerant and Non-Tolerant Distribution Testing

Sourav Chakraborty ✉ 🏠

Indian Statistical Institute, Kolkata, India

Eldar Fischer ✉ 🏠

Technion – Israel Institute of Technology, Haifa, Israel

Arijit Ghosh ✉ 🏠

Indian Statistical Institute, Kolkata, India

Gopinath Mishra ✉ 🏠

University of Warwick, Coventry, UK

Sayantan Sen ✉ 🏠

Indian Statistical Institute, Kolkata, India

---

## Abstract

The framework of distribution testing is currently ubiquitous in the field of property testing. In this model, the input is a probability distribution accessible via independently drawn samples from an oracle. The testing task is to distinguish a distribution that satisfies some property from a distribution that is far in some distance measure from satisfying it. The task of tolerant testing imposes a further restriction, that distributions close to satisfying the property are also accepted.

This work focuses on the connection between the sample complexities of non-tolerant testing of distributions and their tolerant testing counterparts. When limiting our scope to label-invariant (symmetric) properties of distributions, we prove that the gap is at most quadratic, ignoring poly-logarithmic factors. Conversely, the property of being the uniform distribution is indeed known to have an almost-quadratic gap.

When moving to general, not necessarily label-invariant properties, the situation is more complicated, and we show some partial results. We show that if a property requires the distributions to be non-concentrated, that is, the probability mass of the distribution is sufficiently spread out, then it cannot be non-tolerantly tested with  $o(\sqrt{n})$  many samples, where  $n$  denotes the universe size. Clearly, this implies at most a quadratic gap, because a distribution can be learned (and hence tolerantly tested against any property) using  $\mathcal{O}(n)$  many samples. Being non-concentrated is a strong requirement on properties, as we also prove a close to linear lower bound against their tolerant tests.

Apart from the case where the distribution is non-concentrated, we also show if an input distribution is very concentrated, in the sense that it is mostly supported on a subset of size  $s$  of the universe, then it can be learned using only  $\mathcal{O}(s)$  many samples. The learning procedure adapts to the input, and works without knowing  $s$  in advance.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Distribution Testing, Tolerant Testing, Non-tolerant Testing, Sample Complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.27

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2110.09972>

**Funding** Gopinath Mishra’s research is supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP) and by EPSRC award EP/V01305X/1.

**Acknowledgements** The authors would like to thank the reviewers of RANDOM 2022 for their valuable suggestions which improved the presentation of the paper.



© Sourav Chakraborty, Eldar Fischer, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen; licensed under Creative Commons License CC-BY 4.0  
Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakraborti and Chaitanya Swamy; Article No. 27; pp. 27:1–27:23



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

Let  $D$  be a distribution over a finite set  $\Omega$ , and  $\mathcal{P}$  be a property, that is, a set of distributions over  $\Omega$ . Given access to independent random samples from  $\Omega$  according to the distribution  $D$ , we are interested in the problem of distinguishing whether the distribution  $D$  is  $\eta$ -close to having the property  $\mathcal{P}$ , or is  $\varepsilon$ -far from having the property  $\mathcal{P}$ , where  $\eta$  and  $\varepsilon$  are two fixed proximity parameters such that  $0 \leq \eta < \varepsilon \leq 2$ . The distance of the distribution  $D$  from the property  $\mathcal{P}$  is defined as  $\min_{D' \in \mathcal{P}} \|D - D'\|_1$ , where  $\|D - D'\|_1$  denotes the  $\ell_1$ -distance between the distributions  $D$  and  $D'$ <sup>1</sup>. A distribution  $D$  is said to be  $\eta$ -close to  $\mathcal{P}$ , if the distance of  $D$  from  $\mathcal{P}$  is at most  $\eta$ . Similarly,  $D$  is said to be  $\varepsilon$ -far if the distance of  $D$  from  $\mathcal{P}$  is at least  $\varepsilon$ . The goal is to design a tester that uses as few samples as possible. For  $\eta > 0$ , the problem of distinguishing the two cases is referred to as the *tolerant distribution testing* problem of  $\mathcal{P}$ , and the particular case where  $\eta = 0$  is referred to as the *non-tolerant distribution testing* problem of  $\mathcal{P}$ . The sample complexity (tolerant and non-tolerant testing) is the number of samples required by the best algorithm that can distinguish with high probability (usually with probability at least  $\frac{2}{3}$ ) whether the distribution  $D$  is  $\eta$ -close to having the property  $\mathcal{P}$ , or is  $\varepsilon$ -far from having the property  $\mathcal{P}$ .

While results and techniques from distribution testing are already interesting in their own right, they have also found numerous applications in central problems in Theoretical Computer Science, and in particular, in property testing, e.g. graph isomorphism testing [27, 29] and function isomorphism testing [6], learning theory [10, 23, 22], and differential privacy [5, 32, 41, 1]. Thus, understanding the tolerant and non-tolerant sample complexity of distribution testing is a central problem in theoretical computer science.

There have been extensive studies of non-tolerant and tolerant testing of some specific distribution properties like uniformity, identity with a fixed distribution, equality of two distributions and independence of a joint distribution [9, 8, 35, 40, 37, 38]. Various other specific distribution properties have also been studied [7, 24]. Then, some works investigated general tests for the large class of all shape-restricted properties of distributions, which contains properties like monotonicity, log-concavity, modality etc. [15, 26]. This paper proves general results about the gap between tolerant and non-tolerant distribution testing that hold for large classes of properties.

### 1.1 Our results

We now informally present our results. The formal statements of the theorems are presented in the corresponding sections where they are proved, after the formal definitions are presented in Section 2. We assume that the distributions are supported over a set  $\Omega = [n] = \{1, 2, \dots, n\}$ . We first prove a result about label-invariant distribution properties (properties that are invariant under all permutations of  $\Omega$ ). We show that, for any label-invariant distribution property, there is at most a quadratic blowup in its tolerant sample complexity as compared to its non-tolerant counterpart, ignoring poly-logarithmic factors.

► **Theorem 1.1 (Informal).** *Any label-invariant distribution property that can be non-tolerantly tested using  $\Lambda$  samples, can also be tolerantly tested using  $\tilde{O}(\min\{\Lambda^2, n\})$  samples, where  $n$  is the size of the support of the distribution*<sup>2</sup>.

<sup>1</sup> Strictly speaking it is an infimum, but since all properties we consider are compact sets, it is equal to the minimum.

<sup>2</sup>  $\tilde{O}(\cdot)$  hides a poly-logarithmic factor.

This result gives a unified way for obtaining tolerant testers from their non-tolerant counterparts. The above result will be stated and proved formally in Section 3. We also design a constructive variant of the tolerant tester of Theorem 1.1, when the property can be expressed as the feasible solution to a set of linear inequalities.

► **Theorem 1.2 (Informal).** *Any label-invariant distribution property that can be non-tolerantly tested using  $\Lambda$  samples and can be expressed as a feasible solution to  $m$  linear inequalities, can also be tolerantly tested using  $\tilde{O}(\min\{\Lambda^2, n\})$  samples and in time polynomial in  $m$  and  $n$ , where  $n$  is the size of the support of the distribution.*

We believe that this result can be generalized to the case where the property can be expressed as the feasible solution to a set of convex constraints, using more advanced techniques.

Note that if  $\Lambda = \Omega(\sqrt{n})$ , Theorem 1.1 is obvious. It is only interesting if  $\Lambda = o(\sqrt{n})$ . Now we present a property for which this connection is useful. Consider a natural distribution property: given a distribution  $D$  and a parameter  $k$ , we want to decide whether the support size of  $D$  is at most  $k$  or  $\varepsilon$ -far from having support at most  $k$ . If  $k = o(\sqrt{n})$ , the query complexity for testing this problem is  $\mathcal{O}(\frac{k}{\log k})$  [39].

It is a natural question to investigate the extent to which the above theorem can be generalized. Though we are not resolving this question completely, as a first step in the direction of extending the above theorem for properties that are not necessarily label-invariant, we consider the notion of *non-concentrated* properties. By the notion of a non-concentrated distribution, intuitively, we mean that there is no significant portion of the base set of the distribution that carries only a negligible weight, making the probability mass of the distribution well distributed among its indices. Specifically, any subset  $X \subseteq [n]$ , for which  $|X|$  is above some threshold (say  $\beta n$  with  $\beta \in (0, \frac{1}{2})$ ), has probability mass of at least another threshold (say  $\alpha$  with  $\alpha \in (0, \frac{1}{2})$ ). A property is said to be non-concentrated if only non-concentrated distributions can satisfy the property. We prove a lower bound on the testing of any non-concentrated property (not necessarily label-invariant).

► **Theorem 1.3 (Informal).** *In order to non-tolerantly test any non-concentrated distribution property,  $\Omega(\sqrt{n})$  samples are required, where  $n$  is the size of the support of the distribution.*

The quadratic gap between tolerant testing and non-tolerant testing for any non-concentrated property follows from the above theorem, since by a folklore result, only  $\mathcal{O}(n)$  many samples are required to learn any distribution approximately.

The proof of Theorem 1.3 for label-invariant non-concentrated properties is a generalization of the proof of the  $\Omega(\sqrt{n})$  lower bound for classical uniformity testing, while for the whole theorem, that is, for the general (not label-invariant) non-concentrated properties, a more delicate argument is required. The formal proof is presented in Section 5.

The next natural question is about the sample complexity of any tolerant tester for non-concentrated properties. We address this question for label-invariant non-concentrated properties by proving the following theorem in Section 4.2. However, the question is left open for non-label-invariant properties.

► **Theorem 1.4 (Informal).** *The sample complexity for tolerantly testing any non-concentrated label-invariant distribution property is  $\Omega(n^{1-o(1)})$ , where  $n$  is the size of the support of the distribution.*

A natural question related to tolerant testing is:

*How many samples are required to learn a distribution?*

As pointed out earlier, any distribution can be learnt using  $\mathcal{O}(n)$  samples. But what if the distribution happens to be *very concentrated*? We present an upper bound result for learning a distribution, in which the sample complexity depends on the minimum cardinality of any set  $S \subseteq [n]$  over which the unknown distribution is concentrated.

► **Theorem 1.5 (Informal).** *To learn a distribution approximately,  $\mathcal{O}(|S|)$  samples are enough, where  $S \subseteq [n]$  is an unknown set of minimum cardinality whose mass is close to 1. Note that  $|S|$  is also unknown, and the algorithm adapts to it.*

Observe that we cannot learn a distribution supported on the set  $S$  using  $o(|S|)$  samples, so the above result is essentially tight.

## Organization of the paper

Section 2 contains the definitions used throughout the paper. Section 3 contains the formal statement and proof of Theorem 1.1, where some of the lemmas, along with the proof of Theorem 1.2 (the constructive variant), are in the appendix. Theorems 1.3, 1.4 and 1.5 are formally stated and proved in Section 4, Section 5 and Section 6 respectively.

## 1.2 Related works

Several forms of distribution testing have been investigated for over a century in statistical theory [33, 19], while combinatorial properties of distributions have been explored over the last two decades in Algorithm Theory, Machine Learning and Information Theory [28, 34, 20]. In Algorithm Theory, the investigation into testing properties of distributions started with the work of Goldreich and Ron [30], even though it was not directly stated there in these terms. Batu, Fortnow, Rubinfeld, Smith, and White [9] launched the intensive study of property testing of distributions with the problem of equality testing<sup>3</sup>. Later, Batu, Fischer, Fortnow, Kumar, Rubinfeld and White [8] studied the problems of identity and independence testing of distributions<sup>4</sup>. Since then there has been a flurry of interesting works in this model. For example, Paninski [35] proved tight bounds on uniformity testing, Valiant and Valiant [37] resolved the tolerant sample complexity for a large class of label-invariant properties that includes uniformity testing, Acharya, Daskalakis, and Kamath [2] proved various optimal testing results under several distance measures, and Valiant and Valiant [38] studied the sample complexity of instance optimal identity testing. In [7], Batu and Cannone studied the problem of *generalized uniformity testing*, where the distribution is promised to be supported on an unknown set  $S$ , and proved a tight bound of  $\tilde{\Theta}(|S|^{2/3})$  samples for non-tolerant uniformity testing. This is in contrast to the non-tolerant uniformity testing of a distribution supported over  $[n]$ , whose sample complexity is  $\Theta(\sqrt{n})$ , ignoring the dependence on the proximity parameter. Daskalakis, Kamath, and Wright [21] studied the problem of tolerant testing under various distance measures. Very recently, Canonne, Jain, Kamath, and Li [16] revisited the problem of determining the sample complexity of tolerant identity testing, where they proved the optimal dependence on the proximity parameters. Going beyond studying specific properties, Canonne, Diakonikolas, Gouleakis, and Rubinfeld [15] studied the class of

---

<sup>3</sup> Given two unknown probability distributions that can be accessed via samples from their respective oracles, equality testing refers to the problem of distinguishing whether they are same or far from each other.

<sup>4</sup> Given an unknown distribution accessible via samples, the problem of identity testing refers to the problem of distinguishing whether it is identical to a known distribution or far from it.

*shape-restricted* properties of a distribution, a condition general enough to contain several interesting properties like monotonicity, log-concavity,  $t$ -modality etc. Their result was later improved by Fischer, Lachish, and Vasudev [26]. See the survey of Cannone [14] for a more exhaustive list.

While the most studied works concentrate on non-tolerant testing of distributions, a natural extension is to test such properties tolerantly. Since the introduction of tolerant testing in the pioneering work of Parnas, Ron and Rubinfeld [36], that defined this notion for classical (non-distribution) property testing, there have been several works in this framework. Note that it is nontrivial in many cases to construct tolerant testers from their non-tolerant counterparts, as in the case of tolerant junta testing [12] for example. In a series of works, it has been proved that tolerant testing of the most natural distribution properties, like uniformity, requires an almost linear number of samples [40, 37]<sup>5</sup>. Now a natural question arises about how the sampling complexity of tolerant testing is related to non-tolerant testing of distributions in general. To the best of our knowledge, there is no known example with more than a quadratic gap.

It would also be interesting to bound the gap for sample-based testing as defined in the work of Goldreich and Ron [31]. This model was investigated further in the work of Fischer, Lachish and Vasudev [25], where a general upper bound for non-tolerant sample-based testing of strongly testable properties was provided.

## 2 Notation and definitions

A probability distribution  $D$  over a universe  $\Omega = [n]$  is a non-negative function  $D : \Omega \rightarrow [0, 1]$  such that  $\sum_{i \in \Omega} D(i) = 1$ . For  $S \subseteq \Omega$ , the mass of  $S$  is defined as  $D(S) = \sum_{i \in S} D(i)$ , where  $D(i)$  is the mass of  $i$  in  $D$ . The support of a probability distribution  $D$  on  $\Omega$  is denoted by  $\text{SUPP}(D)$ . For any distribution  $D$ , by top  $t$  elements of  $D$ , we refer to the first  $t$  elements in the support of  $D$  when the elements in the support are sorted according to the non-increasing order of their probability masses in  $D$ . When we write  $\tilde{O}(\cdot)$ , it suppresses a poly-logarithmic term in  $n$  and the inverse of the proximity parameter(s)<sup>6</sup>. Although there are several other distance measures, in this work, we mainly focus on the  $\ell_1$  distance. We subsume polynomial dependencies only on the proximity parameters in our results for clarity of presentation.

► **Definition 2.1** (Distribution property). *Let  $\mathcal{D}$  denote the set of all distributions over  $\Omega$ . A distribution property  $\mathcal{P}$  is a topologically closed subset of  $\mathcal{D}$ <sup>7</sup>. A distribution  $D \in \mathcal{P}$  is said to be in the property or to satisfy the property. Otherwise,  $D$  is said to be not in the property or to not satisfy the property.*

► **Definition 2.2** (Label-invariant property). *Let us consider a property  $\mathcal{P}$ . For a distribution  $D$  and a permutation  $\sigma : \Omega \rightarrow \Omega$ , consider the distribution  $D_\sigma$  defined as  $D_\sigma(\sigma(i)) = D(i)$  (equivalently,  $D_\sigma(i) = D(\sigma^{-1}(i))$ ) for each  $i \in \Omega$ . If for every distribution  $D$  in  $\mathcal{P}$ ,  $D_\sigma$  is also in  $\mathcal{P}$  for every permutation  $\sigma$ , then the property  $\mathcal{P}$  is said to be label-invariant.*

<sup>5</sup> To be precise, the exact lower bounds for non-tolerant uniformity testing is  $\Omega(\sqrt{n})$ , and for tolerant uniformity testing it is  $\Omega(\frac{n}{\log n})$ , where  $n$  is the support size of the distribution and the proximity parameter  $\varepsilon$  is constant.

<sup>6</sup> We will also use  $\tilde{O}(\cdot)$  to suppress polynomials of the inverses of the differences of proximity parameters.

<sup>7</sup> We put this restriction to avoid formalism issues. In particular, the investigated distribution properties that we know of (such as monotonicity and being a  $k$ -histogram) are topologically closed.

► **Definition 2.3** (Distance between two distributions). *The distance between two distributions  $D_1$  and  $D_2$  over  $\Omega$  is the standard  $\ell_1$  distance between them, which is defined as  $\|D_1 - D_2\|_1 := \sum_{i \in \Omega} |D_1(i) - D_2(i)|$ . For  $\eta \in [0, 2]$ ,  $D_1$  and  $D_2$  are said to be  $\eta$ -close to each other if  $\|D_1 - D_2\|_1 \leq \eta$ . Similarly, for  $\varepsilon \in [0, 2]$ ,  $D_1$  and  $D_2$  are said to be  $\varepsilon$ -far from each other if  $\|D_1 - D_2\|_1 \geq \varepsilon$ .*

► **Definition 2.4** (Distance of a distribution from a property). *The distance of a distribution  $D$  from a property  $\mathcal{P}$  is the minimum  $\ell_1$ -distance between  $D$  and any distribution in  $\mathcal{P}$ . For  $\eta \in [0, 2]$ , a distribution  $D$  is said to be  $\eta$ -close to  $\mathcal{P}$  if the distance of  $D$  from  $\mathcal{P}$  is at most  $\eta$ . Analogously, for  $\varepsilon \in [0, 2]$ , a distribution  $D$  is said to be  $\varepsilon$ -far from  $\mathcal{P}$  if the distance of  $D$  from  $\mathcal{P}$  is at least  $\varepsilon$ .*

► **Definition 2.5** ( $(\eta, \varepsilon)$ -tester). *An  $(\eta, \varepsilon)$ -tester for a distribution property is a randomized algorithm that has sample access to the unknown distribution (upon query it can receive elements of  $\Omega$ , each drawn according to the unknown distribution, independently of any previous query or the algorithm's private coins), and distinguishes whether the distribution is  $\eta$ -close to the property or  $\varepsilon$ -far from the property, with probability at least  $\frac{2}{3}$ , where  $\eta$  and  $\varepsilon$  are proximity parameters such that  $0 \leq \eta < \varepsilon \leq 2$ . The tester is said to be tolerant when  $\eta > 0$ , and non-tolerant when  $\eta = 0$ .*

Now we define the notions of non-concentrated distributions and non-concentrated properties.

► **Definition 2.6** (Non-Concentrated distribution). *A distribution  $D$  over the domain  $\Omega = [n]$  is said to be  $(\alpha, \beta)$ -non-concentrated if for any set  $S \subseteq \Omega$  with size  $\beta n$ , the probability mass on  $S$  is at least  $\alpha$ , where  $\alpha$  and  $\beta$  are two parameters such that  $0 < \alpha \leq \beta < \frac{1}{2}$ .*

► **Definition 2.7** (Non-Concentrated property). *Let  $0 < \alpha \leq \beta < \frac{1}{2}$ . A distribution property  $\mathcal{P}$  is defined to be  $(\alpha, \beta)$ -non-concentrated, if all distributions in  $\mathcal{P}$  are  $(\alpha, \beta)$ -non-concentrated.*

Note that the uniform distribution is  $(\alpha, \alpha)$ -non-concentrated for every  $\alpha$ , and so is the property of being identical to the uniform distribution. Also, for any  $0 < \alpha < \frac{1}{2}$  such that  $\alpha n$  is an integer, the uniform distribution is the only  $(\alpha, \alpha)$ -non-concentrated one. Finally, observe that any arbitrary distribution is both  $(0, \beta)$ -non-concentrated and  $(\alpha, 1)$ -non-concentrated, for any  $\alpha, \beta \in (0, 1)$ .

### 3 Non-tolerant vs. tolerant sample complexities of label-invariant properties (Proof of Theorem 1.1)

We will prove that for any label-invariant property, the sample complexities of tolerant and non-tolerant testing are separated by at most a quadratic factor (ignoring some poly-logarithmic factors). Formally, the result is stated as follows:

► **Theorem 3.1** (Theorem 1.1 formalized). *Let  $\mathcal{P}$  be a label-invariant distribution property. Also, let there exist an  $(0, \varepsilon)$ -tester (non-tolerant tester) for the property  $\mathcal{P}$  with sample complexity  $\Lambda(n, \varepsilon)$ , where  $\Lambda \in \mathbb{N}$  and  $0 < \varepsilon \leq 2$ . Then for any  $\gamma_1, \gamma_2$  with  $\gamma_1 < \gamma_2$  and  $0 < \gamma_2 + \varepsilon < 2$ , there exists a  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester (tolerant tester) that has sample complexity  $\mathcal{O}\left(\frac{1}{(\gamma_2 - \gamma_1)^2} \cdot \min\{\Lambda^2 \log^2 \Lambda, n\}\right)$ , where  $\Lambda = \Lambda(n, \varepsilon)$ , and  $n$  is the size of the support of the distribution.*

Let us assume that  $D$  is the unknown distribution and  $\Lambda(n, \epsilon) \geq \Omega(\frac{1}{\epsilon})$ <sup>8</sup>. First note that if  $\Lambda = \Omega(\sqrt{n})$ , then we can construct a distribution  $\hat{D}$  such that  $\|D - \hat{D}\|_1 < \frac{\gamma_2 - \gamma_1 + \epsilon}{2}$ , by using  $\mathcal{O}\left(\frac{n}{(\gamma_2 - \gamma_1 + \epsilon)^2}\right)$  samples from  $D$ . Thereafter we can report  $D$  to be  $\gamma_1$ -close to the property if and only if  $\hat{D}$  is  $\frac{\gamma_2 + \gamma_1 + \epsilon}{2}$ -close to the property. In what follows, we discuss an algorithm with sample complexity  $\tilde{\mathcal{O}}(\Lambda^2)$  when  $\Lambda = o(\sqrt{n})$ . Also, we assume that  $n$  and  $\Lambda$  are larger than some suitable constant. Otherwise, the theorem becomes trivial.

The idea behind the proof is to classify the elements of  $\Omega$  with respect to their masses in  $D$  into *high* and *low*, as formally defined below in Definition 3.2. We argue that since  $\mathcal{P}$  is  $(0, \epsilon)$ -testable using  $\Lambda(n, \epsilon) = \mathcal{O}(q)$  samples, there cannot be two distributions  $D_1$  and  $D_2$  that are identical on all elements whose probability mass is at least  $\frac{1}{q^2}$ , for  $q = \theta(\Lambda)$  (the set  $\text{High}_{1/q^2}$  defined below), where  $D_1 \in \mathcal{P}$  but  $D_2$  is  $\epsilon$ -far from  $\mathcal{P}$ . We will formally show this in Lemma 3.3, where we will use the fact that  $\mathcal{P}$  is label-invariant. Using Lemma 3.3, we prove Lemma 3.4, that (informally) says that if two distributions are close with respect to the high mass elements, then it is not possible that one distribution is close to  $\mathcal{P}$  while the other one is far from it. In our algorithm, we intend to maintain the masses of the set  $\text{High}_{1/q^2}$ , and the term  $\Lambda^2$  in the query complexity of our algorithm corresponds to that.

► **Definition 3.2.** For a distribution  $D$  over  $\Omega$  and  $0 < \kappa < 1$ , we define

$$\text{High}_\kappa(D) = \{x \in \Omega \mid D(x) \geq \kappa\}$$

Now we define a quantity  $q \in \mathbb{N}$  where  $q = \Theta(\Lambda)$ <sup>9</sup>. Assume that  $c^*$  is a suitable large constant (independent of  $\Lambda$ ) such that, if we take  $\Lambda$  many samples from a distribution, then with probability at least  $\frac{3}{4}$ , we will not get any sample  $x$  whose mass is at most  $(\frac{c^*}{\Lambda})^2$  more than once. We define

$$q := \frac{\Lambda}{c^*}. \quad (1)$$

We will complete the proof of Theorem 3.1 by using the following two lemmas which we will prove later.

► **Lemma 3.3.** Let  $\mathcal{P}$  be a label-invariant property that is  $(0, \epsilon)$ -testable using  $\Lambda(n, \epsilon)$  samples and consider  $q$  as defined in Equation 1. Let  $D_1$  and  $D_2$  be two distributions such that  $\text{High}_{1/q^2}(D_1) = \text{High}_{1/q^2}(D_2)$ , and for all  $x \in \text{High}_{1/q^2}(D_1)$ , the probability of  $x$  is the same for both distributions, that is,  $D_1(x) = D_2(x)$ . Then it is not possible that  $D_1$  satisfies  $\mathcal{P}$  while  $D_2$  is  $\epsilon$ -far from satisfying  $\mathcal{P}$ .

► **Lemma 3.4.** Let  $\mathcal{P}$  be a label-invariant property that is  $(0, \epsilon)$ -testable using  $\Lambda(n, \epsilon)$  samples, and consider  $q$  as defined in Equation (1). Let  $\bar{D}$  and  $\tilde{D}$  be two distributions over  $\Omega$  ( $|\Omega| > 4q^2$ ) and let  $H$  contain the top  $q^2$  elements of  $\bar{D}$ . Also, assume that  $\left| \tilde{D}(\Omega \setminus H) - \bar{D}(\Omega \setminus H) \right| \leq \gamma$ . If

$$\sum_{x \in H} \left| \bar{D}(x) - \tilde{D}(x) \right| \leq \alpha, \quad (2)$$

then the following hold:

<sup>8</sup> This is a reasonable assumption for any non-trivial property.

<sup>9</sup> Note that  $q$  and  $\Lambda$  essentially denotes the same quantity. We have introduced  $q$  for writing proofs more rigorously.

1. If  $\bar{D}$  is  $\beta$ -close to  $\mathcal{P}$ , then there exists a distribution  $D_1$  in  $\mathcal{P}$  such that  $\text{High}_{1/q^2}(D_1) \subseteq H$  and

$$\sum_{x \in H} \left| D_1(x) - \tilde{D}(x) \right| + \left| D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H) \right| \leq (\alpha + \beta + \gamma). \quad (3)$$

2. If  $\bar{D}$  is  $(\varepsilon + 3\alpha + \beta + 2\gamma)$ -far from  $\mathcal{P}$  and  $D_1$  is a distribution such that  $\text{High}_{1/q^2}(D_1) \subseteq H$  and

$$\sum_{x \in H} \left| D_1(x) - \tilde{D}(x) \right| + \left| D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H) \right| \leq (\alpha + \beta + \gamma), \quad (4)$$

then the distribution  $D_1$  does not satisfy the property  $\mathcal{P}$ .

Using the above two lemmas, we will prove Theorem 3.1 in Section 3.1. We present the proofs of Lemma 3.3 and Lemma 3.4 in Appendix A.

### 3.1 Proof of Theorem 3.1

Let  $D$  be the unknown distribution that we need to test, and assume that  $\zeta = \gamma_1$ ,  $\eta = \gamma_2 - \gamma_1$ , and  $\eta' = \frac{\eta}{64}$ . We now provide a tolerant  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester, that is, a  $(\zeta, \zeta + \varepsilon + \eta)$ -tester for the property  $\mathcal{P}$ , as follows:

1. Draw  $W = \mathcal{O}\left(\frac{q^2}{\eta'} \log q\right)$  many samples from the distribution  $D$ . Let  $S \subseteq \Omega$  be the set of (distinct) samples obtained.
2. Draw additional  $\mathcal{O}\left(\frac{W}{\eta'^2} \log W\right)$  many samples  $Z$  to estimate the value of  $D(x)$  for all  $x \in S$ <sup>10</sup>.
3. Construct a set  $H$  as the union of  $S$  and arbitrary  $q^2$  many elements from  $\Omega \setminus (S \cup Z)$ .
4. Define a distribution  $\tilde{D}$  such that, for  $x \in H$ ,

$$\tilde{D}(x) = \frac{\# x \text{ in the multi-set } Z}{|Z|}.$$

And for each  $x \in \Omega \setminus H$ ,

$$\tilde{D}(x) = \frac{1 - \sum_{x \in H} \tilde{D}(x)}{|\Omega| - |H|}.$$

5. If there exists a distribution  $D_1$  in  $\mathcal{P}$  that satisfies both the following conditions:
  - (A)  $\sum_{x \in H} \left| D_1(x) - \tilde{D}(x) \right| + |D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq 26\eta' + \zeta$ .
  - (B)  $\text{High}_{1/q^2}(D_1) \subseteq H$ .
 then ACCEPT  $D$ .
6. If there does not exist any  $D_1$  in  $\mathcal{P}$  that satisfies both Conditions (A) and (B) above, then REJECT  $D$ .

Note that Step 5 as mentioned above is not completely constructive in a computational sense. In Appendix B, we give a constructive variant of the tester where the property  $\mathcal{P}$  can be expressed as a set of linear inequalities. We also give an example of a natural property that can be expressed as a set of linear inequalities.

<sup>10</sup> Instead of two sets of random samples (where the first one is to generate the set  $S$  and the other one is the multi-set  $Z$ ), one can work with only one set of random samples. But in that case, the sample complexity becomes  $\mathcal{O}(q^2 \log n)$ , as opposed to  $\mathcal{O}(q^2 \log q)$  that we are going to prove.



### Sample Complexity

The sample complexity of the tester is  $\mathcal{O}(\frac{q^2}{\eta^2} \log^2 q) = \mathcal{O}(\frac{\Lambda^2 \log^2 \Lambda}{(\gamma_2 - \gamma_1)^2})$ , which follows from the above description.

### Correctness of the algorithm

The proof of correctness of our algorithm is divided into a sequence of lemmas.

► **Lemma 3.5.** *The set  $H$  and the distribution  $\tilde{D}$  satisfies the following three properties:*

- (i) *With probability at least  $1 - \frac{1}{q}$ ,  $\text{High}_{\eta'/q^2}(D) \subseteq S \subseteq H$ .*
- (ii) *For any  $x \in H$ , if  $D(x) \geq \frac{\eta'}{10W}$ ,  $(1 - \eta')D(x) \leq \tilde{D}(x) \leq (1 + \eta')D(x)$  holds with probability at least  $1 - \frac{1}{q^4}$ .*
- (iii) *For any  $x \in \Omega$  with  $D(x) \leq \frac{\eta'}{10W}$ , either  $x \notin H$ , or  $\tilde{D}(x) \leq (1 + \eta')\frac{\eta'}{10W}$  holds with probability at least  $1 - \frac{1}{q^4}$ .*

**Proof.** Let us prove the three parts one by one:

- (i) Consider any  $x \in \text{High}_{\eta'/q^2}(D)$ , that is,  $D(x) \geq \frac{\eta'}{q^2}$ . Then the probability that  $x \notin H$  is at most  $(1 - \frac{\eta'}{q^2})^{|H|} \leq \frac{1}{q^4}$ . Applying the union bound over all the elements in  $\text{High}_{\eta'/q^2}(D)$  (at most  $\frac{q^2}{\eta'} = \mathcal{O}(q^3)$ <sup>11</sup> many elements), the claim follows.
- (ii) Since  $|Z| = \mathcal{O}(\frac{W}{\eta'^2} \log W)$ , applying Chernoff bound, we see that  $(1 - \eta')D(x) \leq \tilde{D}(x) \leq (1 + \eta')D(x)$  does not hold with probability at most  $\frac{1}{q^4}$ .
- (iii) Since  $|Z| = \mathcal{O}(\frac{W}{\eta'^2} \log W)$ , if  $x$  is in  $H$  (otherwise, we are already done), applying Chernoff bound (only on one side), the bound follows. ◀

We now bound the  $\ell_1$ -distance between  $D$  and  $\tilde{D}$  with respect to  $H$ .

► **Lemma 3.6.**  $\sum_{x \in H} |D(x) - \tilde{D}(x)| \leq 5\eta'(1 + \eta') \leq 10\eta'$  holds with probability at least  $1 - \frac{3}{q}$ .

**Proof.** Recall the definition of  $\text{High}_{\eta'/10W}(D)$ . Note that

$$\sum_{x \in H} |D(x) - \tilde{D}(x)| = \sum_{x \in \text{High}_{\eta'/10W}(D)} |D(x) - \tilde{D}(x)| + \sum_{x \in H \setminus \text{High}_{\eta'/10W}(D)} |D(x) - \tilde{D}(x)|$$

Applying Lemma 3.5 (ii) for each  $x \in \text{High}_{\eta'/10W}(D)$ , and then using union bound over all such  $x \in \text{High}_{\eta'/10W}(D)$ , the first term is bounded by  $\eta'$  with probability at least  $1 - \frac{1}{q}$ .

Now the second term, notice that for each  $x \in H \setminus \text{High}_{\eta'/10W}(D)$ ,  $D(x) \leq \frac{\eta'}{10W}$ . By Lemma 3.5 (iii), and using the union bound over all elements in  $H \setminus \text{High}_{\eta'/10W}(D)$  (note that  $|H| \leq 2W = \mathcal{O}(q^3)$ ), with probability at least  $1 - \frac{2}{q}$ ,  $\tilde{D}(x) \leq \eta'(1 + \eta')/10W$  for all  $x \in H \setminus \text{High}_{\eta'/10W}(D)$ . Since  $|H| \leq 2W$ , the second term is bounded by  $4\eta'(1 + \eta')$  with probability at least  $1 - \frac{2}{q}$ . ◀

Now we prove a lemma that shows that for every distribution  $D$ , there is a another distribution  $\bar{D}$  that is “similar” to  $D$ , and for which  $H$  contains the top  $q^2$  elements of  $\bar{D}$ .

► **Lemma 3.7.** *There exists a distribution  $\bar{D}$  such that  $H$  contains top  $q^2$  elements of  $\bar{D}$ . Moreover, the following hold:*

- (i)  $\|D - \bar{D}\|_1 \leq 2\eta'$ , with probability at least  $1 - \frac{2}{q}$ .

<sup>11</sup>This follows from the assumption that  $\Lambda(n, \epsilon)$  is at least  $\Omega(1/\epsilon)$ .

- (ii)  $\sum_{x \in H} |\overline{D}(x) - \tilde{D}(x)| \leq 12\eta'$ , with probability at least  $1 - \frac{5}{q}$ .
- (iii)  $|\overline{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq 12\eta'$ , with probability at least  $1 - \frac{5}{q}$ .

**Proof.** Let  $T$  be the set of  $q^2$  largest elements of  $D$ . If  $T \subseteq S$ ,  $H$  (as  $S \subset H$ ) contains the largest  $q^2$  elements of  $D$ . In that case, setting  $\overline{D}$  to be  $D$  gives us the above results.

Now, let us consider the case where  $T \not\subseteq S$ . By Lemma 3.5 (part (i)), with probability at least  $1 - \frac{2}{q}$ ,  $\text{High}_{\eta'/q^2}(D) \subseteq S$ . Thus for any  $x \in H \setminus S$ ,  $D(x) < \frac{\eta'}{q^2}$ . Consider the set  $U = T \setminus H$ . Notice that since  $|H \setminus S| = q^2$  and  $|T| = q^2$ ,  $|U| \leq |H \setminus (T \cup S)|$ . Let  $U = \{y_1, \dots, y_{|U|}\} \subset \Omega \setminus H$ , and let  $z_1, \dots, z_{|U|}$  be some  $|U|$  elements of  $H \setminus (T \cup S)$ . Note, by definition of  $T$  and  $U$ , the set  $\{z_1, \dots, z_{|U|}\}$  and the set  $\{y_1, \dots, y_{|U|}\}$  are disjoint.

Consider the distribution  $\overline{D}$  defined as follows:

- For elements in  $\{z_1, \dots, z_{|U|}\}$ , we define  $\overline{D}(z_i) = D(y_i)$ .
- For elements in  $\{y_1, \dots, y_{|U|}\}$ , we define  $\overline{D}(y_i) = D(z_i)$ .
- For all other  $x$ , we define  $\overline{D}(x) = D(x)$ .

Note that since all the elements in the sets  $\{z_1, \dots, z_{|U|}\}$  and  $\{y_1, \dots, y_{|U|}\}$  were from  $\Omega \setminus S$ , from Lemma 3.5 (part (i)), with probability at least  $1 - \frac{2}{q}$ ,  $D(y_i) \leq \frac{\eta'}{q^2}$  and  $D(z_i) \leq \frac{\eta'}{q^2}$ , for all  $i \in \Omega \setminus S$ . Moreover, as  $|U| \leq q^2$ , we have condition (i) as well. Furthermore,  $H$  contains the largest  $q^2$  elements of  $\overline{D}$  due to its construction.

Using the triangle inequality (relative to  $H$ ) along with Lemma 3.6 and the above expression, we can say that, with probability at least  $1 - \frac{5}{q}$ , (ii) follows.

Let us now prove (iii). Since  $\overline{D}$  and  $\tilde{D}$  are distributions,  $\sum_{x \in H} \overline{D}(x) + \sum_{x \in \Omega \setminus H} \overline{D}(x) = \sum_{x \in H} \tilde{D}(x) + \sum_{x \in \Omega \setminus H} \tilde{D}(x)$ . Thus,

$$|\overline{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| = \left| \sum_{x \in H} \tilde{D}(x) - \sum_{x \in H} \overline{D}(x) \right| \leq \sum_{x \in H} |\tilde{D}(x) - \overline{D}(x)| \leq 12\eta'$$

The last inequality follows from (ii). ◀

Now we finally establish the correctness of the algorithm.

**Proof of correctness of the algorithm.** For completeness, consider the case where  $D$  is  $\zeta$ -close to  $\mathcal{P}$ . By Lemma 3.7 (i) and the triangle inequality, we know that there exists a distribution  $\overline{D}$  that is  $(\zeta + 2\eta')$ -close to  $\mathcal{P}$  and  $H$  contains the largest  $q^2$  elements of  $\overline{D}$ . Since  $\sum_{x \in H} |\overline{D}(x) - \tilde{D}(x)| \leq 12\eta'$  and  $|\overline{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq 12\eta'$  hold from Lemma 3.7 (ii) and (iii), following Lemma 3.4 for  $\alpha = 12\eta'$ ,  $\beta = \zeta + 2\eta'$  and  $\gamma = 12\eta'$ , we can say that there exists a distribution  $D_1$  in  $\mathcal{P}$  satisfying Equation (3) (which is same as satisfying **Condition (A)** and **Condition (B)** in Step 5 of the algorithm). Hence, our algorithm accepts  $D$  in Step 5.

For soundness, consider a distribution  $D$  that is  $(\varepsilon + \zeta + \eta)$ -far from  $\mathcal{P}$ . Then following Lemma 3.7 (i), we know that there exists a distribution  $\overline{D}$  that is  $(\varepsilon + \zeta + \eta - 2\eta')$ -far from  $\mathcal{P}$ , that is,  $(\varepsilon + 3\alpha + \beta + 2\gamma)$ -far from  $\mathcal{P}$ , where  $\alpha = 12\eta'$ ,  $\beta = \zeta + 2\eta'$ . Here, we are using that  $\eta = 64\eta'$  and  $\gamma = 12\eta'$ . Also Lemma 3.7 guarantees that  $H$  contains the top  $q^2$  elements of  $\overline{D}$ . Following Lemma 3.4, we know that there does not exist any such distribution  $D_1$  in  $\mathcal{P}$  that satisfies both **Condition (A)** and **Condition (B)** of Step 5 of the algorithm. Thus the algorithm will REJECT the distribution  $D$  in Step 6.

Note that the total failure probability of the algorithm is bounded by the probability that Lemma 3.7 does not hold, which is at most  $\frac{12}{q}$ . ◀

## 4 Sample complexity of testing non-concentrated label-invariant properties

In this section we first prove a lower bound of  $\Omega(\sqrt{n})$  on the sample complexity of non-tolerant testing of any non-concentrated label-invariant property. Then we proceed to prove a tolerant lower bound of  $\Omega(n^{1-o(1)})$  samples for such properties in Section 4.2.

### 4.1 Non-tolerant lower bound (Proof of Theorem 1.3 for label-invariant properties)

Here we first prove a lower bound result analogous to Theorem 1.3, where the properties are non-concentrated and label-invariant. In Section 5, we discuss why the proof of Theorem 4.1 does not directly work for Theorem 1.3, and then prove Theorem 1.3 using a different argument.

► **Theorem 4.1** (Analogous result of Theorem 1.3 for non-concentrated label-invariant properties). *Let  $\mathcal{P}$  be any  $(\alpha, \beta)$ -non-concentrated label-invariant distribution property, where  $0 < \alpha \leq \beta < \frac{1}{2}$ . For  $\varepsilon$  with  $0 < \varepsilon < \alpha$ , any  $(0, \varepsilon)$ -tester for property  $\mathcal{P}$  requires  $\Omega(\sqrt{n})$  many samples, where  $n$  is the size of the support of the distribution.*

**Proof.** Let us first consider a distribution  $D_{yes}$  that satisfies the property. Since  $\mathcal{P}$  is an  $(\alpha, \beta)$ -non-concentrated property, by Definition 2.7,  $D_{yes}$  is an  $(\alpha, \beta)$ -non-concentrated distribution. From  $D_{yes}$ , we generate a distribution  $D_{no}$  such that the support of  $D_{no}$  is a subset of that of  $D_{yes}$ , and  $D_{no}$  is  $\varepsilon$ -far from  $\mathcal{P}$ . Hence, if we apply a random permutation over the elements of  $\Omega$ , we show that  $D_{yes}$  and  $D_{no}$  are indistinguishable, unless we query for  $\Omega(\sqrt{n})$  many samples. Below we formally prove this idea.

We will partition the domain  $\Omega$  into two parts, depending on the probability mass of  $D_{yes}$  on the elements of  $\Omega$ . Given the distribution  $D_{yes}$ , let us first order the elements of  $\Omega$  according to their probability masses. In this ordering, let  $L$  be the smallest  $2\beta n$  elements of  $\Omega$ . We denote  $\Omega \setminus L$  by  $H$ . Before proceeding further, note that the following observation gives an upper bound on the probabilities of the elements in  $L$ .

► **Observation 4.2.** *For all  $x \in L$ ,  $D_{yes}(x) \leq \frac{1-2\alpha}{1-2\beta} \frac{1}{n}$ .*

**Proof of Observation 4.2.** By contradiction, assume that there exists  $x \in L$  such that  $D_{yes}(x) > \frac{1-2\alpha}{1-2\beta} \frac{1}{n}$ . This implies, for every  $y \in H$ , that  $D_{yes}(y) > \frac{1-2\alpha}{1-2\beta} \frac{1}{n}$ . So,

$$1 = \sum_{x \in \Omega} D_{yes}(x) = \sum_{x \in L} D_{yes}(x) + \sum_{y \in H} D_{yes}(y) > D_{yes}(L) + |H| \frac{1-2\alpha}{1-2\beta} \frac{1}{n}.$$

As  $|L| = 2\beta n$  and  $D_{yes}$  is an  $(\alpha, \beta)$ -non-concentrated distribution,  $D_{yes}(L) \geq 2\alpha$ . Also,  $|H| = (1-2\beta)n$ . Plugging these into the above inequality, we get a contradiction. ◀

Note that Observation 4.2 implies that if  $S$  is a multi-set of  $o\left(\sqrt{\frac{1-2\beta}{1-2\alpha}n}\right)$  samples from  $D_{yes}$ , then with probability  $1 - o(1)$ , no element from  $L$  appears in  $S$  more than once. Now using the distribution  $D_{yes}$  and the set  $L$ , let us define a distribution  $D_{no}$  such that  $D_{no}$  is  $\varepsilon$ -far from  $\mathcal{P}$ . Note that  $D_{no}$  is a distribution that comes from a distribution over a set of distributions, all of which are not  $(\alpha, \beta)$ -non-concentrated. The distribution  $D_{no}$  is generated using the following random process:

- We partition  $L$  randomly into two equal sets of size  $\beta n$ . Let the sets be  $\{x_1, \dots, x_{\beta n}\}$  and  $\{y_1, \dots, y_{\beta n}\}$ . We first pair the elements of  $L$  randomly into  $\beta n$  pairs. Let  $(x_1, y_1), \dots, (x_{\beta n}, y_{\beta n})$  be a random pairing of the elements in  $L$ , which is represented as  $P_L$ , that is,  $P_L = \{(x_1, y_1), \dots, (x_{\beta n}, y_{\beta n})\}$ .
- The probability mass of  $D_{no}$  at  $z$  is defined as follows:
  - If  $z \notin L$ , then  $D_{no}(z) = D_{yes}(z)$ .
  - For every pair  $(x_i, y_i) \in P_L$ ,  $D_{no}(x_i) = D_{yes}(x_i) + D_{yes}(y_i)$ , and  $D_{no}(y_i) = 0$ .

We start by observing that the distribution  $D_{no}$  constructed above is supported on a set of at most  $(1 - \beta)n$  elements. So, any distribution  $D_{no}$  constructed using the above procedure is  $\epsilon$ -far from satisfying the property  $\mathcal{P}$  for any  $\epsilon < \alpha$ .

We will now prove that  $D_{yes}$  and  $D_{no}$  both have similar distributions over the sequences of samples. More formally, we will prove that any algorithm that takes  $o(\sqrt{n})$  many samples, cannot distinguish between  $D_{yes}$  from  $D_{no}$  with probability at least  $\frac{2}{3}$ .

Since any  $D_{no}$  produced using the above procedure has exactly the same probability mass on the elements in  $H$  as  $D_{yes}$ , any tester that distinguishes between  $D_{yes}$  and  $D_{no}$  must rely on samples obtained from  $L$ . Recall that the algorithm is given a uniformly random permutation of the distribution. Since  $\text{SUPP}(D_{no}) \subset \text{SUPP}(D_{yes})$  (particularly,  $\text{SUPP}(D_{no}) \cap L \subset \text{SUPP}(D_{yes}) \cap L$ ), it is not possible to distinguish between  $D_{yes}$  and  $D_{no}$ , unless an element of  $L$  appears at least twice. Otherwise, as in the proof of Lemma 3.3, the elements drawn from  $L$  are distributed identically to a uniformly random non-repeating sequence. But observe that  $D_{yes}(i) = \mathcal{O}(\frac{1}{n})$  and  $D_{no}(i) = \mathcal{O}(\frac{1}{n})$  when  $i$  is in  $L$ . Thus any sequence of  $o(\sqrt{n})$  samples will provide only a distance of  $o(1)$  between the two distributions, completing the proof.  $\blacktriangleleft$

## 4.2 Tolerant lower bound (Proof of Theorem 1.4)

► **Theorem 4.3** (Theorem 1.4 formalized). *Let  $\mathcal{P}$  be any  $(\alpha, \beta)$ -non-concentrated label-invariant distribution property, where  $0 < \alpha \leq \beta < \frac{1}{2}$ . For any constant  $\epsilon_1$  and  $\epsilon_2$  with  $0 < \epsilon_1 < \epsilon_2 < \alpha$ , any  $(\epsilon_1, \epsilon_2)$ -tester for  $\mathcal{P}$  requires  $\Omega(n^{1-o(1)})$  samples, where  $n$  is the size of the support of the distribution.*

To prove the above theorem, we recall some notions and a theorem from Valiant's paper on a lower bound for the sample complexity of tolerant testing of symmetric properties [40]. These definitions refer to invariants of distributions, which are essentially a generalization of properties.

► **Definition 4.4.** *Let  $\Pi : \mathcal{D}_n \rightarrow \mathbb{R}$  denote a real-valued function over the set  $\mathcal{D}_n$  of all distributions over  $[n]$ .*

1.  $\Pi$  is said to be label-invariant if for any  $D \in \mathcal{D}_n$  the following holds:  $\Pi(D) = \Pi(D_\sigma)$  for any permutation  $\sigma : [n] \rightarrow [n]$ .
2. For any  $\gamma, \delta$  with  $\gamma \geq 0$  and  $\delta \in [0, 2]$ ,  $\Pi$  is said to be  $(\gamma, \delta)$ -weakly-continuous if for all distributions  $p^+, p^-$  satisfying  $\|p^+ - p^-\|_1 \leq \delta$ , we have  $|\Pi(p^+) - \Pi(p^-)| \leq \gamma$ .

For a property  $\mathcal{P}$  of distributions, we define  $\Pi_{\mathcal{P}} : \mathcal{D}_n \rightarrow [0, 2]$  with respect to property  $\mathcal{P}$  as follows:

For  $D \in \mathcal{D}_n$ ,  $\Pi_{\mathcal{P}}(D) :=$  the distance of  $D$  from  $\mathcal{P}$ .

From the triangle inequality property of  $\ell_1$  distances,  $\Pi_{\mathcal{P}}$  (which refers to the distance function from the property  $\mathcal{P}$ ) is  $(\gamma, \gamma)$ -weakly continuous, for any  $\gamma \in [0, 2]$ .

► **Theorem 4.5** (Low Frequency Blindness [40]). *Consider a function  $\Pi : \mathcal{D}_n \rightarrow \mathbb{R}$  that is label-invariant and  $(\gamma, \delta)$ -weakly-continuous, where  $\gamma \geq 0$  and  $\delta \in [0, 2]$ . Let there exist two distributions  $p^+$  and  $p^-$  in  $\mathcal{D}_n$  with  $n$  being the size of their supports, such that  $\Pi(p^+) > b$ ,  $\Pi(p^-) < a$ , and they are identical for any index occurring with probability at least  $\frac{1}{n}$  in either distribution, where  $a, b \in \mathbb{R}$ . Then any tester that has sample access to an unknown distribution  $D$  and distinguishes between  $\Pi(D) > b - \gamma$  and  $\Pi(D) < a + \gamma$ , requires  $\Omega(n^{1-o_\delta(1)})$  many samples from  $D$  <sup>12</sup>.*

Note that in Theorem 4.5, we have assumed that  $p^+$  and  $p^-$  are identical for any index that has probability mass at least  $\frac{1}{n}$ . We can actually replace this condition to  $\mathcal{O}(\frac{1}{n})$  by adding  $\mathcal{O}(n)$  many “dummy elements” to the support of  $p^+$  and  $p^-$ . Now we are ready to prove Theorem 4.3.

**Proof of Theorem 4.3.** Consider  $\Pi_{\mathcal{P}}$  as defined above. As  $\mathcal{P}$  is a label-invariant property, the function  $\Pi_{\mathcal{P}}$  is also label-invariant. We have already noted that  $\Pi_{\mathcal{P}}$  is  $(\gamma, \gamma)$ -weakly continuous as “distance from a property” satisfies the triangle inequality, for any  $\gamma \in [0, 2]$ . Now recall that the distributions  $D_{yes}$  and  $D_{no}$  considered in the proof of Theorem 4.1. The probability mass of each element in the support of  $D_{yes}$  and  $D_{no}$  is  $\mathcal{O}(\frac{1}{n})$ . Note that  $D_{yes}$  is in  $\mathcal{P}$  and  $D_{no}$  is  $\varepsilon$ -far from  $\mathcal{P}$ , for any  $\varepsilon < \alpha$ , and both of them have a support size of  $\Theta(n)$ . Here we take  $\varepsilon > \varepsilon_2$ . Now, we apply Theorem 4.5 with  $a = 0$ , some  $b < \varepsilon$  and  $\gamma$  with  $\gamma < \min\{\varepsilon_1, \varepsilon - \varepsilon_2\}$ . Observe that this completes the proof of Theorem 4.3. ◀

## 5 Sample complexity of non-concentrated properties (Proof of Theorem 1.3)

► **Theorem 5.1** (Theorem 1.3 formalized). *Let  $\mathcal{P}$  be any  $(\alpha, \beta)$ -non-concentrated distribution property for  $0 < \alpha < \beta < \frac{1}{2}$ . For any  $\varepsilon$  with  $0 < \varepsilon < \alpha$ , any  $(0, \varepsilon)$ -tester for  $\mathcal{P}$  requires  $\Omega(\sqrt{n})$  many samples, where  $n$  is the size of the support of the distribution.*

### Why does the proof of Theorem 4.1 work only for label-invariant properties?

Note that the proof of Theorem 4.1 crucially uses the fact that the property  $\mathcal{P}$  is label-invariant. Recall that, while constructing  $D_{no}$  from  $D_{yes}$ , for each  $i \in [\beta n]$ , moving the masses of both  $x_i$  and  $y_i$  in  $D_{yes}$  to  $x_i$  to produce  $D_{no}$  is possible as the property  $\mathcal{P}$  is label-invariant. Because of this feature, we can apply a random permutation over  $\Omega$ , and still the permuted distribution will behave identically with respect to  $\mathcal{P}$ . After applying the random permutation, the samples coming from  $D_{yes}$  and  $D_{no}$  are indistinguishable as long as there are no collisions among the elements in  $L$ , which is the case when we take  $o(\sqrt{n})$  samples. However, this technique does not work when the property is not label-invariant, as the value of the distribution with respect to  $\mathcal{P}$  may not be invariant under the random permutation over  $\Omega$ . This requires a new argument; although the proof is similar in spirit to the proof of Theorem 4.1, there are some crucial differences, and we present the proof next. In order to prove Theorem 5.1, instead of moving the masses of both  $x_i$  and  $y_i$  in  $D_{yes}$  to  $x_i$  to produce  $D_{no}$ , we randomly move the sum to either  $x_i$  or  $y_i$  proportionally to the masses of  $x_i$  and  $y_i$ .

<sup>12</sup>  $o_\delta(\cdot)$  suppresses a term in  $\delta$ .

## 5.1 Proof of Theorem 5.1

The proof of Theorem 5.1 starts off identically to the proof of Theorem 4.1, but there is a departure in the construction of  $D_{yes}$  and  $D_{no}$ . Due to shortage of space, we only give the constructions of  $D_{yes}$  and  $D_{no}$  here, along with a brief sketch of the proof. See the full version of the paper [17] for the complete proof.

Let us first consider  $D_{yes}$ ,  $L$  and  $P_L$  as discussed in the proof of Theorem 4.1, only here we cannot and will not pass  $D_{yes}$  through a random permutation. The difference starts from the description of the distribution  $D_{no}$ . In fact,  $D_{no}$  will be randomly chosen according to a distribution over a set of distributions, all of which are not  $(\alpha, \beta)$ -non-concentrated. The distribution  $D_{no}$  is generated using the following random process:

- We partition  $L$  arbitrarily into two equal sets of size  $\beta n$ . Let the sets be  $\{x_1, \dots, x_{\beta n}\}$  and  $\{y_1, \dots, y_{\beta n}\}$ . We first pair the elements of  $L$  arbitrarily into  $\beta n$  pairs. Let  $(x_1, y_1), \dots, (x_{\beta n}, y_{\beta n})$  be an arbitrary pairing of the elements in  $L$ . Let  $P_L$  be the set of pairs. So  $P_L = \{(x_1, y_1), \dots, (x_{\beta n}, y_{\beta n})\}$ . We refer to  $x_i$  and  $y_i$  as the corresponding elements of each other with respect to  $P_L$ , and denote  $\pi(x_i) = y_i$  and  $\pi(y_i) = x_i$ .
- The probability mass of  $D_{no}$  at  $z$  is defined as follows:
  - If  $z \notin L$ , then  $D_{no}(z) = D_{yes}(z)$ .
  - For every pair  $(x_i, y_i) \in P_L$ , use independent random coins and
    - \* With probability  $\frac{D_{yes}(x_i)}{D_{yes}(x_i) + D_{yes}(y_i)}$ , set  $D_{no}(x_i) = D_{yes}(x_i) + D_{yes}(y_i)$  and  $D_{no}(y_i) = 0$ .
    - \* With the remaining probability, that is, with probability  $\frac{D_{yes}(y_i)}{D_{yes}(x_i) + D_{yes}(y_i)}$ , set  $D_{no}(x_i) = 0$  and  $D_{no}(y_i) = D_{yes}(x_i) + D_{yes}(y_i)$ .

Observe that any  $D_{no}$  constructed by the above procedure is supported on a set of at most  $(1 - \beta)n$  elements. So, any distribution  $D_{no}$  constructed using the above procedure is  $\varepsilon$ -far from satisfying the property  $\mathcal{P}$ , for any  $\varepsilon < \alpha$ . But since any  $D_{no}$  produced using the above procedure has exactly the same probability mass on elements in  $H$  as  $D_{yes}$ , any tester that distinguishes between  $D_{yes}$  and  $D_{no}$  must rely on samples obtained from  $L$ . However, we can prove that unless we receive two samples from the same pair in  $L$  (which occurs with low probability), the sample sequence cannot distinguish  $D_{yes}$  from  $D_{no}$ .

## 6 Learning a distribution (Proof of Theorem 1.5)

In this section, we prove an upper bound related to the tolerant testing of more general properties. Following a folklore result, when provided with oracle access to an unknown distribution  $D$ , we can always construct a distribution  $D'$ , such that the  $\ell_1$  distance between  $D'$  and  $D$  (the unknown distribution) is at most  $\varepsilon$ , by using  $\mathcal{O}(\frac{n}{\varepsilon^2})$  samples from  $D$ <sup>13</sup>. In this section, we provide a procedure that can be used for tolerant testing of properties, and in particular hints at how general tolerance gap bounds could be proved in the future. Our algorithm learns an unknown distribution approximately with high probability, adapting to the input, using as few samples as possible. Specifically, we prove that given a distribution  $D$ , if there exists a subset  $S \subseteq [n]$  which holds most of the total probability mass of  $D$ , then the distribution  $D$  can be learnt using  $\mathcal{O}(|S|)$  samples (even if the algorithm is unaware of  $|S|$  in advance). Our result is formally stated as follows:

<sup>13</sup>There is a writeup of this folklore result by Cannone [13].

► **Theorem 6.1** (Theorem 1.5 formalized). *Let  $D$  denote the unknown distribution over  $\Omega = [n]$ , and assume that there exists a set  $S \subseteq [n]$  with  $D(S) \geq 1 - \frac{\eta}{2}$ <sup>14</sup>, where  $\eta \in [0, 2]$  is known but  $S$  and  $|S|$  are unknown. Then there exists an algorithm that takes  $\delta \in (0, 2]$  as input and constructs a distribution  $D'$  satisfying  $\|D - D'\|_1 \leq \eta + \delta$  with probability at least  $\frac{2}{3}$ . Moreover, the algorithm uses, in expectation,  $\mathcal{O}\left(\frac{|S|}{\delta^2}\right)$  many samples from  $D$ .*

Note that in the above theorem, the algorithm has no prior knowledge of  $|S|$ . Before directly proving the above, we first show that if  $|S|$  is known, then  $\mathcal{O}(|S|)$  many samples are enough to approximately learn the distribution  $D$ . We would like to point out that similar question has been studied under the local differential privacy model with communication constraints by Acharya, Kairouz, Liu and Sun [4] and by Chen, Kairouz and Özgür [18].

► **Lemma 6.2** (Theorem 6.1 when  $|S|$  is known). *Let  $D$  be the unknown distribution over  $\Omega = [n]$  such that there exists a set  $S \subseteq [n]$  with  $|S| = s$ , and  $\eta \in [0, 2)$  such that  $D(S) \geq 1 - \frac{\eta}{2}$ , where  $s \in [n]$  and  $\eta \in (0, 1)$  are known. Then there exists an algorithm that takes  $\delta \in (0, 2]$  as an input and constructs a distribution  $D'$  satisfying  $\|D - D'\|_1 \leq \eta + \delta$  with probability at least  $\frac{9}{10}$ . Moreover, the algorithm uses  $\mathcal{O}\left(\frac{s}{\delta^2}\right)$  many samples from  $D$ .*

We note that Lemma 6.2 can be obtained from the work of Acharya, Diakonikolas, Li and Schmidt [3] (Theorem 2). For completeness, we give a self-contained proof of this lemma in the full version of the paper [17].

We later adapt the algorithm of Lemma 6.2 to give a proof to the scenario where  $|S|$  is unknown, using a guessing technique. The idea is to guess  $|S| = s$  starting from  $s = 1$ , and then to query for  $\mathcal{O}(s)$  many samples from the unknown distribution  $D$ . From the samples obtained, we construct a distribution  $D_s$ , and use Lemma 6.3 presented below to distinguish whether  $D_s$  and  $D$  are close or far. We argue that, for  $s \geq |S|$ ,  $D_s$  will be close to  $D$  with probability at least  $\frac{9}{10}$ . We bound the total probability for the algorithm reporting a  $D'$  that is too far from  $D$  (for example when terminating before  $s \geq |S|$ ), and also bound the probability of the algorithm not terminating in time when  $s$  becomes at least as large as  $|S|$ .

► **Lemma 6.3** ([37]). *Let  $D_u$  and  $D_k$  denote the unknown and known distributions over  $\Omega = [n]$  such that the support of  $D_u$  is a set of  $s$  elements of  $[n]$ . Then there exists an algorithm  $\text{TOL-ALG}(D_u, D_k, \varepsilon_1, \varepsilon_2, \kappa)$  that takes the full description of  $D_k$ , two proximity parameters  $\varepsilon_1, \varepsilon_2$  with  $0 \leq \varepsilon_1 < \varepsilon_2 \leq 2$  and  $\kappa \in (0, 1)$  as inputs, queries  $\mathcal{O}\left(\frac{1}{(\varepsilon_2 - \varepsilon_1)^2} \frac{s}{\log s} \log \frac{1}{\kappa}\right)$  many samples from  $D_u$ , and distinguishes whether  $\|D_u - D_k\|_1 \leq \varepsilon_1$  or  $\|D_u - D_k\|_1 \geq \varepsilon_2$  with probability at least  $1 - \kappa$ <sup>15</sup>.*

Note that Theorem 6.1 talks about learning a distribution with  $\mathcal{O}(s)$  samples, where there exists an unknown set  $S$  with  $s$  elements and  $D(S) \geq 1 - \eta/2$ . To prove Theorem 6.1, we use Lemma 6.3 that crucially uses less than  $s$  queries for tolerant identity testing (as opposed to learning).

The original bound following the paper of Valiant and Valiant [37] is  $\mathcal{O}\left(\frac{1}{(\varepsilon_2 - \varepsilon_1)^2} \frac{n}{\log n}\right)$ , which holds for any general distributions  $D_u$  and  $D_k$  with constant success probability. When deploying Lemma 6.3, we “contract” the set  $\Omega \setminus \text{SUPP}(D_k)$  to a single element, which allows us to substitute  $s + 1$  for  $n$ . Note that this does not change the distance between  $D_k$  and

<sup>14</sup> Recall that the variation distance between two distribution is half than that of  $\ell_1$  distance between them. So, we take  $D(S) \geq 1 - \frac{\eta}{2}$  (with  $\eta \in [0, 2)$ ) instead of  $D(S) \geq 1 - \eta$  (with  $\eta \in [0, 1)$ ).

<sup>15</sup> The multiplicative factor  $\log \frac{1}{\kappa}$  is for amplifying the success probability from  $\frac{2}{3}$  to  $1 - \kappa$ .



$D_u$ . Hence,  $\mathcal{O}\left(\frac{1}{(\varepsilon_2 - \varepsilon_1)^2} \frac{s}{\log s}\right)$  samples from  $D_u$  are enough for constant success probability. Following a recent work of Cannone, Jain, Kamath and Li [16], the dependence on the proximity parameters can be slightly improved. However we are not using that result since the focus of this work is different.

**Proof of Theorem 6.1.** The algorithm is as follows:

1. Set  $s = 1$ .
2. Query for a multi-set  $Z_s$  of  $\mathcal{O}\left(\frac{s}{\delta^2}\right)$  many samples from  $D$ .
3. Construct a distribution  $D_s : [n] \rightarrow [0, 1]$  such that

$$D_s(x) = \frac{\# \text{ times } x \text{ appears in } Z_s}{|Z_s|}$$

4. Call the algorithm TOL-ALG  $\left(D_s, D, \eta + \frac{\delta}{2}, \eta + \delta, \frac{1}{100 \log^2 s}\right)$  (corresponding to Lemma 6.3) to distinguish whether  $\|D - D_s\|_1 \leq \eta + \frac{\delta}{2}$  or  $\|D - D_s\|_1 \geq \eta + \delta$ . If we get  $\|D - D_s\|_1 \leq \eta + \frac{\delta}{2}$  as the output of TOL-ALG, then we report  $D'$  as the output and QUIT. Otherwise, we double the value of  $s$ . If  $s \leq 2n$ , go back to Step 2. Otherwise, report FAILURE.

Let  $\mathcal{S}$  denote the event that the algorithm quits with the desired output. We first show that  $\Pr(\mathcal{S}) \geq \frac{2}{3}$ . Then we analyze the expected sample complexity of the algorithm.

Observe that the algorithm quits after an iteration with guess  $s$  such that ALG-TOL reports  $\|D - D_s\|_1 \leq \eta + \frac{\delta}{2}$ . So, in that case, the probability that the algorithm exits with an output not satisfying  $\|D - D_s\|_1 \leq \eta + \delta$  is at most  $\frac{1}{100 \log^2 s}$ . When summing this up over all possible  $s$  (all powers of  $k$ , even up to infinity), the probability that the algorithm does not produce the desired output, given that it quits, is at most  $\sum_{k=1}^{\infty} \frac{1}{100k^2} \leq \frac{1}{10}$ . So, denoting  $\mathcal{Q}$  as the event that the algorithm quits without reporting FAILURE,  $\Pr(\mathcal{S} \mid \mathcal{Q}) \geq \frac{9}{10}$ .

For the lower bound on  $\Pr(\mathcal{Q})$ , consider the case where  $s \geq |S|$ . In this case,  $\|D_s - D\|_1 \leq \eta + \frac{\delta}{2}$  with probability at least  $\frac{9}{10}$ , and TOL-ALG quits by reporting  $D_s$  as the output with probability at least  $1 - \frac{1}{100 \log^2 s}$ . So, for any guess  $s \geq |S|$ , the algorithm quits and reports the desired output with probability at least  $\frac{4}{5}$ . So, the probability that the algorithm quits without reporting failure is at least the probability that the algorithm quits with a desired output at some iteration with a guess  $s \geq |S|$ , which is at least  $1 - \left(\frac{1}{5}\right)^{(\log n - \log |S| + 1)}$ . That is,  $\Pr(\mathcal{Q}) \geq \frac{4}{5}$ .

Hence, the success probability of the algorithm can be lower-bounded as

$$\Pr(\mathcal{S}) \geq \Pr(\mathcal{Q}) \cdot \Pr(\mathcal{S} \mid \mathcal{Q}) \geq \frac{9}{10} \cdot \frac{4}{5} > \frac{2}{3}.$$

Now, we analyze the sample complexity of the algorithm. The algorithm queries for  $\mathcal{O}(s)$  samples when it runs the iteration whose guess is  $s$ . The algorithm goes to the iteration with guess  $s > |S|$  if all prior iterations which guessed more than  $|S|$  failed, which holds with probability at most  $\mathcal{O}\left(\left(\frac{1}{5}\right)^{\lfloor \log s / |S| \rfloor}\right)$ . Hence the expected sample complexity of the algorithm is at most

$$\sum_{k: s=2^k < |S|} \mathcal{O}(s) + \sum_{k: s=2^k \geq |S|} \mathcal{O}\left(\left(\frac{1}{5}\right)^{\lfloor \log(s/|S|) \rfloor} \cdot s\right) = \mathcal{O}(|S|).$$

To explain the above equality, note that in the LHS of the above equation, each term of the second sum is bounded by  $\mathcal{O}\left(\left(\frac{1}{5}\right)^{(k - \log |S|)} \cdot 2^{(k - \log |S|)} \cdot |S|\right)$ . Thus, substituting  $k - \log(|S|)$  by  $r$ , we see that the second part of the LHS is upper bounded by  $\sum_{r \geq 0} \mathcal{O}\left(\left(\frac{2}{5}\right)^r \cdot |S|\right)$  which is clearly  $\mathcal{O}(|S|)$ . Thus we have the above bound.  $\blacktriangleleft$

## References

- 1 Jayadev Acharya, Clément L. Canonne, Cody Freitag, Ziteng Sun, and Himanshu Tyagi. Inference under information constraints III: local privacy constraints. *IEEE J. Sel. Areas Inf. Theory*, pages 253–267, 2021. doi:10.1109/JSAIT.2021.3053569.
- 2 Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal testing for properties of distributions. In *NIPS*, pages 3591–3599, 2015. URL: <https://proceedings.neurips.cc/paper/2015/hash/1f36c15d6a3d18d52e8d493bc8187cb9-Abstract.html>.
- 3 Jayadev Acharya, Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. Sample-optimal density estimation in nearly-linear time. In *SODA*, pages 1278–1289, 2017. doi:10.1137/1.9781611974782.83.
- 4 Jayadev Acharya, Peter Kairouz, Yuhang Liu, and Ziteng Sun. Estimating sparse discrete distributions under privacy and communication constraints. In *ALT*, pages 79–98, 2021. URL: <http://proceedings.mlr.press/v132/acharya21b.html>.
- 5 Maryam Aliakbarpour, Ilias Diakonikolas, Daniel Kane, and Ronitt Rubinfeld. Private testing of distributions via sample permutations. In *NeurIPS*, pages 10877–10888, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/8e036cc193d0af59aa9b22821248292b-Abstract.html>.
- 6 Noga Alon, Eric Blais, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. *SIAM J. Comput.*, 42(2):459–493, 2013. doi:10.1137/110832677.
- 7 Tugkan Batu and Clément L. Canonne. Generalized uniformity testing. In *FOCS*, pages 880–889, 2017. doi:10.1109/FOCS.2017.86.
- 8 Tugkan Batu, Lance Fortnow, Eldar Fischer, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *FOCS*, pages 442–451, 2001. doi:10.1109/SFCS.2001.959920.
- 9 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *FOCS*, pages 259–269, 2000. doi:10.1109/SFCS.2000.892113.
- 10 Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Mach. Learn.*, pages 151–175, 2010. doi:10.1007/s10994-009-5152-4.
- 11 Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.
- 12 Eric Blais, Clément L. Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. *ACM Trans. Comput. Theory*, pages 24:1–24:33, 2019. doi:10.1145/3337789.
- 13 Clément L. Canonne. A short note on learning discrete distributions. <https://github.com/ccanonne/probabilitydistributiontoolbox/blob/master/learning.pdf>, 2020.
- 14 Clément L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Theory of Computing Library, 2020. doi:10.4086/toc.gs.2020.009.
- 15 Clément L. Canonne, Ilias Diakonikolas, Themis Gouleakis, and Ronitt Rubinfeld. Testing shape restrictions of discrete distributions. *Theory Comput. Syst.*, pages 4–62, 2018. doi:10.1007/s00224-017-9785-6.
- 16 Clément L. Canonne, Ayush Jain, Gautam Kamath, and Jerry Li. The price of tolerance in distribution testing. In *COLT*, pages 573–624, 2022. URL: <https://proceedings.mlr.press/v178/canonne22a.html>.
- 17 Sourav Chakraborty, Eldar Fischer, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen. Exploring the gap between tolerant and non-tolerant distribution testing. *CoRR*, 2021. arXiv:2110.09972.
- 18 Wei-Ning Chen, Peter Kairouz, and Ayfer Özgür. Breaking the communication-privacy-accuracy trilemma. In *NeurIPS*, 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/222afbe0d68c61de60374b96f1d86715-Abstract.html>.

- 19 Gregory W Corder and Dale I Foreman. *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons, 2014.
- 20 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001. doi:10.1002/0471200611.
- 21 Constantinos Daskalakis, Gautam Kamath, and John Wright. Which distribution distances are sublinearly testable? In *SODA*, pages 2747–2764, 2018. doi:10.1137/1.9781611975031.175.
- 22 Ilias Diakonikolas and Daniel M. Kane. A new approach for testing properties of discrete distributions. In *FOCS*, pages 685–694, 2016. doi:10.1109/FOCS.2016.78.
- 23 Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *FOCS*, pages 73–84, 2017. doi:10.1109/FOCS.2017.16.
- 24 Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Sharp bounds for generalized uniformity testing. In *NeurIPS*, pages 6204–6213, 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/fc325d4b598aaede18b53dca4ecfcb9c-Abstract.html>.
- 25 Eldar Fischer, Oded Lachish, and Yadu Vasudev. Trading query complexity for sample-based testing and multi-testing scalability. In *FOCS*, pages 1163–1182, 2015. doi:10.1109/FOCS.2015.75.
- 26 Eldar Fischer, Oded Lachish, and Yadu Vasudev. Improving and extending the testing of distributions for shape-restricted properties. In *STACS*, pages 31:1–31:14, 2017. doi:10.4230/LIPIcs.STACS.2017.31.
- 27 Eldar Fischer and Arie Matsliah. Testing graph isomorphism. *SIAM J. Comput.*, pages 207–225, 2008. doi:10.1137/070680795.
- 28 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.
- 29 Oded Goldreich. Testing isomorphism in the bounded-degree graph model. *Electron. Colloquium Comput. Complex.*, page 102, 2019. URL: <https://eccc.weizmann.ac.il/report/2019/102>.
- 30 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electron. Colloquium Comput. Complex.*, 7(20), 2000. URL: <https://eccc.weizmann.ac.il/eccc-reports/2000/TR00-020/index.html>.
- 31 Oded Goldreich and Dana Ron. On sample-based testers. *ACM Trans. Comput. Theory*, pages 7:1–7:54, 2016. doi:10.1145/2898355.
- 32 Sivakanth Gopi, Gautam Kamath, Janardhan Kulkarni, Aleksandar Nikolov, Zhiwei Steven Wu, and Huanyu Zhang. Locally private hypothesis selection. In *COLT*, pages 1785–1816, 2020. URL: <http://proceedings.mlr.press/v125/gopi20a.html>.
- 33 Terry King. *A guide to chi-squared testing*. Taylor & Francis, 1997.
- 34 David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.
- 35 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Trans. Inf. Theory*, pages 4750–4755, 2008. doi:10.1109/TIT.2008.928987.
- 36 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, pages 1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.
- 37 Gregory Valiant and Paul Valiant. The power of linear estimators. In *FOCS*, pages 403–412, 2011. doi:10.1109/FOCS.2011.81.
- 38 Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM J. Comput.*, pages 429–455, 2017. doi:10.1137/151002526.
- 39 Gregory Valiant and Paul Valiant. Estimating the unseen: Improved estimators for entropy and other properties. *J. ACM*, pages 37:1–37:41, 2017. doi:10.1145/3125643.
- 40 Paul Valiant. Testing symmetric properties of distributions. *SIAM J. Comput.*, pages 1927–1968, 2011. doi:10.1137/080734066.
- 41 Huanyu Zhang. Statistical inference in the differential privacy model. *CoRR*, abs/2108.05000, 2021. arXiv:2108.05000.

### A Proof of Lemma 3.3 and Lemma 3.4

**Proof of Lemma 3.3.** We will prove this by contradiction. Let us assume that there are two distributions  $D_{yes}$  and  $D_{no}$  such that (i)  $D_{yes} \in \mathcal{P}$ , (ii)  $D_{no}$  is  $\varepsilon$ -far from  $\mathcal{P}$ , (iii)  $\text{High}_{1/q^2}(D_{yes}) = \text{High}_{1/q^2}(D_{no}) = A$ , and (iv) for all  $x \in A$ ,  $D_{yes}(x) = D_{no}(x)$ . Now we argue that any  $(0, \varepsilon)$ -non-tolerant tester requires more than  $\Lambda(n, \varepsilon)$  samples from the unknown distribution  $D$  to distinguish whether  $D$  is in the property or  $\varepsilon$ -far from it.

Let  $D_Y$  be a distribution obtained from  $D_{yes}$  by permuting the labels of  $\Omega \setminus A$  using a uniformly random permutation. Specifically, consider a random permutation  $\pi : \Omega \setminus A \rightarrow \Omega \setminus A$ . The distribution  $D_Y$  is as follows: (i)  $D_Y(x) = D_{yes}(x)$  for each  $x \in A$  and (ii)  $D_Y(\pi(x)) = D_{yes}(x)$  for each  $x \in \Omega \setminus A$ . Similarly, consider the distribution  $D_N$  obtained from  $D_{no}$  by permuting the labels of  $\Omega \setminus A$  using a uniformly random permutation. Note that  $D_Y$  is in  $\mathcal{P}$ , whereas  $D_N$  is  $\varepsilon$ -far from  $\mathcal{P}$ , which follows from  $\mathcal{P}$  being label-invariant.

We will now prove that  $D_Y$  and  $D_N$  provide similar distributions over sample sequences. More formally, we will prove that any algorithm that takes at most  $\Lambda(n, \varepsilon)$  many samples, cannot distinguish  $D_Y$  from  $D_N$  with probability at least  $2/3$ . We argue that this claim holds even if the algorithm is provided with additional information about the input: Namely, for all  $x \in A$ , it is told the value of  $D_Y(x)$  (which is the same as  $D_N(x)$ ). When the algorithm is provided with this information, it can ignore all samples obtained from  $A$ .

By the definition of  $A$ , for all  $x \in \Omega \setminus A$ , both  $D_Y(x)$  and  $D_N(x)$  are at most  $1/q^2$ . Let  $S_Y$  be a sequence of samples drawn according to  $D_Y$ . If  $|S_Y| \leq \Lambda(n, \varepsilon)$ , then with probability at least  $3/4$ , the sequence  $(\Omega \setminus A) \cap S_Y$  has no element that appears twice. In other words, the set  $(\Omega \setminus A) \cap S_Y$  is a set of at most  $\Lambda(n, \varepsilon)$  distinct elements from  $\Omega \setminus A$ . Since the elements of  $\Omega \setminus A$  were permuted using a uniformly random permutation, with probability at least  $3/4$ , the sequence  $(\Omega \setminus A) \cap S_Y$  is a uniformly random sequence of distinct elements from  $\Omega \setminus A$ . Similarly, if  $S_N$  is a sequence of samples drawn according to  $D_N$ , then with probability at least  $3/4$ , the sequence  $(\Omega \setminus A) \cap S_N$  is a uniformly random sequence of distinct elements from  $\Omega \setminus A$ . Thus, the distributions over the received sample sequence obtained from  $D_Y$  or  $D_N$  are of distance  $1/4$  of each other, which is strictly less than  $1/3$ .

Hence, if the algorithm obtains at most  $\Lambda(n, \varepsilon)$  many samples from the unknown distribution  $D$ , it cannot distinguish, with probability at least  $2/3$ , whether the samples are coming from  $D_Y$  or  $D_N$ .  $\blacktriangleleft$

To prove Lemma 3.4, we will need the following simple claim, whose proof we omit here.

$\triangleright$  **Claim A.1.** Let  $\sigma : [n] \rightarrow [n]$  be a permutation and let  $a_1, a_2, \dots, a_n$  and  $b_1, b_2, \dots, b_n$  be two sets of  $n$  positive real numbers. If  $a_1 \geq a_2 \geq \dots \geq a_n$  and  $b_1 \geq b_2 \geq \dots \geq b_n$  and  $\sum_{i \in [n]} a_i = \sum_{i \in [n]} b_i = 1$ , then the sum  $\sum_{i \in [n]} |a_i - b_{\sigma(i)}|$  is minimized when  $\sigma$  is the identity permutation.

Now we present the proof of Lemma 3.4.

**Proof of Lemma 3.4.** We consider the two cases separately. **(1)** If  $\overline{D}$  is  $\beta$ -close to  $\mathcal{P}$ , there exists a distribution  $D_1$  in  $\mathcal{P}$  such that  $\sum_x |\overline{D}(x) - D_1(x)| \leq \beta$ . Since  $\mathcal{P}$  is label-invariant, any permutation of  $D_1$  is also in  $\mathcal{P}$ . Without loss of generality, let us assume that the domain  $\Omega$  is a subset of  $\{1, \dots, n\}$ .

By Claim A.1, the permutation  $\sigma$  that minimizes  $\sum_x |\overline{D}(x) - D_1(\sigma(x))| \leq \beta$  is the one that orders the  $i$ -th largest element of  $D_1$  with the  $i$ -th largest element of  $\overline{D}$ , that is, if  $x$  is the element with the  $i$ -th largest probability mass in  $D_1$ , then  $\sigma(x)$  has the  $i$ -th largest probability mass in  $\overline{D}$ . Consider the distribution  $D_1^\sigma$  that is defined by  $D_1^\sigma(x) = D_1(\sigma(x))$ .

Clearly,  $H$  contains the largest  $q^2$  elements of  $\bar{D}$ , and hence also  $\text{High}_{1/q^2}(D_1^\sigma) \subseteq H$ . As  $\sum_{x \in \Omega} |D_1^\sigma(x) - \bar{D}(x)| \leq \beta$ ,  $\sum_{x \in H} |\bar{D}(x) - \tilde{D}(x)| \leq \alpha$  and  $|\bar{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq \gamma$ , by the triangle inequality, we obtain

$$\begin{aligned}
& \sum_{x \in H} |D_1^\sigma(x) - \tilde{D}(x)| + |D_1^\sigma(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \\
\leq & \sum_{x \in H} |D_1^\sigma(x) - \bar{D}(x)| + \sum_{x \in H} |\bar{D}(x) - \tilde{D}(x)| \\
& + |D_1^\sigma(\Omega \setminus H) - \bar{D}(\Omega \setminus H)| + |\bar{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \\
\leq & \sum_{x \in H} |D_1^\sigma(x) - \bar{D}(x)| + \sum_{x \in H} |\bar{D}(x) - \tilde{D}(x)| \\
& + \sum_{x \in \Omega \setminus H} |D_1^\sigma(x) - \bar{D}(x)| + |\bar{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \\
= & \sum_{x \in \Omega} |D_1^\sigma(x) - \bar{D}(x)| + \sum_{x \in H} |\bar{D}(x) - \tilde{D}(x)| + |\bar{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \\
\leq & \alpha + \beta + \gamma
\end{aligned}$$

(2) We will prove this case by contradiction. Let  $D_1 \in \mathcal{P}$  be a distribution such that  $\text{High}_{1/q^2}(D_1) \subseteq H$  and  $\sum_{x \in H} |D_1(x) - \tilde{D}(x)| + |D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq \alpha + \beta + \gamma$ . Then, as  $\sum_{x \in H} |\bar{D}(x) - \tilde{D}(x)| \leq \alpha$ , by the triangle inequality, we have

$$\sum_{x \in H} |D_1(x) - \bar{D}(x)| + |D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq 2\alpha + \beta + \gamma. \quad (5)$$

Consider the distribution  $\hat{D}$  defined as follows:

- For all  $x \in H$ ,  $\hat{D}(x) = D_1(x)$ .
- If  $D_1(H) \geq \bar{D}(H)$ , then for all  $x \in \Omega \setminus H$ ,  $\hat{D}(x) = \bar{D}(x) \cdot \phi$ , where  $\phi = \frac{1 - D_1(H)}{1 - \bar{D}(H)}$ . Notice that in this case  $\phi \leq 1$ .
- If  $D_1(H) \leq \bar{D}(H)$ , then pick the set  $T \subset \Omega \setminus H$  with  $|T| = 2q^2$  that minimizes  $\bar{D}(T)$ . Then for all  $x \in T$ ,  $\hat{D}(x) = \bar{D}(x) + \frac{\bar{D}(H) - D_1(H)}{2q^2}$  and for all  $x \in \Omega \setminus (T \cup H)$ ,  $\hat{D}(x) = \bar{D}(x)$ .

Let us first prove that  $\text{High}_{1/q^2}(\hat{D}) \subseteq H$ . In the case where  $D_1(H) \geq \bar{D}(H)$ , for all  $x \in \Omega \setminus H$ ,  $\hat{D}(x) \leq \bar{D}(x)$ . Since  $\text{High}_{1/q^2}(\bar{D}) \subseteq H$ ,  $\text{High}_{1/q^2}(\hat{D}) \subseteq H$ . Now, in the case where  $D_1(H) \leq \bar{D}(H)$ , the only  $x \in \Omega \setminus H$  for which  $\hat{D}(x) > \bar{D}(x)$  are those in  $T$ . Since  $|\Omega| > 4q^2$ , the lowest  $2q^2$  elements on  $\bar{D}$  must each have mass less than  $\frac{1}{2q^2}$ . So even if we add  $\frac{1}{2q^2}$  for any element  $x \in T$ ,  $\hat{D}(x) < 1/q^2$ . Hence in this case also  $\text{High}_{1/q^2}(\hat{D}) \subseteq H$  since  $\text{High}_{1/q^2}(\bar{D}) \subseteq H$  and  $\text{High}_{1/q^2}(D_1) \subseteq H$ . Now let us bound the  $\ell_1$  distance between  $\hat{D}$  and  $\bar{D}$ . Observe that  $\sum_{x \in \Omega \setminus H} |\hat{D}(x) - \bar{D}(x)| = |\hat{D}(\Omega \setminus H) - \bar{D}(\Omega \setminus H)|$ . This is because, in the case where  $\hat{D}(H) \geq \bar{D}(H)$ , we have  $\hat{D}(x) = \phi \cdot \bar{D}(x) \leq \bar{D}(x)$  for all  $x \in \Omega \setminus H$ . On the other hand, in the case where  $\hat{D}(H) \leq \bar{D}(H)$  then for all  $x \in \Omega \setminus H$ ,  $\hat{D}(x) \geq \bar{D}(x)$ . Thus,

$$\begin{aligned}
\sum_{x \in \Omega \setminus H} |\hat{D}(x) - \bar{D}(x)| &= |\hat{D}(\Omega \setminus H) - \bar{D}(\Omega \setminus H)| \\
&\leq |\hat{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| + |\bar{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \\
&\leq |\hat{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| + \gamma
\end{aligned}$$

Also note that, from the construction of  $\hat{D}$ , we have for all  $x \in H$ ,  $\hat{D}(x) = D_1(x)$  and thus  $\hat{D}(\Omega \setminus H) = D_1(\Omega \setminus H)$ . Thus,

$$\begin{aligned}
\|\hat{D} - \bar{D}\|_1 &= \sum_{x \in H} |\hat{D}(x) - \bar{D}(x)| + \sum_{x \in \Omega \setminus H} |\hat{D}(x) - \bar{D}(x)| \\
&\leq \sum_{x \in H} |\hat{D}(x) - \bar{D}(x)| + |\hat{D}(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| + \gamma \\
&= \left( \sum_{x \in H} |D_1(x) - \bar{D}(x)| + |D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \right) + \gamma \\
&\quad \text{(From the construction of } \hat{D} \text{)} \\
&\leq 2\alpha + \beta + 2\gamma \quad \text{(By Equation (5))}
\end{aligned}$$

Moreover, as  $\text{High}_{1/q^2}(D_1) \subseteq H$  and by the construction of  $\hat{D}$ , we have  $\text{High}_{1/q^2}(D_1) = \text{High}_{1/q^2}(\hat{D})$  and for all  $x \in \text{High}_{1/q^2}(D_1)$ ,  $D_1(x) = \hat{D}(x)$ . Since we assumed that  $D_1$  is in  $\mathcal{P}$ , using Lemma 3.3,  $\hat{D}$  is  $\varepsilon$ -close to  $\mathcal{P}$ . And since  $\|\hat{D} - \bar{D}\|_1 \leq 2\alpha + \beta + 2\gamma$ , we conclude that  $\bar{D}$  is  $(\varepsilon + 2\alpha + \beta + 2\gamma)$ -close to  $\mathcal{P}$ , which is a contradiction.  $\blacktriangleleft$

## B Computationally efficient tolerant testers

In this section we present a constructive variant of the tolerant tester studied in Section 3.1. Let us first recall the definitions of *polyhedron* and *projection map*.

► **Definition B.1** (Polyhedron). *Let  $A$  be a  $M \times N$  real matrix,  $b \in \mathbb{R}^M$  be a real vector and  $Ax \leq b$  be a system of linear inequalities. The solution set  $\{x \in \mathbb{R}^N \mid Ax \leq b\}$  of the system of inequalities is called a polyhedron. The complexity of a polyhedron is defined as  $MN$ .*

► **Definition B.2** (Projection map). *Let  $n$  be an integer. For all integers  $N \leq n$ , a projection map is denoted as  $\pi_n : \mathbb{R}^N \rightarrow \mathbb{R}^n$  and is defined as the projection of the points in  $\mathbb{R}^N$  on the first  $n$  coordinates.*

Before proceeding to our results, we first define linear property.

► **Definition B.3** (Linear property). *Without loss of generality, let us assume  $\Omega = [n]$ . A distribution property  $\mathcal{P}$  is said to be a linear property if there exists a polyhedron  $\mathcal{LP} = \{x \in \mathbb{R}^N \mid Ax \leq b\}$ , where  $A$  is a  $M \times N$  real matrix and  $b \in \mathbb{R}^M$  be a real vector, and  $\pi_n(\mathcal{LP})$ <sup>16</sup> is the set of distributions satisfying the property  $\mathcal{P}$ , that is, for every  $z := (z_1, \dots, z_n, \dots, z_N) \in \mathcal{LP}$ , the distribution  $D_z$ , defined as  $D_z(i) = z_i$ ,  $\forall i \in [n]$  satisfies the property  $\mathcal{P}$ . Conversely, for every distribution  $D$  that satisfies  $\mathcal{P}$ , there exists some  $z \in \mathcal{LP}$  such that  $D = D_z$  as defined above. The complexity of  $\mathcal{P}$  is defined as  $M \times \max\{N, n\}$ .*

Now we give an example of a linear property.

► **Remark B.4** (An example of a linear property: Approximate uniformity property). A distribution  $D$  over  $[n]$  is said to be uniform if  $D(i) = \frac{1}{n}$  for all  $i \in [n]$ . Let the property  $\mathcal{P}_{u,\varepsilon}$  denote the set of all distributions that are  $\varepsilon$ -close to the uniform distribution, where  $\varepsilon \in (0, 1)$  is a parameter. Consider the following polyhedron  $\mathcal{LP}_{u,\varepsilon}$  in  $\mathbb{R}^{2n}$ :

$$\left( \sum_{i \in [n]} z_{n+i} \leq \varepsilon \right) \quad \wedge \quad (z_i \geq 0 \quad \forall i \in [2n]) \quad \wedge \quad \left( -z_{n+i} \leq z_i - \frac{1}{n} \leq z_{n+i} \quad \forall i \in [n] \right)$$

<sup>16</sup>Note that  $\pi_n(\mathcal{LP})$  will also be a polyhedron in  $\mathbb{R}^n$ , see, e.g., Corollary 2.5 in Chapter 2 from the book by Bertsimas and Tsitsiklis [11]. However, the number of linear inequalities defining the property, which affects the running time of the tester, can sometimes be greatly reduced by using this projection.



Now observe that  $\pi_n(\mathcal{LP}_{u,\varepsilon})$  will give us the set of distributions that are  $\varepsilon$ -close to uniform, i.e., the set  $\mathcal{P}_{u,\varepsilon}$  (this would serve as the linear transformation mentioned in Definition B.3). Also note that approximate uniformity property has complexity  $\mathcal{O}(n^2)$ .

For a distribution property  $\mathcal{P}$ , let  $\mathcal{CP} \subset \mathbb{R}^n$  denote the *geometric representation* of the set of probability distributions over the set  $[n]$  that satisfy  $\mathcal{P}$  by considering of each distribution over  $[n]$  as a point in  $\mathbb{R}^n$ . For all  $\beta \in [0, 1]$ ,  $k \leq n$  and  $a \in \mathbb{R}^n$ , we define the following convex set:

$$\Delta(k, q, a, \beta) := \left\{ x \in \mathbb{R}^d : \sum_{i=1}^k |x_i - a_i| + \left| \sum_{j>k} x_j - \sum_{j>k} a_j \right| \leq \beta \wedge \forall_{i>k} x_i < \frac{1}{q^2} \right\}$$

► **Theorem B.5.** *Let  $\mathcal{P}$  be a label-invariant distribution property. If there is a  $(0, \varepsilon)$ -tester (non-tolerant tester) with sample complexity  $\Lambda(n, \varepsilon)$ , then for any  $\gamma_1, \gamma_2$  with  $\gamma_1 < \gamma_2$  and  $0 < \gamma_1 < \gamma_2 + \varepsilon < 2$ , there exists a  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester (tolerant tester) that takes  $s = \tilde{\mathcal{O}}(\Lambda^2)$  many samples and makes a single emptiness query to the set  $\mathcal{CP} \cap \Delta(\tilde{\mathcal{O}}(s), \Lambda, \tilde{D}, \beta)$ , where  $\tilde{D}$  is a known probability distribution and  $\beta = \gamma_1 + \frac{\gamma_2 - \gamma_1}{3}$ .*

**Proof.** Recall that in Step 5 of the tolerant tester presented in Section 3.1, the tester checks whether there is any distribution  $D_1 \in \mathcal{P}$  that satisfies the following two conditions:

- (i)  $\sum_{x \in H} |D_1(x) - \tilde{D}(x)| + |D_1(\Omega \setminus H) - \tilde{D}(\Omega \setminus H)| \leq 26\eta' + \zeta$
- (ii)  $\text{High}_{1/q^2}(D_1) \subseteq H$

where  $\zeta = \gamma_1$ ,  $\eta = \gamma_2 - \gamma_1$ ,  $\eta' = \frac{\eta}{64}$ . The set  $H$  and the distribution  $\tilde{D}$  are defined in the tolerant tester presented in Section 3.1.

Without loss of generality, we can assume that  $H = \{1, \dots, |H|\}$ . Therefore, in order to perform Step 5 of the tolerant tester, the following equations are needed to be satisfied:

$$D_1 \in \mathcal{CP} \text{ and } D_1 \in \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$$

We now present the tolerant  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester in its entirety, that is, a  $(\zeta, \zeta + \varepsilon + \eta)$ -tester for the property  $\mathcal{P}$ , where  $\zeta = \gamma_1$ ,  $\eta = \gamma_2 - \gamma_1$ , and  $\eta' = \frac{\eta}{64}$ .

1. Draw  $W = \mathcal{O}\left(\frac{q^2}{\eta'} \log q\right)$  many samples from the distribution  $D$ . Let  $S \subseteq \Omega$  be the set of (distinct) samples obtained.
2. Draw additional  $\mathcal{O}\left(\frac{W}{\eta'^2} \log W\right)$  samples  $Z$  to estimate the value of  $D(x)$  for all  $x \in S$ .
3. Construct a set  $H$  as the union of  $S$  and arbitrary  $q^2$  many elements from  $\Omega \setminus (S \cup Z)$ .
4. Define a distribution  $\tilde{D}$  such that, for  $x \in H$ ,  $\tilde{D}(x) = \frac{\# x \text{ in the multi-set } Z}{|Z|}$ , and for each  $x \in \Omega \setminus H$ ,  $\tilde{D}(x) = \frac{1 - \sum_{x \in H} \tilde{D}(x)}{|\Omega| - |H|}$ .
5. If there exists a distribution  $D_1 \in \mathcal{CP} \cap \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$ , then ACCEPT  $D$ .
6. If there does not exist any distribution  $D_1$  that passes Step 5, then REJECT  $D$ .

Observe that the sample complexity of the tester is  $\mathcal{O}(q^2 \log^2 q / \eta^2) = \tilde{\mathcal{O}}(\Lambda^2)$  in addition to a single emptiness query to the set  $\mathcal{P} \in \mathcal{CP} \cap \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$  in Step 5. The correctness proof of the above tester follows from the correctness argument presented in Section 3.1. ◀



### B.1 Emptiness checking when $\mathcal{P}$ is a linear property: Proof of Theorem 1.2

Now we proceed to analyze the time complexity of the  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester described in Theorem B.5 when  $\mathcal{P}$  is also a linear property. Recall that as  $\mathcal{P}$  is a linear property, there exists a polyhedron  $\mathcal{LP} = \{x \in \mathbb{R}^N \mid Ax \leq b\}$ , where  $A$  is a  $M \times N$  real matrix and  $b \in \mathbb{R}^M$  be a real vector, and  $\pi_n(\mathcal{LP})$  is the set of distributions satisfying the property  $\mathcal{P}$  (See, Definition B.3). Now in Observation B.6, we show that checking emptiness of  $\pi_n(\mathcal{LP}) \cap \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$  is equivalent to testing the feasibility of a family of inequalities.

► **Observation B.6.** *Without loss of generality, assume that  $H = \{1, \dots, |H|\}$  and  $\Omega = \{1, \dots, n\}$ . Checking emptiness of  $\pi_n(\mathcal{LP}) \cap \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$  is equivalent to testing the feasibility of the following set of inequalities:*

$$(Az \leq b) \quad \wedge \quad (z_i < \frac{1}{q^2} \quad \forall i \in [n] \setminus \{1, \dots, |H|\})$$

$$\sum_{i=1}^{|H|} |z_i - \tilde{D}(i)| + \left| \sum_{i=|H|+1}^n z_i - \sum_{i=|H|+1}^n \tilde{D}(i) \right| \leq 26\eta' + \zeta \quad (6)$$

Note that the inequality in Equation (6) can be expressed as the following set of linear inequalities using slack variables  $z_{N+i}$  for all  $i \in [|H| + 1]$ :

$$(\sum_{i=1}^{|H|} z_{N+i} + z_{N+|H|+1} \leq 26\eta' + \zeta) \quad \wedge \quad (z_{N+i} \geq 0 \quad \forall i \in [|H| + 1])$$

$$-z_{N+i} \leq z_i - \tilde{D}(i) \leq z_{N+i} \quad \forall i \in [|H|]$$

$$-z_{N+|H|+1} \leq \sum_{i=|H|+1}^n z_i - \sum_{i=|H|+1}^n \tilde{D}(i) \leq z_{N+|H|+1}$$

Therefore checking the emptiness of  $\pi_n(\mathcal{LP}) \cap \Delta(|H|, q, \tilde{D}, 26\eta' + \zeta)$  is equivalent to checking the feasibility of the following set of linear inequalities:

$$(Az \leq b) \quad \wedge \quad (\sum_{i=1}^{|H|} z_{N+i} + z_{N+|H|+1} \leq 26\eta' + \zeta) \quad \wedge \quad (z_i < \frac{1}{q^2} \quad \forall i \in [n] \setminus \{1, \dots, |H|\})$$

$$(z_{N+i} \geq 0 \quad \forall i \in [|H| + 1]) \quad \wedge \quad (-z_{N+i} \leq z_i - \tilde{D}(i) \leq z_{N+i} \quad \forall i \in [|H|])$$

$$-z_{N+|H|+1} \leq \sum_{i=|H|+1}^n z_i - \sum_{i=|H|+1}^n \tilde{D}(i) \leq z_{N+|H|+1}$$


The feasibility of the above set of linear inequalities can be solved in a polynomial time in the complexity of the polyhedron, that is, in a polynomial time in  $N$  and  $M$  using the Ellipsoid Method, where recall that  $A$  is a  $M \times N$  real matrix (see, e.g., [11]). Thus, we have an efficient  $(\gamma_1, \gamma_2 + \varepsilon)$ -tester for  $\mathcal{P}$ , that runs in time polynomial in the complexity of the linear property  $\mathcal{P}$  which is also label-invariant. This concludes the proof of Theorem 1.2.



# A Sublinear Local Access Implementation for the Chinese Restaurant Process

Peter Mörters 

Universität zu Köln, Germany

Christian Sohler 

Universität zu Köln, Germany

Stefan Walzer 

Universität zu Köln, Germany

---

## Abstract

The Chinese restaurant process is a stochastic process closely related to the Dirichlet process that groups sequentially arriving objects into a variable number of classes, such that within each class objects are cyclically ordered. A popular description involves a restaurant, where customers arrive one by one and either sit down next to a randomly chosen customer at one of the existing tables or open a new table. The full state of the process after  $n$  steps is given by a permutation of the  $n$  objects and cannot be represented in sublinear space. In particular, if we only need specific information about a few objects or classes it would be preferable to obtain the answers without simulating the process completely.

A recent line of research [15, 28, 5, 12] attempts to provide access to huge random objects without fully instantiating them. Such *local access implementations* provide answers to a sequence of queries about the random object, following the same distribution as if the object was fully generated. In this paper, we provide a local access implementation for a generalization of the Chinese restaurant process described above. Our implementation can be used to answer any sequence of adaptive queries about class affiliation of objects, number and sizes of classes at any time, position of elements within a class, or founding time of a class. The running time per query is polylogarithmic in the total size of the object, with high probability. Our approach relies on some ideas from the recent local access implementation for preferential attachment trees by Even et al. [12]. Such trees are related to the Chinese restaurant process in the sense that both involve a “rich-get-richer” phenomenon. A novel ingredient in our implementation is to embed the process in continuous time, in which the evolution of the different classes becomes stochastically independent [21]. This independence is used to keep the probabilistic structure manageable even if many queries have already been answered. As similar embeddings are available for a wide range of urn processes [2], we believe that our approach may be applicable more generally. Moreover, local access implementations for birth and death processes that we encounter along the way may be of independent interest.

**2012 ACM Subject Classification** Theory of computation → Generating random combinatorial structures; Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Chinese restaurant process, Dirichlet process, sublinear time algorithm, random recursive tree, random permutation, random partition, Ewens distribution, simulation, local access implementation, continuous time embedding

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.28

**Category** RANDOM

**Funding** *Stefan Walzer*: DFG grant WA 5025/1-1.

## 1 Introduction

Random objects are often used to model how data is generated. Examples include Gaussian mixture models, random graph models such as the preferential attachment model [3], Erdős-Rényi graphs [11] and the stochastic block model [19]. Usually, the process to generate such an object is fairly easy to describe and implement. However, if we think of these objects as



© Peter Mörters, Christian Sohler, and Stefan Walzer;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 28; pp. 28:1–28:18



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

modelling very large data sets it may be time and space consuming to generate an instance of the model. In particular, if we want to evaluate a sampling based algorithm on such a model or we want to study some local properties of the generated object, it would be interesting to provide local access to the object without fully instantiating it immediately. For example, can we easily determine the neighbourhood of a vertex in a webgraph taken from the preferential attachment model without first fully computing the graph? Can we compute the nearest neighbour of a point in a Gaussian mixture model without generating all data points? If we were able to provide consistent and efficient answers to such queries, we could, for example, run sampling algorithms on very large input graphs without fully generating them. Of course, such a local access must provide to all sequences of queries and answers the same distribution as if the object was immediately fully instantiated.

The challenge is that answers to queries must be correlated in the right way, i.e. the distribution of the answer for a query must be a conditional distribution that takes into account all answers given to previous queries. In a Gaussian mixture model, for instance, revealing the location  $x$  of one point makes it more likely that additional points are located close to  $x$ .

**The Chinese restaurant process.** An intuitive description of the *Chinese restaurant process* (CRP) as generalised in [6] involves a restaurant with round tables of unbounded capacity, corresponding to classes of objects, and customers, corresponding to the objects. We imagine that every dish has an objective tastiness and a distribution  $\Phi$  on  $(0, \infty)$  captures how tasty a random dish from the menu is. In every round one customer arrives, and the  $n$ -th customer has  $n$  options. She may sit down next to one of the  $n - 1$  earlier customers and order the same dish as him, or she sits at a new table and orders a random dish from the menu. She makes each choice with a probability proportional to how appealing it is. The appeal of sitting next to customer  $c$  is the tastiness of the dish of  $c$  and the appeal of sitting at a new table is 1. Applications in biology motivate us to speak of *fitness* rather than tastiness in the following. Instead of being assigned to dishes, we may instead assume that fitness values are assigned to the customers themselves or simply to the tables, as all customers at a table always order the same dish.

The CRP is closely related to Dirichlet processes and the Pólya urn model. Its easy iterative definition has contributed to its popularity as a model for clustering in Bayesian statistics, for example for gene expression data [27, 29] or in image analysis [25]. The random partition induced by the CRP is the Ewens distribution that describes the allelic partition of DNA in the infinite alleles models under assumptions of neutrality and no recombination [10, 8].

**Our Contribution.** In this paper we provide a local access implementation of the  $N$ -round CRP and several related processes. The number of steps required to compute the answer to a query is polylogarithmic in  $N$ . An informal version of our main theorem is as follows.

► **Theorem 1** (Main, informal). *Let  $\Phi$  be a distribution on  $(0, \infty)$  with some means of sampling from  $\Phi$ , and let  $N \in \mathbb{N}$ . A local access implementation of the  $N$ -round CRP with parameter  $\Phi$  can be achieved such that any (possibly adaptive) sequence of queries from the list below can be answered in  $\text{polylog}(N)$  time per query whp.*

- Which customer sits right/left of customer  $c$  after  $n \leq N$  rounds?
- What is the fitness of customer  $c$ 's table?
- Who founded customer  $c$ 's table?
- How many customers are at each of the tables after  $n \leq N$  rounds?

A formal version of this theorem can be found in Section 2.2.

## 1.1 Related Work

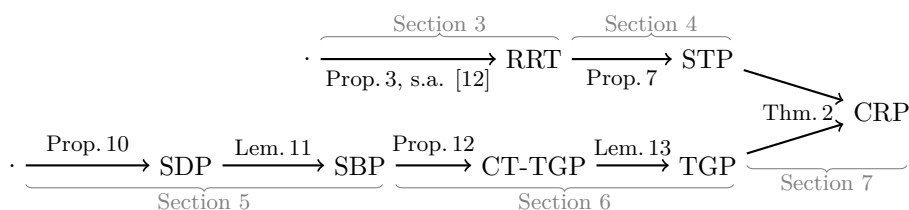
The problem of providing local access to a random object has appeared in several earlier works. Goldreich, Goldwasser and Nussboim [15] define a framework to study implementations of random objects. They assume the random object can be described as a function and require that queries to the function are answered in polylogarithmic time. They discuss *perfect implementations* (which have the same distribution as the original object), *close implementations* (which have approximately the same distribution) and *pseudoimplementations* (which cannot be efficiently distinguished from the original object). They also consider the truthfulness of implementations meaning that if every random object has property  $T$  then also every object in the implementation is required to have property  $T$ . They give several examples of implementations and further examples are found in [28]. Perfect implementations are also known as local access implementations or local access generators. These have been developed for preferential attachment trees by Even et al. [12], for Erdős-Renyi graphs and Dyck paths by Biswas et al. [5], and for random walks by Biswas et al. [4].

A related line of research includes partitioning oracles [17, 23, 22] that provide sublinear time access to a random partition of an input graph. While the partition is also a random object, randomness is used to guarantee that the partition cuts only few edges and that queries can be answered efficiently.

Yet another related direction is local computation algorithms developed by Rubinfeld et al. [30] and Alon et al. [1] that give sublinear time local access to the solution of a computational task. Examples for problems for which local computation algorithms are known include sparse spanning graphs [24], set cover [16], mechanism design [18], clustering [13] and maximum matching [26].

## 1.2 Technical Overview

In our pursuit of a local access implementation of the CRP we develop local access implementations of several related processes that may be of independent interest. An overview is given in Figure 1. Two aspects of the CRP are captured separately. In the following, we outline some difficulties in dealing with them as well as how we overcome these difficulties.



■ **Figure 1** Random processes that capture aspects of the CRP. The picture indicates the theorem, proposition or lemma that establishes a local access implementation of these processes, the sections where they are found, as well as what each result builds upon.

**The Single Table Process (STP).** The simpler aspect concerns the order of customers at a single table, i.e. we have to answer who sits left or right of whom. Let us ignore other tables for now and simply assume customers  $[n] := \{1, \dots, n\}$  join a table one by one, and each  $i \in \{2, \dots, n\}$  sits down to the right of a customer sampled uniformly at random from  $[i - 1]$ .

The final ordering of the customers is given by a uniformly random cyclic permutation  $\pi : [n] \rightarrow [n]$ , with  $\pi(c)$  being  $c$ 's final right neighbour. A local access implementation of  $\pi$  would be easy to come by but is insufficient for our purposes. This is because we permit queries regarding earlier points in time, i.e. a query may request the right neighbour of customer  $c$  after  $n' \in \{c, \dots, n\}$  steps of the process. Within the sequence  $(\pi(c), \pi^2(c), \pi^3(c), \dots)$  of customers towards the right of  $c$  in the final ordering, the query asks for the first element  $c'$  with  $c' \leq n'$ , i.e. the first customer who was already present at time  $n'$ . Doing this naively by generating the sequence may be too costly.

To allow computing  $c'$  quickly, we consider the **random recursive tree** (RRT), which is a rooted tree  $T$  with vertex set  $[n]$  generated as customers arrive. We begin with just customer 1 as a root vertex. When customer  $c \geq 2$  arrives and takes her seat to the right of customer  $p$ , then  $c$  is prepended to the list of children of vertex  $p$  in  $T$ . As it turns out, a depth first search of  $T$  then visits the vertices in the order  $(1, \pi(1), \dots, \pi^{n-1}(1))$ , reflecting the final ordering of customers. Since  $T$  has logarithmic depth and logarithmic maximum degree whp we can determine the predecessor and successor of a given vertex in the DFS ordering by issuing a logarithmic number of neighbourhood queries to  $T$  whp. Moreover – and crucially – information about the order of customers at any time step  $n' < n$  is naturally contained in  $T$  within the subtree  $T'$  induced by vertex set  $[n']$ . In other words, to find the right neighbour of a vertex  $c$  at time  $n'$  we find its DFS successor in  $T'$ .

We therefore have to implement local access to a random recursive tree  $T$ . This has already been achieved by Even et al [12]. We simplify their implementation in two ways. Firstly, we always reveal the neighbourhood of any requested vertex in its entirety instead of revealing only the “next child” in its adjacency list. This simplifies the structure of the residual probability space without negatively impacting running time. Secondly, we employ what we call the “harmonic sampling trick” which allows sampling a set  $X \subseteq [N]$  in  $\mathcal{O}(|X|)$  steps that contains each  $i \in [N]$  independently with probability  $1/i$ . The algorithm is quite simple: Sample  $x$  uniformly from  $[N]$ , then recursively sample a set  $X' \subseteq [x-1]$  that contains each  $i \in [x-1]$  independently with probability  $1/i$ , and return  $X = \{x\} \cup X'$ . The intuition why the trick is useful is that vertex 1 in  $T$  has vertices  $2, 3, 4, \dots$  as children with probability  $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \dots$ , respectively. When combined with rejection sampling, the trick remains useful also when vertices other than 1 are concerned and when partial information about  $T$  has been revealed.

**The Table Growth Process (TGP).** The more difficult aspect concerns the number of customers at each of the tables over time. The growth of a single table  $T$  is correlated with the other tables. This is true for the obvious reason that after  $n$  steps all table sizes add to  $n$ , but also because the creation of a new table with high fitness can significantly hamper the ability of other tables to attract customers in the future. Even in the absence of table creations and with identical fitness values a considerable challenge remains: Assume we have revealed that at times  $n_1$  and  $n_2$  with  $n_1 < n_2$  there are  $\ell$  tables with fitness value 1 each and that the numbers of customers at these tables has risen from  $a_1, a_2, \dots, a_\ell$  to  $a_1 + d_1, a_2 + d_2, \dots, a_\ell + d_\ell$ . Assume further that the customer counts  $a_1 + d'_1, a_2 + d'_2, \dots, a_\ell + d'_\ell$  at an intermediate time  $n \in \{n_1, \dots, n_2\}$  is requested. Then  $(d'_1, \dots, d'_\ell)$  has a *multivariate hypergeometric distribution*, i.e.  $d'_i$  is the number of balls of colour  $i$  when drawing  $n - n_1$  balls from an urn without replacement where the urn contains  $d_j$  balls of colour  $j$  for each  $j \in [\ell]$ . The intimidating prospect of having to efficiently sample from such a distribution prompted us to choose a different path. While we have since been made aware that fast approximate sampling from multivariate hypergeometric distributions has been achieved in [5, Appendix D], we still do not know how table creations and fitness values could be incorporated into such an approach.

We dodge these complications by adopting a continuous time view of the table growth process where each table gains customers *independently* of all other tables. More precisely, every table gains additional customers with a *rate* proportional to both its fitness and size, and new tables are founded with a rate of 1. Properties of the exponential distribution ensure that the next customer to arrive will always be destined for a specific (new or old) table with a probability proportional to that table's rate, as required. A complication is that the times at which customers arrive are now *random*. In particular, if we are interested in the state of the process after  $n$  customers have arrived, we first have to locate a corresponding point in time using binary search in the time dimension.

In our continuous time table growth process (CT-TGP), the growth of a single table with fitness 1 is governed by a **simple birth process** (SBP, also called Yule process) that begins with a single element and from then on gains elements with a rate equal to its size. A fitness  $\neq 1$  merely amounts to rescaling time. A SBP can be seen as a **simple death process** (SDP) played in reverse, where a SDP starts with some number  $N$  of elements and each element dies independently of the others with a rate of 1, i.e. it has an  $\text{Exp}(1)$ -distributed lifetime.

Our most fundamental problem is therefore that of providing a local access implementation to the SDP, which can answer for a given time  $t \in \mathbb{R}_{\geq 0}$  how many elements die until time  $t$ . The rough idea for dealing with the first query is as follows. Let  $m$  be the median of the exponential distribution. The number of elements dying at a time in  $[0, m]$  has distribution  $N' \sim \text{Bin}(N, \frac{1}{2})$  and if  $t < m$  we need only obtain further information on those  $N'$  elements to answer the query. In that case, let  $0 < m' < m$  be the median of the exponential distribution when conditioned on attaining values in  $[0, m]$ . The number  $N''$  of elements dying within  $[0, m']$  has distribution  $N'' \sim \text{Bin}(N', \frac{1}{2})$  and depending on whether  $t \leq m'$  or  $t > m'$  we would have to continue worrying about the precise death times of only  $N''$  or  $N' - N''$  elements – roughly half in expectation. To fully answer the query at hand and further queries like it we lazily construct a binary tree where inner nodes record numbers of elements (such as  $N'$  and  $N''$ ) with death times falling within respective ranges. The elements themselves are represented in leafs which are at depth  $\mathcal{O}(\log N)$  whp. To sample the necessary Binomial random variables in  $\mathcal{O}(\log N)$  time whp we use a result by [7].

## 2 Preliminaries and formal statement of main result

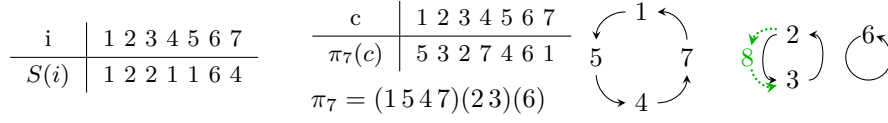
### 2.1 The Chinese restaurant process with table fitness

The variant of the *Chinese restaurant process* considered in this paper was proposed in [6] and is parametrised by a distribution  $\Phi$  on  $(0, \infty)$ . An outcome of the CRP is given by a (random) mapping

$$S: \mathbb{N} \rightarrow \mathbb{N} \text{ such that } S(n) \in [n]$$

where  $n := \{1, \dots, n\}$ . It can be interpreted as follows. In a Chinese restaurant an unbounded number of circular tables are initially empty and customers arrive one by one. If  $S(i) < i$  then the  $i$ -th customer takes her place at an existing table directly to the right of customer  $S(i)$ . If  $S(i) = i$  then customer  $i$  is understood to become her own right neighbour by becoming the *founder* of a new table (customer 1 is necessarily a founder). This process gives rise to a sequence  $(\pi_n)_{n \in \mathbb{N}}$  of permutations  $\pi_n$  in the symmetric group  $\text{Sym}_n$ , where  $\pi_n$  indicates for a customer  $i \in [n]$  the customer  $\pi_n(i)$  sitting directly to the right of customer  $i$  at time  $n$ . An example is given in Figure 2. The figure also visualises the *cycles* of  $\pi_n$  as the connected components in the graph  $([n], \{(i, \pi(i)) \mid i \in [n]\})$  and showcases the well-subscribed cycle notation.





■ **Figure 2** Some values for  $S$ , the permutation  $\pi_7$  they give rise to, the cycle representation of  $\pi_7$  and a visualisation. In the interpretation of the CRP the elements 1, 2 and 6 are founders and thereby representatives of three tables. In green we show the effect on the second table if  $S(8) = 2$ .

The distribution of  $S$  involves fitness values  $f_1, f_2, \dots \in (0, \infty)$  assigned to customers (say the tastiness of their dish). Formally, when  $n - 1$  customers have entered the restaurant, and  $S(1), \dots, S(n - 1)$ ,  $f_1, \dots, f_{n-1}$  and  $\pi_1, \dots, \pi_{n-1}$  are given, the  $n$ -th customer enters.

- She either chooses  $S(n) = i \in [n - 1]$  at random with probability

$$\mathbb{P}(S(n) = i) = \frac{f_i}{1 + \sum_{j=1}^{n-1} f_j}.$$

In this case  $f_n = f_i$  (customer  $n$  orders the same dish as her neighbour) and  $\pi_n(i) = n$ ,  $\pi_n(n) = \pi_{n-1}(i)$  and  $\pi_n(k) = \pi_{n-1}(k)$  for  $k \in [n] \setminus \{i, n\}$ .

- Otherwise she chooses  $S(n) = n$  with probability

$$\mathbb{P}(S(n) = n) = \frac{1}{1 + \sum_{j=1}^{n-1} f_j}.$$

In this case customer  $n$  occupies a new table. We sample  $f_n \sim \Phi$  and let  $\pi_n(n) = n$  and  $\pi_n(k) = \pi_{n-1}(k)$  for  $k \in [n - 1]$ . This always occurs for customer 1.

Further quantities of interest are easily derived from  $S$  and  $(\pi_n)$ . The set  $\mathcal{F}$  of founders is precisely the set of fixed points of  $S$ , i.e.

$$\mathcal{F} = \{n \in \mathbb{N} : S(n) = n\}.$$

The founders  $\mathcal{F} = \{n_1 < n_2 < \dots\}$  can act as representatives of their tables, i.e. by the  $k$ -th table we mean the table at which  $n_k$  is placed. The number of non-empty tables at time  $n$  is  $k_n = |\mathcal{F} \cap [n]|$ , which is also the number of cycles in the permutation  $\pi_n$ . The customers at table  $n_k$  at time  $n$  are the elements of the cycle of  $\pi_n$  containing  $n_k$ . To find the customer founder( $n$ ) who founded the table at which customer  $n$  is sitting we can iteratively apply  $S$  starting with the argument  $n$ , until (after at most  $n$  iterations) we find a fixed point.

**Related Objects.** The classical single type Chinese restaurant process has only a single parameter  $\theta > 0$ , which is a weight for the probability of founding a new table. It arises as a special case of our variant when  $\Phi$  is the trivial distribution putting all probability mass on the value  $\theta^{-1}$ . Several aspects of the CRP are known under different names.

- If  $a_i$  denotes the number of tables of size  $i$  after  $n$  steps of the CRP with parameter  $\theta$  then the distribution of  $(a_1, \dots, a_n)$  is known as the Ewens distribution.
- If we treat all fitness values as though they were 1 and reinterpret  $f_i$  as the dish ordered by customer  $i$  (with all dishes having the same quality) then  $f_1, f_2, \dots$  is the Dirichlet process with base distribution  $\Phi$ .
- If we initialise the CRP with some customers at some of the tables, all of which have the same fitness, and condition on the event that no further tables are created, then the number of customers at the tables over time is a Pólya urn model.

- Call a directed graph with vertex set  $V = [n]$ , where each  $v \in V$  has exactly one outgoing edge towards  $p(v) \leq v$  (loops are allowed) a *recursive forest*. If  $\theta = 1$  then the graph  $T = ([n], \{(i, S(i)) \mid i \in [n]\})$  generated from the outcome  $S$  of the CRP is a uniformly random recursive forest. If we condition on the event  $\{\forall i > 1 : S(i) \neq i\}$  then  $T$  is a uniformly random recursive tree.

## 2.2 Formal statement of main result

In this paper, an event  $E_n$  occurs with high probability (whp) if for any constant  $c > 0$  we have  $1 - \mathbb{P}(E_n) = \mathcal{O}(n^{-c})$ . We allow the constant hidden by  $\mathcal{O}$ -notation to depend on  $c$ . For a random variable  $X_n$  and a function  $g$  we say  $X_n = \mathcal{O}(g(n))$  whp if

$$\forall c > 0 : \exists n_0, C : \forall n \geq n_0 : \mathbb{P}(X_n > C \cdot g(n)) \leq n^{-c},$$

in particular  $C$  may depend on  $c$ . Conveniently, whenever we have a polynomial number of events that individually occur whp, then these events jointly occur whp. We say  $X_n$  is  $\text{polylog}(n)$  whp if  $X_n = \mathcal{O}(\log^b(n))$  whp for some constant  $b > 0$ .

**Local Access Implementation.** Our goal is known under several different names. We loosely follow the terminology of *local access implementation* by [5], which is a *stateful implementation* in the sense of [15] and a *random access generator* in the sense of [12].

Assume we are given a huge random object  $X$  by a distribution  $D$  on a set  $\mathbb{X}$ . A local access implementation of a family  $F_1, \dots, F_q : \mathbb{X} \rightarrow R$  of attributes (random variables) with values in some set  $R$  is a data structure that answers a sequence of (possibly adaptive) queries  $i_1, i_2, \dots$  with a sequence of results  $r_1, r_2, \dots$  such that  $(r_1, r_2, \dots)$  has the same distribution as  $(F_{i_1}(X), F_{i_2}(X), \dots)$ . If, for every  $x \in \mathbb{X}$ , the attributes  $F_1(x), \dots, F_q(x)$  determine the object  $x$  completely we also speak of a local access implementation of  $X$  (via  $F_1, \dots, F_q$ ).

We can now formally state our main result.

► **Theorem 2** (Local Access Implementation of the CRP). *Let  $\Phi$  be a distribution on  $(0, \infty)$  with some means of sampling from  $\Phi$  (e.g. using an oracle), and let  $N \in \mathbb{N}$ . A local access implementation of the  $N$ -round CRP with parameter  $\Phi$  providing access via the attributes listed in Table 1 can be achieved such that each (possibly adaptive) query takes  $\text{polylog}(N)$  time whp.*

Note that the family of attributes grows with  $N$ . For instance, we can query the Chinese restaurant process for  $\pi_n(c)$  for any  $1 \leq c \leq n \leq N$ . In the flat view used above this constitutes a separate attribute for every valid pair  $(n, c)$ . Our implementation must be given a parameter  $N \in \mathbb{N}$  beforehand and then one can query all attributes arising from rounds with index  $n \leq N$ .

**Model of Computation.** Since we use an embedding of the CRP in continuous time, our algorithms involve uniform sampling from  $[0, 1] \subseteq \mathbb{R}$  and arithmetic operations on real numbers as would be allowed in the real RAM model. We suspect that our construction could be adapted for the word RAM model with moderate technical complications regarding the content of Section 6, but we do not pursue such a result.

## 3 Local access implementation of Random Recursive Trees

The *random recursive tree*  $T_N$  is the  $N$ -th element in a random sequence  $(T_n)_{n \in \mathbb{N}}$  where  $T_n$  is a rooted tree with vertex set  $[n]$  and  $T_{n+1}$  arises from  $T_n$  by assigning the new vertex  $n+1$  a single neighbour  $\text{parent}(n+1) \in [n]$  uniformly at random. In this section, we provide a random access generator for  $T_N$  in the following sense.

■ **Table 1** Attributes that can be queried by our local access implementation of the CRP and corresponding parameters.

| attribute                      | parameters               | interpretation                                                                                             |
|--------------------------------|--------------------------|------------------------------------------------------------------------------------------------------------|
| $\pi_n(c)$ and $\pi_n^{-1}(c)$ | $1 \leq c \leq n \leq N$ | customer sitting right and left of customer $c$ after $n$ rounds                                           |
| $f_c$                          | $1 \leq c \leq N$        | fitness of customer $c$ , i.e. weight for the probability with which customers are seated next to $c$ .    |
| founder( $c$ )                 | $1 \leq c \leq N$        | founder of customer $c$ 's table                                                                           |
| tableSizes( $n$ )              | $1 \leq n \leq N$        | number of customers at each of the $k_n$ tables after $n$ rounds in the order in which tables were founded |
| $\mathcal{F} \cap [N]$         | —                        | set of all $k_N$ founders within the first $N$ rounds                                                      |

► **Proposition 3** (Local access implementation of the RRT). *Let  $N \in \mathbb{N}$ . A local access implementation of  $T_N$  that provides for any given vertex  $v \in [N]$  access to all neighbours of  $v$  can be achieved such that each query takes  $\text{polylog}(N)$  time whp.*

A proof of this proposition is implicit in the work of Even et al. [12]. We still provide our own construction, which uses a slightly simplified invariant and exploits what we call the harmonic sampling trick. We begin with two facts that are also stated in [12].

► **Lemma 4.**

- (i) *The maximum degree of  $T_N$  is  $\mathcal{O}(\log(N))$  whp.*
- (ii) *The height of  $T_N$  is  $\mathcal{O}(\log(N))$  whp.*

**Proof.**

- (i) Let  $c_i$  be the number of children of vertex  $i$ . We have  $c_i = \sum_{j=i+1}^N X_j$  where  $X_j$  is the indicator that  $i$  is the parent of  $j$ . These indicators are independent and  $X_j$  is Bernoulli distributed with parameter  $\frac{1}{j-1}$ , hence  $\mathbb{E}[c_i] = \mathcal{O}(\log(N))$ . Moreover,  $c_i$  is *Poisson binomial* distributed and simple Chernoff bounds for this case suffice to show that  $c_i = \mathcal{O}(\log(N))$  whp. A more fine-grained analysis is provided in [9, Thm 1].

- (ii) A proof can be found in [14, Thm 6.32]. ◀

To prove Proposition 3 we need two simple ideas presented in the following two lemmas.

► **Lemma 5** (Harmonic Sampling Trick). *For  $i \in [N]$  let  $X_i \sim \text{Ber}(\frac{1}{i})$  be independent Bernoulli random variables and  $X = \{i \in [N] \mid X_i = 1\}$ . The algorithm *HST* samples  $X$  in  $\mathcal{O}(\log N)$  time whp.*

**Algorithm *HST*( $N \in \mathbb{N}$ ):**

```

 $X \leftarrow \emptyset$
while $N \geq 1$ do
 sample $i \in [N]$ uniformly
 $X \leftarrow X \cup \{i\}$
 $N \leftarrow i - 1$
return X
```

**Proof.** First note that  $\Pr[\max(X) = i] = \Pr[X_i = 1, X_{i+1} = \dots = X_N = 0] = \frac{1}{i} \cdot \frac{i}{i+1} \cdot \dots \cdot \frac{n-1}{n} = \frac{1}{n}$ . Hence  $\max(X)$  is uniformly distributed in  $[N]$  and is correctly sampled in the first iteration of the while-loop. The remaining set  $X \setminus \{\max(X)\} = \{j \in [\max(X) - 1] \mid X_j = 1\}$  can then be sampled using the same method since no information on  $X_1, \dots, X_{\max(X)-1}$  has been revealed.

To bound the running time of HST, we use a connection to random recursive trees. In  $T_{N+1}$ , the parent of a vertex  $v \in [N+1] \setminus \{1\}$  is uniformly distributed in  $[v-1]$ . Therefore the set  $X'$  of ancestors of vertex  $N+1$  in  $T_{N+1}$  can be sampled using HST and  $X'$  therefore has the same distribution as  $X$ . We can now use Lemma 4 (ii) to conclude that  $X$  has size  $\mathcal{O}(\log N)$  whp and thus HST runs in time  $\mathcal{O}(\log N)$  whp.  $\blacktriangleleft$

► **Lemma 6 (Auxiliary Set Data Structure).** *There is a data structure for representing a dynamic set  $Q \subseteq [N]$  (initialised with  $Q = \emptyset$ ) and its complement  $\bar{Q} := [N] \setminus Q$  that uses  $\mathcal{O}((|Q|+1)\log N)$  bits and supports the following operations in  $\mathcal{O}(\log N)$  time.*

- (i) *Deciding for  $v \in [N]$  whether  $v \in Q$ .*
- (ii) *Adding an element  $v \in \bar{Q}$  to  $Q$ .*
- (iii) *Computing for  $v \in \bar{Q}$  the number  $\text{rank}_{\bar{Q}}(v) = |\bar{Q} \cap [v]|$ .*
- (iv) *Selecting for  $r \in [N]$  the unique element  $v \in \bar{Q}$  with  $\text{rank}_{\bar{Q}}(v) = r$ , if it exists.*

**Proof.** Simply store  $Q$  in a balanced search tree data structure (e.g. an AVL tree) where subtrees are annotated with the number of elements they contain (see e.g. [31]). Implementing the operations as stated is then straightforward.  $\blacktriangleleft$

**Proof of Proposition 3.** The tree  $T_N$  is determined by a family  $(\text{parent}(v))_{2 \leq v \leq N}$  of independent random variables. Whenever we reveal the neighbourhood of some  $v_0 \in [N]$  then this fully reveals  $\text{parent}(v_0)$  and  $\text{parent}(c_1), \dots, \text{parent}(c_k)$  for the children  $c_1, \dots, c_k$  of  $v_0$ . It also reveals that  $\text{parent}(v) \neq v_0$  for all  $v \in \{v_0 + 1, \dots, N\} \setminus \{c_1, \dots, c_k\}$ . Since every piece of information relates only to one random variable, the family  $(\text{parent}(v))_{2 \leq v \leq N}$  is still independent conditioned on that information (this would not be true if we only revealed  $\deg(v_0)$  for instance).

In contrast to Even et al. [12] we always reveal the entire neighbourhood of a requested vertex  $v_0$ . At any point in time let  $Q$  be the set of vertices that were previously queried and  $\bar{Q} := [N] \setminus Q$ . Our invariant is very simple:

Conditioned on the information we have revealed, the family  $(\text{parent}(v))_{2 \leq v \leq N}$  is still independent. For each  $2 \leq v \leq N$ , either  $\text{parent}(v)$  is known (and not random any more) or uniformly distributed in  $\bar{Q} \cap [v-1]$ .

We use the data structure from Lemma 6 to store  $Q$  and  $\bar{Q}$ . Moreover we store all edges that were previously revealed.

We now explain how a query for  $v \in [N]$  is handled. If  $v \in Q$  then we simply reproduce the answer previously returned for  $v$ . Now assume  $v \in \bar{Q}$ . If  $\text{parent}(v)$  is not yet revealed we have to select it uniformly from  $\bar{Q} \cap [v-1]$ . To this end we compute  $\text{rank}_{\bar{Q}}(v) = |\bar{Q} \cap [v-1]|$ , then sample  $r'$  uniformly from  $[\text{rank}_{\bar{Q}}(v) - 1]$  and select as  $\text{parent}(v)$  the element  $v'$  with  $\text{rank}_{\bar{Q}}(v') = r'$  using the  $\mathcal{O}(\log N)$  time operations from Lemma 6. Some vertices from  $Q$  may already be known children of  $v$ . Moreover, every vertex  $v' \in \{v+1, \dots, N\} \cap \bar{Q}$  where  $\text{parent}(v')$  is not yet known has a probability of  $\frac{1}{\text{rank}_{\bar{Q}}(v') - 1}$  to have  $v$  as its parent. Call such vertices *potential children* of  $v$ . It is important to note that two potential children  $v'_1 < v'_2$  have distinct unit fractions from  $\{\frac{1}{1}, \frac{1}{2}, \dots, \frac{1}{N}\}$  as a probability for having  $v$  as parent because the potential parents of  $v'_2$  include all potential parents of  $v'_1$  and  $v'_1$  itself. We sample  $X \subseteq [N]$  using Lemma 5, which contains each  $i \in [N]$  with probability  $\frac{1}{i}$ . We then make a potential child  $v'$  of  $v$  an actual child of  $v$  if  $\text{rank}(v') - 1 \in X$ . To do this quickly, we iterate over the (small) set  $X$  and check for each  $i \in X$  with a select-query whether a corresponding potential child of  $v$  exists (and ignoring  $i$  if this is not the case). As a last step we add  $v$  to  $Q$  and return  $v$ 's parent and children.

Note that (adaptive) queries can affect the way in which  $T_N$  is sampled but not its distribution. Since a query for  $v \in [N]$  takes time  $\deg(v) \cdot \mathcal{O}(\log n)$  its running time is  $\text{polylog}(N)$  whp by Lemma 4 (i). ◀

#### 4 Local access implementation of the Single Table Process

The *single table process* (STP) is the CRP conditioned on the event  $\{\mathcal{F} = \{1\}\}$ , i.e. no tables are founded after the first. This makes the parameter  $\Phi$  irrelevant. In simple terms, the STP generates a sequence  $(\tau_n)_{n \in \mathbb{N}}$  of cyclic permutations where  $\tau_1 = (1)$  and where  $\tau_{n+1} \in \text{Sym}_{n+1}$  is obtained from  $\tau_n$  by inserting  $n+1$  into the cycle at a uniformly random position. Clearly this makes  $\tau_n$  a uniformly random cyclic permutation. We now implement fast random access to this process in the following sense by exploiting a close correspondence between the STP and the random recursive trees  $(T_n)_{1 \leq n \leq N}$  from the previous section.

► **Proposition 7** (Local access implementation of the STP). *Let  $N \in \mathbb{N}$ . A local access implementation of the  $N$ -round STP that provides access to  $\tau_n(c)$  and  $\tau_n^{-1}(c)$  for any given  $1 \leq c \leq n \leq N$  can be achieved such that queries take  $\text{polylog}(N)$  time whp.*

**Proof of Proposition 7.** While  $T_n$  is defined as an unordered tree, we may assume that the children of a vertex are implicitly decreasingly ordered. The unique depth first search traversal of  $T_n$  produces a vertex list  $L_n = (v_1 = 1, v_2, \dots, v_n)$ . We can associate this list with a cyclic permutation  $\tau'_n$  via

$$\tau'_n(v_i) = v_{i+1} \text{ for } i \in [n-1] \text{ and } \tau'_n(v_n) = v_1.$$

By definition  $T_{n+1}$  is obtained from  $T_n$  by prepending the vertex  $n+1$  to the child list of a uniformly random vertex from  $[n]$ . Thus, the list  $L_{n+1}$  is obtained from  $L_n$  by inserting  $n+1$  after a uniformly random element, and  $\tau'_{n+1}$  is obtained from  $\tau'_n$  by inserting  $n+1$  into a uniformly random position of the cycle. So clearly the sequence  $(\tau'_n)_{1 \leq n \leq N}$  has the same distribution as the sequence  $(\tau_n)_{1 \leq n \leq N}$  from the STP. A query for  $\tau_n(i)$  and  $\tau_n^{-1}(i)$  concerning the STP corresponds to a query for the preorder successor and preorder predecessor of the vertex  $i$  in  $T_n$  (where  $v_1$  is regarded as the successor of  $v_n$  in  $T_n$ ).

By Proposition 3 we have access to adjacencies in  $T_N$  in polylogarithmic time. This also gives us access to adjacencies in the subtree  $T_n$  of  $T_N$  simply by ignoring any incidences to vertices  $v > n$ . To determine preorder successors and predecessors in  $T_n$  as required, we need only follow a path in  $T_n$  which comes with an additional  $\mathcal{O}(\log(N))$  factor by Lemma 4 (ii). ◀

#### 5 Local access implementation of the Simple Birth Process

In this section we provide a random access generator for simple death processes. By reversing time, we then extend the result to (bounded) simple *birth* processes.

**Simple Death Process.** In a *simple death process* (SDP) with  $N$  elements and parameter  $f > 0$ , each element  $i \in [N]$  is assigned a lifetime  $\ell_i \sim \text{Exp}(f)$  independently. The SDP is then  $(Y_t)_{t \geq 0}$  where  $Y_t := |\{i \in [N] \mid \ell_i \geq t\}|$  is the number of elements surviving until time  $t$  and the *duration*  $\tau$  of the process is defined as  $\tau := \inf\{t \geq 0 \mid Y_t = 0\}$ .

Recall four simple facts abouts exponentially distributed random variables.

► **Fact 8.**

- (i)  $\mathbb{P}(X \geq t) = e^{-\lambda \cdot t}$  for  $\lambda > 0$ ,  $t \geq 0$  and  $X \sim \text{Exp}(\lambda)$ .
- (ii)  $\mathbb{P}(X \geq t \mid X \geq t_0) = \mathbb{P}(X \geq t - t_0)$  for  $\lambda > 0$ ,  $t \geq t_0 \geq 0$  and  $X \sim \text{Exp}(\lambda)$ .  
This property is known as the *memorylessness of the exponential distribution*.
- (iii) For  $k \in \mathbb{N}$ ,  $\lambda_1, \dots, \lambda_k > 0$  and independent  $X_i \sim \text{Exp}(\lambda_i)$  for  $i \in [k]$  we have

$$\min_{i \in [k]} X_i \sim \text{Exp}\left(\sum_{i \in [k]} \lambda_i\right) \text{ and } \mathbb{P}(j = \arg \min_{i \in [k]} X_i) = \frac{\lambda_j}{\sum_{i \in [k]} \lambda_i} \text{ for } j \in [k].$$

In the lemma that follows we need to quickly sample binomial random variables and use the following theorem from [7], which even works in the word RAM model.

► **Theorem 9** ([7, Section A.2 Thm 5]). *On a word RAM with word size  $\Omega(\log N)$  it is possible to sample from  $\text{Bin}(N, \frac{1}{2})$  in expected time  $\mathcal{O}(1)$  and whp in time  $\mathcal{O}(\log N)$ .*

► **Proposition 10** (Local access implementation of the SDP). *Let  $N \in \mathbb{N}$  and  $f > 0$ . A local access implementation of a SDP  $(Y_t)_{t \geq 0}$  with parameters  $N$  and  $f$  that provides access to  $\tau$  and  $Y_t$  for given  $t \in [0, \tau]$  can be achieved such that queries take  $\text{polylog}(N)$  time whp.*

**Proof.** If  $(Y_t)_{t \geq 0}$  and  $(Y'_t)_{t \geq 0}$  are SDPs with parameters 1 and  $f$ , respectively, and both with  $N$  elements, then  $(Y_{ft})_{t \geq 0} \stackrel{d}{=} (Y'_t)_{t \geq 0}$ . In other words, the parameter  $f$  only rescales time so we may assume  $f = 1$  without loss of generality. The proof idea is to first describe how an outcome of a SDP can be represented using a binary tree. Second, we show how queries can be answered quickly using the tree. Lastly, we argue that the tree can be lazily generated.

Rather than sampling the lifetimes  $(\ell_i)_{i \in [N]}$  from  $\text{Exp}(1)$ , we sample  $(\ell'_i)_{i \in [N]}$  uniformly from the interval  $(0, 1)$  and define  $\ell_i := -\ln(\ell'_i)$ . This works because for  $t \geq 0$

$$\mathbb{P}(\ell_i > t) = \mathbb{P}(-\ln(\ell'_i) > t) = \mathbb{P}(\ln(\ell'_i) < -t) = \mathbb{P}(\ell'_i < e^{-t}) = e^{-t}, \quad (1)$$

meaning  $\ell_i \sim \text{Exp}(1)$  as desired. We may assume that the set  $L = \{\ell'_1, \dots, \ell'_N\}$  has size  $N$  (values are pairwise distinct). Instead of  $L$  we consider a binary tree  $T = T(L)$  that is defined recursively. The root of  $T$  is *responsible* for  $[0, 1)$ . If a vertex  $v$  of  $T$  is responsible for an interval  $[a, b) \subseteq [0, 1)$  then the subtree rooted at  $v$  stores  $L \cap [a, b)$ . Let  $s(v) := |L \cap [a, b)|$ . If  $s(v) \geq 2$  then  $v$  has two children  $v_1$  and  $v_2$  responsible for  $[a, \frac{a+b}{2})$  and  $[\frac{a+b}{2}, b)$ , respectively. If  $s(v) \leq 1$  then  $v$  is a leaf and if  $s(v) = 1$  then  $v$  is annotated with the unique element from  $L \cap [a, b)$ .

Note that we need not store the values  $a$  and  $b$  because they are implicit in the location of  $v$  in  $T$ . More precisely we have  $a = \frac{i}{2^d}$  and  $b = \frac{i+1}{2^d}$  where  $d$  is the depth of  $v$  in  $T$  and  $i$  is the binary number obtained when encoding the path from the root of  $T$  to  $v$  as a sequence of left (0) and right (1) choices. We do however store the value  $s(v)$  explicitly in  $v$ .

It is quite clear that  $T$  has height  $\mathcal{O}(\log N)$  whp because an inner vertex at depth  $d$  means that two values  $x, y \in L$  fall within the same interval of size  $2^{-d}$ . The expected number of such pairs is  $\mathcal{O}(N^2 2^{-d})$ , which is  $\mathcal{O}(N^{-c})$  for  $d \geq (c+2) \log_2 N$ . It is also clear that  $T$  allows us to answer any query in time proportional to the depth of  $T$ : To compute  $\tau$  it suffices to find the left-most leaf of  $\tau$  (with minimal  $\ell'_i$ , hence corresponding to maximal  $\ell_i$ ). To compute  $Y_t$ , observe that

$$Y_t = |\{i \in [N] \mid \ell_i \geq t\}| = |\{i \in [N] \mid -\ln(\ell'_i) \geq t\}| = |\{i \in [N] \mid \ell'_i \leq e^{-t}\}|.$$

It therefore suffices to locate where  $e^{-t}$  would be in  $T$  and compute, using the values  $s(v)$  along the path, the number of leaves that lie left of the path towards  $e^{-t}$ .

We now describe how  $T$  can be generated on the fly (or more precisely a tree with the same distribution). We generate the children of a vertex only when the vertex is first looked at. Initially there is only the root  $r$  annotated with  $s(r) = N$ . Now assume a vertex is reached for the first time. If  $s(v) = 0$  then there is nothing to do. If  $s(v) \geq 2$  then we create its two children  $v_1$  and  $v_2$ . Since each child is responsible for exactly half the interval that  $v$  is responsible for, we have  $s(v_1) \sim \text{Bin}(s(v), \frac{1}{2})$  and  $s(v_2) = s(v) - s(v_1)$ . We can sample  $s(v_1)$  in  $\mathcal{O}(\log N)$  time whp by Theorem 9. If  $s(v) = 1$ , then we instantiate the single value  $\ell \in L$  that is represented in  $v$  by sampling it uniformly from the interval that  $v$  is responsible for.

During a single query one descending path in  $T$  has to be generated in this way, which takes  $\mathcal{O}(\log(N) \cdot \text{height}(T)) = \mathcal{O}(\log^2(N))$  time whp. Note again that the (possibly adaptive) queries may affect the way in which  $T$  is generated, but not its distribution.  $\blacktriangleleft$

**Simple Birth Process.** Consider a Markov process  $(X_t)_{t \geq 0}$  with  $X_t \in \mathbb{N}_0$  for  $t \in \mathbb{R}_{\geq 0}$  that is monotonic in  $t$ . Let  $t_i := \inf\{t \geq 0 \mid X_t \geq i + 1\}$  be the time when it jumps from  $\leq i$  to  $\geq i + 1$  for  $i \in \mathbb{N}_0$ . If  $X_0 = 1$  and the waiting times  $\Delta_i := t_i - t_{i-1}$  are independent random variables with distribution  $\text{Exp}(i \cdot f)$  for some  $f > 0$ , then  $(X_t)_{t \geq 0}$  is called a *simple birth process* (SBP) or *Yule-process* with parameter  $f$ . Intuitively a SBP is just a SDP that is run in reverse. The following lemma makes this precise.

► **Lemma 11.** *Let  $N \in \mathbb{N}$  and  $f > 0$ . Let  $(X_t)_{t \geq 0}$  be a SBP with parameter  $f$  and  $(Y_t)_{t \geq 0}$  a SDP with parameters  $N$  and  $f$ . Let  $\tau$  denote the random duration of  $(Y_t)_{t \geq 0}$  and  $t_N := \inf\{t \geq 0 \mid X_t > N\}$ . Then*

$$(\tau, (Y_t)_{t > 0}) \stackrel{d}{=} (t_N, (X_{t_N-t})_{t > 0}) \text{ where we define } X_t = 0 \text{ for } t < 0.$$

**Proof.** The amount of time that  $(Y_t)_{t \geq 0}$  lingers at value  $N$  is  $\min_{i \in [N]} \ell_i \sim \text{Exp}(N \cdot f)$  by Fact 8 (iii). Using the memorylessness of the exponential distribution (Fact 8 (ii)) and induction we can say more generally that  $(Y_t)_{t \geq 0}$  lingers at value  $i$  for an  $\text{Exp}(i \cdot f)$ -distributed time, independently for each  $i \in [N]$ . By definition, the same is true for  $(X_t)_{t \geq 0}$ , except in reverse order and for all  $i \in \mathbb{N}$ . The claimed distributional equality follows easily. Note that we had to remove the exceptional case  $t = 0$  due to  $Y_0 = N \neq N + 1 = X_\tau$ .  $\blacktriangleleft$

## 6 Local access implementation of the Table Growth Process

In this section we capture the aspects of the CRP that remain if customers are indistinguishable, i.e. those aspects related only to the sizes of tables.

**The Table Growth Process.** The *table growth process* (TGP) is, like the CRP, parametrised by a distribution  $\Phi$  on  $(0, \infty)$ . For  $n \in \mathbb{N}_0$ , the  $n$ -th state  $S_n$  has the form

$$S_n = \left( (a_n^{(1)}, \dots, a_n^{(k_n)}), (f^{(1)}, \dots, f^{(k_n)}) \right)$$

where  $k_n \leq n$  is a number of tables,  $(a_n^{(1)}, \dots, a_n^{(k_n)})$  are table sizes with  $\sum_{j=1}^{k_n} a_n^{(j)} = n$ , and  $(f^{(1)}, \dots, f^{(k_n)})$  are table fitness values. The process begins with  $k_0 = 0$ , i.e.  $S_0 = (( ), ( ))$ . Given the  $n$ -th state, there are  $k_n + 1$  possibilities for the  $(n + 1)$ -th state: Either for some  $j \in [k_n]$  the  $j$ -th table gains a customer or a new table is created. With the normalisation factor  $Z = 1 + \sum_{j=1}^{k_n} a_n^{(j)} f^{(j)}$  we have: With probability  $a_n^{(j)} f^{(j)} / Z$  the  $j$ -th table grows to size  $a_{n+1}^{(j)} = a_n^{(j)} + 1$  while all other table sizes are unchanged. With the remaining probability  $1/Z$  a new table is founded, meaning that  $k_{n+1} = k_n + 1$ ,  $a_{n+1}^{(k_{n+1})} = 1$  and  $f^{(k_{n+1})}$  is sampled from  $\Phi$ . We prove the following.



► **Proposition 12** (Local access implementation of the TGP). *Let  $\Phi$  be a distribution on  $(0, \infty)$  with some means of sampling from  $\Phi$ , and let  $N \in \mathbb{N}$ . A local access implementation of the  $N$ -round TGP that provides access to  $S_n$  for any given  $1 \leq n \leq N$  can be achieved such that queries take  $\text{polylog}(N)$  time whp.*

Note that this implies that the total number  $k_N$  of tables is at most  $\text{polylog}(N)$  whp simply because a query's output size is a lower bound on its running time.

**Continuous Time TGP.** We now describe the *continuous-time table growth process* (CT-TGP), which is closely linked to the TGP with parameter  $\Phi$ . Let  $\delta_j \sim \text{Exp}(1)$  for  $j \in \mathbb{N}$  be independent random variables, let  $s_j := \sum_{i=1}^j \delta_i$  be the *creation time* of the  $j$ -th table, let  $f^{(j)} \sim \Phi$  be the fitness of the  $j$ -th table and let  $(X_t^{(j)})_{t \geq 0}$  be a SBP with parameter  $f^{(j)}$  that describes the size  $X_{t-s_j}^{(j)}$  of the  $j$ -th table at any time  $t \geq s_j$ . The state  $S'(t)$  of the CT-TGP at time  $t \geq 0$  describes the fitness and sizes of all tables at time  $t$ , formally defined as

$$S'(t) := \left( (X_{t-s_1}^{(1)}, \dots, X_{t-s_{k(t)}}^{(k(t))}), (f^{(1)}, \dots, f^{(k(t))}) \right)$$

where  $k(t) := \max\{j \in \mathbb{N}_0 \mid s_j \leq t\}$  is the number of tables created at time  $t$ . The number  $\text{count}(t) := \sum_{j=1}^{k(t)} X_{t-s_j}^{(j)}$  of customers at time  $t$  is clearly monotone in  $t$ . Let us define the  $n$ -th state  $S'_n$  of the CT-TGP as  $S'_n = S'(t_n)$  where  $t_n := \inf\{t \geq 0 \mid \text{count}(t) \geq n\}$ . With probability 1 all  $t_n$  are distinct and  $\text{count}(t_n) = n$  for all  $n \in \mathbb{N}_0$ .

We now show that the sequence of states in the CT-TGP and in the TGP have the same distribution. This type of argument, known as Athreya-Karlin embedding in the literature on urn processes, is well suited for the study of processes with type or fitness dependent progression rules, see for example [20].

► **Lemma 13.** *Let  $\Phi$  be as in Proposition 12. The sequences  $(S_n)_{n \in \mathbb{N}_0}$  and  $(S'_n)_{n \in \mathbb{N}_0}$  of states traversed by the TGP and the CT-TGP, respectively, have the same distribution.*

**Proof.** The sequences of fitness values are independently sampled from  $\Phi$  in both cases, so it suffices to show that the distribution of the table sizes coincide for the TGP and the CT-TGP, when both processes are conditioned on using any fixed sequence  $(f^{(j)})_{j \in \mathbb{N}}$  of fitness values. We may then suppress fitness values when writing states, i.e. we write  $S_n = (a_n^{(1)}, \dots, a_n^{(k_n)})$  and  $S'_n = (X_{t-s_1}^{(1)}, \dots, X_{t-s_{k(t)}}^{(k(t))})$ . We shall consider the probabilities for all possible state transitions. Let therefore  $Y = (y^{(1)}, \dots, y^{(k)}) \in \mathbb{N}^k$  be any state with  $k \in \mathbb{N}$  and  $\sum_{j \in [k]} y^{(j)} = n$ . Moreover let  $Y_j := (y^{(1)}, \dots, y^{(j)} + 1, \dots, y^{(k)})$  for  $j \in [k]$  and  $Y_0 := (y^{(1)}, \dots, y^{(k)}, 1)$  be possible successor states of  $Y$ . Since both the TGP and the CT-TGP are Markov processes starting with  $S_0 = S'_0 = ()$ , it suffices to show that

$$\mathbb{P}(S_{n+1} = Y_j \mid S_n = Y) = \mathbb{P}(S'_{n+1} = Y_j \mid S'_n = Y) \text{ for all } Y \text{ and all } j = 0, \dots, k. \quad (2)$$

In the TGP we have with  $Z = 1 + \sum_{j \in [k]} y^{(j)} \cdot f^{(j)}$  by definition

$$\mathbb{P}(S_{n+1} = Y_j \mid S_n = Y) = \begin{cases} y^{(j)} \cdot f^{(j)} / Z & \text{for } j \in [k] \\ 1/Z & \text{for } j = 0. \end{cases}$$

For the CT-TGP the situation is similar: Conditioned on  $S'_n = Y$  being the state at time  $t_n$  the delay until the next time  $t^{(j)} > t_n$  when table  $j \in [k]$  grows has distribution  $t^{(j)} - t_n \sim \text{Exp}(y^{(j)} \cdot f^{(j)})$  by the definition of SBPs and by the memorylessness of the exponential distribution (see Fact 8 (ii)). Similarly the delay until the time  $t^{(0)} = s_{k+1}$  when the  $(k+1)$ -th table is founded has distribution  $t^{(0)} - t_n \sim \text{Exp}(1)$ . Whichever of these  $k+1$  events occurs first determines  $S'_{n+1}$ , i.e.

$$\begin{aligned} \mathbb{P}(S'_{n+1} = Y_j \mid S'_n = Y) &= \mathbb{P}(\arg \min_{i \in \{0, \dots, k\}} t^{(i)} - t_n = j \mid S'_n = Y) \\ &\stackrel{\text{Fact 8(iii)}}{=} \begin{cases} y^{(j)} \cdot f^{(j)} / Z & \text{for } j \in [k] \\ 1/Z & \text{for } j = 0. \end{cases} \end{aligned}$$

This establishes Equation (2). ◀

This equivalence is the first crucial step for local access to the TGP.

**Proof of Proposition 12.** We promised a local access implementation of the TGP with parameters  $\Phi$  and  $N$ . By Lemma 13 we may instead give a local access implementation of the CT-TGP that provides access to  $S'_n$  for given  $n \in [N]$ .

We begin by describing the *setup phase* where we determine the parameters of all relevant tables, i.e. those tables that are created before the sum of table sizes exceeds  $N$ . Using the same trick as in Equation (1) on Page 11, we sample  $\delta'_1, \delta'_2, \dots$  uniformly from  $[0, 1]$  and define the delays  $\delta_1, \delta_2, \dots$  between table creations as  $\delta_k = -\ln(\delta'_k)$ , which ensures  $\delta_1, \delta_2, \dots \sim \text{Exp}(1)$ . We can then compute the creation times  $s_1, s_2, \dots$  of tables as  $s_k := \sum_{j=1}^k \delta_j$  and sample the fitness values  $f^{(1)}, f^{(2)}, \dots \sim \Phi$ . For the  $k$ -th table we would instantiate, by definition of the CT-TGP a SBP with parameter  $f^{(k)}$ . However, we are interested in this SBP only until its size reaches  $N$  and may, by Lemma 11, instead instantiate a SDP with parameters  $f^{(k)}$  and  $N + 1$ . Let  $\tau_k$  be the duration of the SDP. The size of table  $k$  for any time  $t \in [s_k, s_k + \tau_k)$  can be assessed by querying the SDP for time  $s_k + \tau_k - t$ .

To decide after the creation of the first  $k$  tables whether a  $(k + 1)$ -th table is needed, we consider its designated birth time  $s_{k+1}$ . If  $s_{k+1} \geq s_j + \tau_j$  for some  $j \in [k]$  then the birth of the  $(k + 1)$ -th table would occur after the  $j$ -th table has grown to size  $N + 1$ , so it is not needed. Otherwise we determine the size of the first  $k$  tables at time  $s_{k+1}$ . Let  $N_k$  be the sum of these sizes. If  $N_k < N$  then the  $(k + 1)$ -th table is created, otherwise it is not needed.

We now argue that at most a polylogarithmic number of tables is created whp. This implies that the setup just described can be carried out in  $\text{polylog}(N)$  time by Proposition 10. We begin with a simple tail bound on the duration  $\tau$  of a SDP with parameters  $f$  and  $N$  (recall that  $\ell_i \sim \text{Exp}(f)$  is the lifetime of the  $i$ -th element):

$$\mathbb{P}(\tau > t) = \mathbb{P}(\max_{i \in [N]} \ell_i > t) \leq N \cdot \mathbb{P}(\ell_1 > t) = N \cdot e^{-ft}. \quad (3)$$

In particular  $\tau = \mathcal{O}(\frac{\log N}{f})$  whp. Since we made no assumptions on  $\Phi$ , we may occasionally see very small fitness values, but since we do not permit  $\Phi$  to depend on  $N$  there is a constant  $\varepsilon > 0$  such that  $\mathbb{P}_{f \sim \Phi}(f \geq \varepsilon) \geq \frac{1}{2}$ . Therefore there is whp at least one table  $j$  with parameter  $f^{(j)} \geq \varepsilon$  among the first  $\mathcal{O}(\log N)$  tables, which guarantees  $\tau_j = \mathcal{O}(\log N)$  whp. The total number of tables is then at most  $j + X$  where  $X$  is the number of tables scheduled for creation within  $[s_j, s_j + \tau_j)$ . Though we have not yet stated it in this way, table creation is governed by a Poisson process with parameter 1 and hence  $X \sim \text{Po}(\tau_j)$ . A simple concentration argument implies that  $X$  is  $\mathcal{O}(\log N)$  whp. Hence, the total number of tables is  $j + X = \mathcal{O}(\log N)$  whp.

We now describe how queries to the CT-TGP are answered. Reporting fitness values is straightforward as all of them have been determined during setup. The challenge is to report for any given  $n \in [N]$  the sizes of each table at some point in time when the sum of these sizes is  $n$ . The idea is quite simple: First determine the number of tables  $k = k(n)$  that exist at such times by iterating over the numbers  $N_0 = 0, N_1, N_2, \dots$  encountered during

setup. Typically,  $k$  is characterised by  $N_{k-1} < n \leq N_k$  and we should focus on the time interval  $I = [s_k, s_{k+1})$  when  $k$  tables exist. In the special case where  $n > N_{k-1}$  and the  $(k+1)$ -th table was never created, we may not have computed  $N_k$  and we use the time interval  $I = [s_k, \min_{j \in [k]} s_j + \tau_j)$ . In both cases we use binary search to find a time point  $t \in I$  where table sizes add up to  $n$  and answer the query with the corresponding state  $S'(t)$  of the CT-TGP.

It should be clear that this approach yields the correct result. Moreover, we have already argued that the relevant number of tables is  $k = \mathcal{O}(\log N)$  whp and by Proposition 10 a single table's size can be determined in  $\text{polylog}(N)$  time whp for any  $t \in I$ . To obtain  $\text{polylog}(N)$  running time overall whp the only thing left to consider is the number of rounds needed by the binary search. For this let  $\hat{f} := \max_{j \in [k]} f^{(j)}$ . Because  $I \subseteq [s_j, s_j + \tau_j]$  for every  $j \in [k]$  and using  $\tau_j = \mathcal{O}(\frac{\log N}{f^{(j)}})$  by Equation (3), the size  $|I|$  of  $I$  satisfies

$$|I| \leq \min_{j \in [k]} \tau_j = \mathcal{O}(\min_{j \in [k]} \frac{\log N}{f^{(j)}}) = \mathcal{O}(\frac{\log N}{\hat{f}}) \text{ whp.}$$

We also need a lower bound on the delays  $\Delta_i = t_{i+1} - t_i$  between the arrival times of two consecutive customers. Recall that in the “flat view” of the CRP  $f_c$  for  $c \in [N]$  is the fitness of customer  $c$ . Given the state at time  $t_i$  we have  $\Delta_i \sim \text{Exp}(r_i)$  where  $r_i = 1 + \sum_{c \in [i]} f_c$ . We say the  $i$ -th delay is  $p$ -long if  $\Delta_i \geq \frac{p}{r_i}$ , which is the case with probability  $\Pr[\Delta_i \geq \frac{p}{r_i}] = \Pr[\text{Exp}(r_i) \geq \frac{p}{r_i}] = e^{-p} \geq 1 - p$ . In particular, for any  $c > 0$  any  $i \in [N-1]$  the  $i$ -th delay is  $N^{-c}$ -long with probability  $1 - N^{-c}$ . By union bound, all  $N-1$  delays are (simultaneously)  $N^{-c-1}$ -long with probability  $1 - N^{-c}$ . In this sense all delays are  $N^{-\mathcal{O}(1)}$ -long whp. Let now  $\hat{\Delta}$  be the smallest delay between any two consecutive arrivals within  $I$ . Assuming that all delays are  $N^{-\mathcal{O}(1)}$ -long we have  $\hat{\Delta} \geq N^{-\mathcal{O}(1)}/\hat{r}$  where  $\hat{r}$  is the largest arrival rate ( $r_i$  above) that occurs within  $I$ . Since the number of customers is bounded by  $N$  and the fitness of any customer is bounded by  $\hat{f}$  we have  $\hat{r} \leq N\hat{f}$ . Hence

$$\hat{\Delta} = \frac{N^{-\mathcal{O}(1)}}{\hat{r}} = \frac{N^{-\mathcal{O}(1)}}{N\hat{f}} = \frac{N^{-\mathcal{O}(1)}}{\hat{f}}.$$

Using our bounds on  $|I|$  and  $\hat{\Delta}$  we conclude that the binary search takes at most

$$\log_2(|I|/\hat{\Delta}) = \log_2\left(\frac{\log N}{N^{-\mathcal{O}(1)}}\right) = \log_2(N^{\mathcal{O}(1)}) = \mathcal{O}(\log N)$$

steps whp as desired.

So far we have not explicitly worried about the fact that queries can be adaptive. Could an attacker, after collecting some information, concoct a specific query that is – while generally fast whp – exceptionally difficult conditioned on the information that has been revealed? Luckily this worry can be dispelled for the reason that the number of distinct attributes is small enough: Our bounds on query time relate only to circumstances regarding the random processes that the attacker does not control. Since these circumstances are favourable whp for any *fixed* query, they are – using our definition of “whp” – also simultaneously favourable for each of the polynomially many possible queries. This concludes the argument. ◀

## 7 Local access implementation of the Chinese Restaurant Process

We are finally ready to prove Theorem 2 based on Propositions 7 and 12.

**Proof of Theorem 2.** The setup is straightforward: To provide a local access implementation of the  $N$ -round CRP with parameter  $\Phi$ , we use our local access implementation the  $N$ -round TGP with parameter  $\Phi$ , which determines how customers are distributed to tables. This

yields the correct distribution because customers that share a table in the CRP process also share the same fitness value, so the way in which fitness is assigned to tables in the TGP is adequate. We query the TGP for  $n = N$  to learn the final table sizes  $(a_N^{(1)}, \dots, a_N^{(k_N)})$  and instantiate  $k_N$  copies of the STP with these sizes as parameters. The  $j$ -th STP provides access to a sequence  $(\tau_n^{(j)})_{1 \leq j \leq a_N^{(j)}}$  of permutations and is responsible for the ordering of the customers at the  $j$ -th table. Note that fitness parameters are not needed since whenever a customer joins a specific table in the CRP all positions at that table are equally likely due to shared fitness values. Any query to the TGP takes  $\text{polylog}(N)$  time whp by Proposition 12 and any query to a STP takes  $\text{polylog}(N)$  time whp by Proposition 7. In particular the described setup takes  $\text{polylog}(N)$  time whp.

To answer CRP queries (see below), we have to translate between the two distinct ways in which customers are referenced by the TGP and the STPs. If the  $c$ -th customer overall joins the  $j$ -th table and is the  $c'$ -th customer at that table, then we call  $(c', j)$  the *local identity* of the customer while  $c$  is her *global identity*. Given the global identity  $c$  of a customer we can obtain her local identity by querying the TGP for round  $c - 1$  and round  $c$ . Let  $j$  be the index of the unique table that either grew in size or was newly created in round  $c$ . Then  $(a_c^{(j)}, j)$  is the local identity of  $c$ . Conversely, given the local identity  $(c', j)$  of a customer, her global identity is the unique number  $c$  with  $a_c^{(j)} = c'$  and  $a_{c-1}^{(j)} = c' - 1$  (assuming  $a_c^{(j)}$  has an implicit value of 0 when  $k_c < j$ ). This number can be determined with binary search in  $\mathcal{O}(\log N)$  queries to the TGP since  $a_c^{(j)}$  is monotonic in  $c$ . Any translation operation takes  $\text{polylog}(N)$  time whp.

We now show how any query to the CRP can be answered by issuing an at most polylogarithmic number of queries to the  $k_N + 1$  processes we have instantiated. In that context, we may freely translate between local and global identities as explained in the previous paragraph.

$\pi_n(c), \pi_n^{-1}(c)$ . Let  $(c', j)$  be the local identity of customer  $c$  and  $n' = a_n^{(j)}$  the size of her table after round  $n$ . The query asks for the global identities of the customers with local identities  $(\tau_{n'}^{(j)}(c'), j)$  and  $((\tau_{n'}^{(j)})^{-1}(c'), j)$ , respectively. We obtain the values  $\tau_{n'}^{(j)}(c')$  and  $(\tau_{n'}^{(j)})^{-1}(c')$  by querying the  $j$ -th STP.

$f_c$ . Simply return  $f^{(j)}$  where  $(c', j)$  is the local identity of customer  $c$ .

**founder** $(c)$ . Let  $(c', j)$  be the local identity of customer  $c$ . Then **founder** $(c)$  is the global identity of the customer with local identity  $(1, j)$ .

$\mathcal{F} \cap [N]$ . This asks for the global identities of the customers with local identities  $(1, j)$  for all  $1 \leq j \leq k_N$ .

**tableSizes** $(n)$ . This query can simply be forwarded to the TGP, the correct answer being  $(a_n^{(1)}, \dots, a_n^{(k_n)})$ . ◀

For reasons already discussed in the proof of Proposition 12, the fact that queries can be chosen adaptively poses no problem.

---

## References

- 1 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1132–1139. SIAM, 2012.
- 2 Krishna B. Athreya and Samuel Karlin. Embedding of urn schemes into continuous time markov branching processes and related limit theorems. *The Annals of Mathematical Statistics*, 39(6):1801–1817, 1968. URL: <http://www.jstor.org/stable/2239282>.

- 3 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- 4 Amartya Shankha Biswas, Edward Pyne, and Ronitt Rubinfeld. Local access to random walks. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS*, volume 215 of *LIPIcs*, pages 24:1–24:22, 2022.
- 5 Amartya Shankha Biswas, Ronitt Rubinfeld, and Anak Yodpinyanee. Local access to huge random objects through partial sampling. In *11th ITCS*, volume 151 of *LIPIcs*, pages 27:1–27:65. Dagstuhl Publishing, 2020. doi:10.4230/LIPIcs.ITCS.2020.27.
- 6 Jakob Björnberg, Cécile Mailler, Peter Mörters, and Daniel Ueltschi. The weighted chinese restaurant process condenses in at most two tables. *In progress.*, 2022.
- 7 Karl Bringmann, Fabian Kuhn, Konstantinos Panagiotou, Ueli Peter, and Henning Thomas. Internal DLA: efficient simulation of a physical growth model – (extended abstract). In *41st ICALP*, volume 8572 of *LNCS*, pages 247–258. Springer, 2014. doi:10.1007/978-3-662-43948-7\_21.
- 8 Harry Crane. The Ubiquitous Ewens Sampling Formula. *Statistical Science*, 31(1):1–19, 2016. doi:10.1214/15-STS529.
- 9 Michael Drmota. *Random Trees: An Interplay between Combinatorics and Probability*. Springer, 1st edition, 2009. doi:10.1007/978-3-211-75357-6.
- 10 Richard Durrett. *Probability models for DNA sequence evolution*. Springer, 2008. URL: [http://www.math.duke.edu/~rtd/Gbook/PM4DNA\\_0317.pdf](http://www.math.duke.edu/~rtd/Gbook/PM4DNA_0317.pdf).
- 11 Pál Erdős and Alfréd Rényi. On random graphs. I. *Publ. Math.*, 6:290–297, 1959.
- 12 Guy Even, Reut Levi, Moti Medina, and Adi Rosén. Sublinear random access generators for preferential attachment graphs. *ACM Trans. Algorithms*, 17(4):28:1–28:26, 2021. doi:10.1145/3464958.
- 13 Grzegorz Gluch, Michael Kapralov, Silvio Lattanzi, Aida Mousavifar, and Christian Sohler. Spectral clustering oracles in sublinear time. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1598–1617. SIAM, 2021.
- 14 William Goh and Eric Schmutz. Limit distribution for the maximum degree of a random recursive tree. *Journal of Computational and Applied Mathematics*, 142(1):61–82, 2002. Probabilistic Methods in Combinatorics and Combinatorial Optimization. doi:10.1016/S0377-0427(01)00460-5.
- 15 Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM J. Comput.*, 39(7):2761–2822, 2010. doi:10.1137/080722771.
- 16 Christoph Grunau, Slobodan Mitrovic, Ronitt Rubinfeld, and Ali Vakilian. Improved local computation algorithm for set cover via sparsification. In Shuchi Chawla, editor, *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2993–3011. SIAM, 2020.
- 17 Avinatan Hassidim, Jonathan A. Kelner, Huy N. Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 22–31, 2009.
- 18 Avinatan Hassidim, Yishay Mansour, and Shai Vardi. Local computation mechanism design. *ACM Trans. Economics and Comput.*, 4(4):21:1–21:24, 2016.
- 19 Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.
- 20 Svante Janson. Functional limit theorems for multitype branching processes and generalized pólya urns. *Stochastic Processes and their Applications*, 110(2):177–245, 2004. doi:10.1016/j.spa.2003.12.002.
- 21 Paul Joyce and Simon Tavaré. Cycles, permutations and the structure of the yule process with immigration. *Stochastic Processes and their Applications*, 25:309–314, 1987. URL: <https://ideas.repec.org/a/eee/spapps/v25y1987ip309-314.html>.

- 22 Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors III:  $\text{poly}(d/\epsilon)$ -time partition oracles for minor-free graph classes. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 257–268. IEEE, 2021.
- 23 Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Trans. Algorithms*, 11(3):24:1–24:13, 2015.
- 24 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. *Algorithmica*, 82(4):747–786, 2020.
- 25 Meng Li and Subhashis Ghosal. Bayesian Multiscale Smoothing of Gaussian Noised Images. *Bayesian Analysis*, 9(3):733–758, 2014. doi:10.1214/14-BA871.
- 26 Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX, and 17th International Workshop, RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pages 260–273. Springer, 2013.
- 27 M. Medvedovic and S. Sivaganesan. Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics*, 18:1194–1206, 2002.
- 28 Moni Naor and Asaf Nussboim. Implementing huge sparse random graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX, and 11th International Workshop, RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 596–608, 2007.
- 29 Zhaohui S. Qin. Clustering microarray gene expression data using weighted Chinese restaurant process. *Bioinformatics*, 22(16):1988–1997, 2006. doi:10.1093/bioinformatics/btl284.
- 30 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In Bernard Chazelle, editor, *Innovations in Computer Science – ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238. Tsinghua University Press, 2011.
- 31 Peter Sanders, Kurt Mehlhorn, Martin Dietzfelbinger, and Roman Dementiev. *Sequential and Parallel Algorithms and Data Structures – The Basic Toolbox*. Springer, 2019. doi:10.1007/978-3-030-25209-0.



# A Fully Adaptive Strategy for Hamiltonian Cycles in the Semi-Random Graph Process

Pu Gao ✉🏠

Department of Combinatorics and Optimization, University of Waterloo, Canada

Calum MacRury ✉🏠

Department of Computer Science, University of Toronto, Canada

Paweł Prałat ✉🏠

Department of Mathematics, Toronto Metropolitan University, Canada

---

## Abstract

The semi-random graph process is a single player game in which the player is initially presented an empty graph on  $n$  vertices. In each round, a vertex  $u$  is presented to the player independently and uniformly at random. The player then adaptively selects a vertex  $v$ , and adds the edge  $uv$  to the graph. For a fixed monotone graph property, the objective of the player is to force the graph to satisfy this property with high probability in as few rounds as possible.

We focus on the problem of constructing a Hamiltonian cycle in as few rounds as possible. In particular, we present an adaptive strategy for the player which achieves it in  $\alpha n$  rounds, where  $\alpha < 2.01678$  is derived from the solution to some system of differential equations. We also show that the player cannot achieve the desired property in less than  $\beta n$  rounds, where  $\beta > 1.26575$ . These results improve the previously best known bounds and, as a result, the gap between the upper and lower bounds is decreased from 1.39162 to 0.75102.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures

**Keywords and phrases** Random graphs and processes, Online adaptive algorithms, Hamiltonian cycles, Differential equation method

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.29

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2205.02350> [10]

**Acknowledgements** The numerical results presented in this paper were obtained using the Julia language [5]. We would like to thank Bogumił Kamiński from SGH Warsaw School of Economics for helping us to implement it. The program is available on-line.<sup>1</sup>

## 1 Introduction and Main Results

### 1.1 Definitions

In this paper, we consider the **semi-random graph process** suggested by Peleg Michaeli, introduced formally in [3], and studied recently in [2, 9, 11, 1, 7, 13] that can be viewed as a “one player game”. The process starts from  $G_0$ , the empty graph on the vertex set  $[n] := \{1, \dots, n\}$  where  $n \geq 1$ . In each **step**  $t$ , a vertex  $u_t$  is chosen uniformly at random from  $[n]$ . Then, the player (who is aware of graph  $G_t$  and vertex  $u_t$ ) must select a vertex  $v_t$  and add the edge  $u_t v_t$  to  $G_t$  to form  $G_{t+1}$ . The goal of the player is to build a (multi)graph satisfying a given property  $\mathcal{P}$  as quickly as possible. It is convenient to refer to  $u_t$  as a

---

<sup>1</sup> <https://math.ryerson.ca/~pralat/research.html#publications>



**square**, and  $v_t$  as a **circle** so every edge in  $G_t$  joins a square with a circle. We say that  $v_t$  is paired to  $u_t$  in step  $t$ . Moreover, we say that vertex  $x \in [n]$  is **covered** by the square  $u_t$  arriving at round  $t$ , provided  $u_t = x$ . The analogous definition extends to the circle  $v_t$ . Equivalently, we may view  $G_t$  as a directed graph where each arc directs from  $u_t$  to  $v_t$ , and thus we may use  $(u_t, v_t)$  to denote the edge added in step  $t$ . For this paper, it is easier to consider squares and circles for counting arguments.

A **strategy**  $\mathcal{S}$  is defined by specifying for each  $n \geq 1$ , a sequence of functions  $(f_t)_{t=1}^\infty$ , where for each  $t \in \mathbb{N}$ ,  $f_t(u_1, v_1, \dots, u_{t-1}, v_{t-1}, u_t)$  is a distribution over  $[n]$  which depends on the vertex  $u_t$ , and the history of the process up until step  $t-1$ . Then,  $v_t$  is chosen according to this distribution. If  $f_t$  is an atomic distribution, then  $v_t$  is determined by  $u_1, v_1, \dots, u_{t-1}, v_{t-1}, u_t$ . We then denote  $(G_i^{\mathcal{S}}(n))_{i=0}^t$  as the sequence of random (multi)graphs obtained by following the strategy  $\mathcal{S}$  for  $t$  rounds; where we shorten  $G_t^{\mathcal{S}}(n)$  to  $G_t$  or  $G_t(n)$  when clear.

Suppose  $\mathcal{P}$  is a monotonely increasing property. Given a strategy  $\mathcal{S}$  and a constant  $0 < q < 1$ , let  $\tau_{\mathcal{P}}(\mathcal{S}, q, n)$  be the minimum  $t \geq 0$  for which  $\mathbb{P}[G_t \in \mathcal{P}] \geq q$ , where  $\tau_{\mathcal{P}}(\mathcal{S}, q, n) := \infty$  if no such  $t$  exists. Define

$$\tau_{\mathcal{P}}(q, n) = \inf_{\mathcal{S}} \tau_{\mathcal{P}}(\mathcal{S}, q, n),$$

where the infimum is over all strategies on  $[n]$ . Observe that for each  $n \geq 1$ , if  $0 \leq q_1 \leq q_2 \leq 1$ , then  $\tau_{\mathcal{P}}(q_1, n) \leq \tau_{\mathcal{P}}(q_2, n)$  as  $\mathcal{P}$  is increasing. Thus, the function  $q \rightarrow \limsup_{n \rightarrow \infty} \tau_{\mathcal{P}}(q, n)$  is non-decreasing, and so the limit,

$$\tau_{\mathcal{P}} := \lim_{q \rightarrow 1^-} \limsup_{n \rightarrow \infty} \frac{\tau_{\mathcal{P}}(q, n)}{n},$$

is guaranteed to exist. The goal is typically to compute upper and lower bounds on  $\tau_{\mathcal{P}}$  for various properties  $\mathcal{P}$ .

## 1.2 Main Results

In this paper, we concentrate on the property of having a Hamiltonian cycle, which we denote by **HAM**. As observed in [3], if  $G_t$  has a Hamiltonian cycle, then  $G_t$  has minimum degree at least 2. Thus,  $\tau_{\text{HAM}} \geq \tau_{\mathcal{P}} = \ln 2 + \ln(1 + \ln 2) \geq 1.21973$ , where  $\mathcal{P}$  corresponds to having minimum degree 2. On the other hand, it is known that the famous 3-out process is Hamiltonian with probability tending to 1 as  $n \rightarrow \infty$  (*a.a.s.*) [6]. As the semi-random process can be coupled with the 3-out process, we get that  $\tau_{\text{HAM}} \leq 3$ . A new upper bound was obtained in [9] in terms of an optimal solution to an optimization problem whose value is believed to be 2.61135 by numerical support. In the same paper, the lower bound mentioned above was shown to not be tight. The lower bound was increased by  $\varepsilon = 10^{-8}$  and so numerically negligible.

The upper bound on  $\tau_{\text{HAM}}$  of 3 obtained by simulating the 3-out process is **non-adaptive**. That is, the strategy does *not* depend on the history of the semi-random process. The above mentioned improvement proposed in [9] uses an adaptive strategy but in a weak sense. The strategy consists of 4 phases, each lasting a linear number of rounds, and the strategy is adjusted *only* at the end of each phase (for example, the player might identify vertices of low degree, and then focus on connecting circles to them during the next phase).

In this paper, we propose a fully adaptive strategy that pays attention to the graph  $G_t$  and the position of  $u_t$  for every single step  $t$ . As expected, such a strategy creates a Hamiltonian cycle substantially faster than our weakly adaptive strategy, and it allows us to improve the upper bound from 2.61135 to 2.01678.

► **Theorem 1.**  $\tau_{\text{HAM}} \leq \alpha \leq 2.01678$ , where  $\alpha$  is derived from a system of differential equations.

Moreover, by investigating some specific structures that are generated by the semi-random process, which guarantee the existence of a large set of families of edges that cannot simultaneously contribute to the construction of a Hamiltonian cycle, we improve the lower bound of  $\ln 2 + \ln(1 + \ln 2) \geq 1.21973$  to 1.26575. The structures we investigate in this work are different from the ones in [9]. We attain a simpler proof than in [9], and a much stronger bound.

► **Theorem 2.** Let  $f(s) = 2 + e^{-3s}(s+1)\left(1 - \frac{s^2}{2} - \frac{s^3}{3} - \frac{s^4}{8}\right) + e^{-2s}\left(2s + \frac{5s^2}{2} + \frac{s^3}{2}\right) - e^{-s}(3+2s)$ , and let  $\beta \approx 1.26575$  be the positive root of  $f(s) - 1 = 0$ . Then,  $\tau_{\text{HAM}} \geq \beta$ .

### 1.3 Previous Results

Let us briefly describe a few known results on the semi-random process. In the very first paper [3], it was shown that the process is general enough to approximate (using suitable strategies) several well-studied random graph models. In the same paper, the process was studied for various natural properties such as having minimum degree  $k \in \mathbb{N}$  or having a fixed graph  $H$  as a subgraph. In particular, it was shown that *a.a.s.* one can construct  $H$  in less than  $n^{(d-1)/d}\omega$  rounds where  $d \geq 2$  is the degeneracy of  $G$  and  $\omega = \omega(n)$  is any function that tends to infinity as  $n \rightarrow \infty$ . This property was recently revisited in [1] where the conjecture from [3] was proven for any graph  $H$ : *a.a.s.* it takes at least  $n^{(d-1)/d}/\omega$  rounds to create  $H$ .

Another property that was studied in the context of semi-random processes is a property of having a perfect matching, which we denote by **PM**. Since the 2-out process has a perfect matching *a.a.s.* [8], we immediately get that  $\tau_{\text{PM}} \leq 2$ . By coupling the semi-random process with another random graph that is known to have a perfect matching *a.a.s.* [12], the bound can be improved to  $1 + 2/e < 1.73576$ . This bound was recently improved by the authors of this paper by investigating another fully adaptive algorithm [11]. The currently best upper bound is  $\tau_{\text{PM}} < 1.20524$ . In the same paper, the lower bound observed in [3] ( $\tau_{\text{PM}} \geq \ln(2) > 0.69314$ ) was improved as well, and now we know that  $\tau_{\text{PM}} > 0.93261$  [11].

Finally, let us discuss what is known about the property of containing a given spanning graph  $H$  as a subgraph. It was asked by Noga Alon whether for any bounded-degree  $H$ , one can construct a copy of  $H$  *a.a.s.* in  $O(n)$  rounds. This question was answered positively in a strong sense in [2], in which it was shown that any graph with maximum degree  $\Delta$  can be constructed *a.a.s.* in  $(3\Delta/2 + o(\Delta))n$  rounds. They also proved that if  $\Delta = \omega(\log(n))$ , then this upper bound improves to  $(\Delta/2 + o(\Delta))n$  rounds. Note that both of these upper bounds are asymptotic in  $\Delta$ . When  $\Delta$  is constant in  $n$ , such as in both the perfect matching and Hamiltonian cycle setting, determining the optimal dependence on  $\Delta$  for the number of rounds needed to construct  $H$  remains open.

## 2 Proof of Theorem 1

### 2.1 Algorithmic Preliminaries

In this section, we introduce some notation as well as the basic ideas used in the design of all of our strategies.

The main ingredient for proving Theorem 1 is to specify a strategy which keeps augmenting or extending a path, until the path becomes Hamiltonian. Then, with a few more steps, the Hamiltonian path can be completed into a Hamiltonian cycle. Let us suppose that after  $t \geq 0$  steps, we have constructed the graph  $G_t$  which contains the path  $P_t$ . Define  $U_t$  to be

the set of vertices *not* in  $P_t$ , which we refer to as the **unsaturated** vertices of  $[n]$ . It will be convenient to denote the (induced) distance between vertices  $x, y \in V(P_t)$  on the path  $P_t$  by  $d_{P_t}(x, y)$ . We also define  $d_{P_t}(x, Q) := \min_{q \in Q} d_{P_t}(x, q)$  for  $x \in V(P_t)$  and  $Q \subseteq V(P_t)$ .

Let us first assume that  $u_{t+1}$  lands in  $U_t$ . In this case, we can clearly extend the path  $P_t$  by an edge by choosing  $v_{t+1}$  to be an endpoint of  $P_t$ . We call such a move a **(greedy) path extension**. Now, suppose that  $u_{t+1}$  lands on a vertex  $x \in P_t$ . In this case, we cannot perform a greedy path extension, however we can still choose  $v_{t+1}$  in a way that will help us extend the path in the future rounds. Specifically, set  $v_{t+1} := r$  for some  $r \in U_t$ , and **colour** the vertex  $x$  as well as the edge  $xr$ . Suppose that in some round  $i > t + 1$ ,  $u_i$  lands on  $y$  next to the coloured vertex  $x$  on  $P_i$  (i.e.,  $d_{P_i}(x, y) = 1$ ). In this case, set  $v_i = r$ . Observe now that we can add  $r$  to the current path by adding the edges  $yr$  and  $xr$  to it, and by removing the edge  $yx$ . Thus, despite  $u_s$  not landing on an unsaturated vertex, we are still able to perform a move which extends its length by one. We call such an operation a **path augmentation**.

## 2.2 Proof Overview

In order to prove Theorem 1, we analyze a strategy which proceeds in three distinct **stages**. In the first stage, we execute **DegreeGreedy**, an algorithm which makes greedy path extensions whenever possible, and otherwise sets up path augmentation operations for future rounds in a degree greedy manner. Specifically,  $v_{t+1}$  is chosen amongst the unsaturated vertices of minimum coloured in-degree. This degree greedy decision is done to minimize the number of coloured vertices which are destroyed when path augmentations and extensions are made in later rounds. This stage lasts for  $N$  **phases**, where  $N$  is any non-negative integer that may be viewed as the parameter of the algorithm (here a phase is a contiguous set of steps within the current stage). For the claimed (numerical) upper bound of Theorem 1,  $N$  is set to 100. Setting smaller values of the parameter  $N$  – in particular, setting  $N = 0$  – yields an algorithm that is easier to analyse. Setting  $N > 100$  can slightly improve the bound in Theorem 1, but the gain is rather insignificant. The second stage starts at some random step  $t_0$  (i.e.  $t_0 - 1$  is the total number of steps in stage one), and we execute **FullyRandomized**, an algorithm which makes greedy path extensions whenever possible, and otherwise chooses  $v_{t+1}$  randomly amongst the unsaturated vertices. We execute **FullyRandomized** until we are left with  $\varepsilon n$  unsaturated vertices, where  $\varepsilon = \varepsilon(n)$  tends to 0 as  $n \rightarrow \infty$  arbitrarily slowly. At this point, we proceed to the final stage where a clean-up algorithm is run, which also uses path augmentations. Using elementary concentration inequalities we prove that a Hamiltonian cycle can be constructed in an additional  $O(\sqrt{\varepsilon}n) = o(n)$  steps.

In Section 2.3, we first describe **FullyRandomized**, as it is easier to state and analyze than **DegreeGreedy**. Moreover, if we take  $N = 0$ , which corresponds to executing **FullyRandomized** from the beginning, then we will be left with a path on all but  $\varepsilon n$  vertices after  $\alpha^* n$  steps where  $\alpha^* \leq 2.07721$ . Our third stage clean-up algorithm from Section 2.4 allows us to complete the Hamiltonian cycle in another  $o(n)$  steps. Thus, Sections 2.3 and 2.4 provide a self-contained proof of an upper bound on  $\tau_{\text{HAM}}$  of  $\alpha^* \leq 2.07721$  (see Theorem 9). Afterwards, in Section 2.5 we formally state and analyze our first stage algorithm. This is the most technical section of the paper, as **DegreeGreedy** makes decisions in a more intelligent manner than **FullyRandomized** which necessitates more random variables in its analysis. By executing these three stages in the aforementioned order, we attain the claimed upper bound of Theorem 1.

## 2.3 A Fully Randomized Algorithm

In order to describe our algorithm, it will be convenient to colour certain edges of  $G_t$  red. This helps us define certain vertices used by our algorithm for path augmentations. Specifically,  $x \in V(P_t)$  is **one-red** provided it is adjacent to precisely one red edge of  $G_t$ . Similarly,  $x \in V(P_t)$  is **two-red**, provided it is adjacent to precisely two red edges of  $G_t$ . We denote the one-red vertices and two-red vertices by  $\mathcal{L}_t^1$  and  $\mathcal{L}_t^2$ , respectively, and refer to  $\mathcal{L}_t := \mathcal{L}_t^1 \cup \mathcal{L}_t^2$  as the **red** vertices of  $G_t$ . By definition,  $\mathcal{L}_t^1$  and  $\mathcal{L}_t^2$  are disjoint. It will also be convenient to maintain a set of **permissible vertices**  $\mathcal{Q}_t \subseteq V(P_t)$  which specifies which uncoloured vertices on the path can be turned red. In order to simplify our analysis, we specify the size of  $\mathcal{Q}_t$  and ensure that it only contains vertices of path distance at least 3 from the red vertices on  $P_t$ . Formally:

- (i)  $|\mathcal{Q}_t| = |V(P_t)| - 5|\mathcal{L}_t|$ .
- (ii) If  $\mathcal{L}_t \neq \emptyset$ , then each  $x \in \mathcal{Q}_t$  satisfies  $d_{P_t}(x, \mathcal{L}_t) \geq 3$ .

When  $\mathcal{L}_t = \emptyset$ , we simply take  $\mathcal{Q}_t = V(P_t)$ . Otherwise, since  $|\{x \in V(P_t) : d_{P_t}(x, \mathcal{L}_t) \leq 2\}| \leq 5|\mathcal{L}_t|$ , we can maintain these properties by initially taking  $\{x \in V(P_t) : d_{P_t}(x, \mathcal{L}_t) \geq 3\}$ , and then (if needed) arbitrarily removing  $|\{x \in V(P_t) : d_{P_t}(x, \mathcal{L}_t) \geq 3\}| - (|V(P_t)| - 5|\mathcal{L}_t|)$  vertices from it.

Upon the arrival of  $u_{t+1}$ , there are four main cases our algorithm must handle. The first two cases involve extending the length of the path, whereas the latter two describe what to do when it is not possible to extend the path in the current round.

1. If  $u_{t+1}$  lands within  $U_t$ , then greedily extend  $P_t$ .
2. If  $u_{t+1}$  lands at path distance one from some  $x \in \mathcal{L}_t$ , then augment  $P_t$  via an arbitrary red edge of  $x$ .
3. If  $u_{t+1}$  lands in  $\mathcal{Q}_t$ , then choose  $v_{t+1}$  u.a.r. amongst  $U_t$ , and colour  $u_{t+1}v_{t+1}$  red. This case creates a one-red vertex.
4. If  $u_{t+1}$  lands in  $\mathcal{L}_t^1$ , then choose  $v_{t+1}$  u.a.r. amongst  $U_t$  and colour  $u_{t+1}v_{t+1}$  red. This case converts a one-red vertex to a two-red vertex.

In all the remaining cases, we choose  $v_{t+1}$  arbitrarily, and interpret the algorithm as *passing* on the round, meaning the edge  $u_t v_t$  will not be used to construct a Hamiltonian cycle. In particular, the algorithm passes rounds in which  $u_{t+1}$  lands at path distance two from some  $x \in \mathcal{L}_t$ . This guarantees that no two red vertices are at distance two from each other and so when  $u_{t+1}$  lands next to a red vertex, this neighbouring red vertex is uniquely identified. Let us say that a red vertex is **well-spaced**, provided it is at distance at least 3 on the path from all other red vertices, and it is *not* an endpoint of  $P_t$ . Observe that each well-spaced red vertex yields precisely two vertices on  $P_t$  where a path augmentation involving  $u_{t+1}$  can occur. By construction, all but at most 2 of the algorithm's red vertices are well-spaced.

We now formally describe step  $t + 1$  of the algorithm when  $u_{t+1}$  is drawn u.a.r. from  $[n]$ . Specifically, we describe how the algorithm chooses  $v_{t+1}$ , how it constructs  $P_{t+1}$ , and how it adjusts the colours of  $G_{t+1}$ , thus updating  $\mathcal{L}_t^1$  and  $\mathcal{L}_t^2$ .

We define the random variables  $X(t) = |V(P_t)|$ ,  $L_1(t) = |\mathcal{L}_t^1|$ ,  $L_2(t) = |\mathcal{L}_t^2|$ , and  $L(t) = |\mathcal{L}_t| = L_1(t) + L_2(t)$ . Note that  $L(t)$  is an auxiliary random variable which we define only for convenience. We use  $\Delta$  to denote the one step changes in our random variables (i.e.,  $\Delta X(t) := X(t+1) - X(t)$ ). Recall that  $t_0$  is the step when **FullyRandomized** is called. Let us first show that our random variables cannot change drastically in one round.

► **Lemma 3** (Boundedness Hypothesis – FullyRandomized). *With probability  $1 - O(n^{-1})$ ,*

$$\max\{|\Delta X(t)|, |\Delta L_1(t)|, |\Delta L_2(t)|\} = O(\log n)$$

for all  $t_0 \leq t \leq 3n$  with  $n - X(t) \geq n/\log n$ .

---

**Algorithm FullyRandomized Step  $t + 1$ .**


---

- 1: **if**  $u_{t+1} \in U_t$  **then** ▷ greedily extend the path.
  - 2:   Let  $v_{t+1}$  be an arbitrarily chosen endpoint of  $P_t$ .
  - 3:   Set  $V(P_{t+1}) = V(P_t) \cup \{u_{t+1}\}$ ,  $E(P_{t+1}) = E(P_t) \cup \{u_{t+1}v_{t+1}\}$ .
  - 4:   Uncolour all of the edges adjacent to  $u_{t+1}$ .
  - 5: **else if**  $d_{P_t}(u_{t+1}, \mathcal{L}_t) = 1$  **then** ▷ path augment via red vertices
  - 6:   Let  $x \in \mathcal{L}_t$  be the (unique) red vertex adjacent to  $u_{t+1}$
  - 7:   Denote  $xr \in E(G_t)$  an arbitrary red edge of  $x$ , and set  $v_{t+1} = r$ , where  $r \in U_t$ .
  - 8:   Set  $V(P_{t+1}) = V(P_t) \cup \{r\}$  and  $E(P_{t+1}) = E(P_t) \cup \{u_{t+1}r, xr\} \setminus \{u_{t+1}x\}$ .
  - 9:   Uncolour all of the edges adjacent to  $r$ .
  - 10: **else if**  $u_{t+1} \in \mathcal{Q}_t \cup \mathcal{L}_t$  **then** ▷ construct red vertices
  - 11:   Choose  $v_{t+1}$  u.a.r. from  $U_t$ .
  - 12:   Colour  $u_{t+1}v_{t+1}$  red. ▷ construct a one-red or two-red vertex
  - 13:   Set  $P_{t+1} = P_t$ .
  - 14: **else** ▷ pass on using edge  $u_{t+1}v_{t+1}$ .
  - 15:   Choose  $v_{t+1}$  arbitrarily from  $[n]$ .
  - 16:   Set  $P_{t+1} = P_t$ .
  - 17: **end if**
  - 18: Update  $\mathcal{Q}_{t+1}$  such that  $|\mathcal{Q}_{t+1}| = |V(P_{t+1})| - 5|\mathcal{L}_{t+1}|$ .
- 

**Proof.** Note that, by design, the path can only increase its length but it cannot absorb more than one vertex in each round. Hence, the desired property clearly holds for the random variable  $X(t)$ . To estimate the maximum change for the random variables  $L_1(t)$  and  $L_2(t)$ , we need to upper bound the number of red edges adjacent to any particular unsaturated vertex  $v$ . Observe that at any step  $t \leq 3n$ , since we have assumed there are at least  $n/\log n$  unsaturated vertices, the number of red edges adjacent to  $v$  is stochastically upper bounded by the binomial random variable  $\text{Bin}(3n, \log n/n)$  with expectation  $3 \log n$ . It follows immediately from Chernoff's bound that with probability  $1 - O(n^{-3})$ , the number of red edges adjacent to  $v$  is  $O(\log n)$ , and so the desired bound holds by union bounding over all  $3n^2$  vertices and steps. ◀

Let us denote  $H_t = (X(i), L_1(i), L_2(i))_{0 \leq i \leq t}$ . Note that  $H_t$  does *not* encompass the entire history of the random process after  $t$  rounds (i.e.,  $G_0, \dots, G_t$ , the first  $t + 1$  graphs constructed by the algorithm). This deferred information exposure permits a tractable analysis of the random positioning of  $v_t$  when  $u_t$  is red. We observe the following expected difference equations.

► **Lemma 4** (Trend Hypothesis – FullyRandomized). *For each  $t \geq t_0$ , if  $n - X(t) \geq n/\log n$ , then*

$$\mathbb{E}[\Delta X(t) \mid H_t] = 1 - \frac{X(t)}{n} + \frac{2L(t)}{n} + O(\log n/n) \quad (1)$$

$$\begin{aligned} \mathbb{E}[\Delta L_1(t) \mid H_t] &= \frac{X(t) - 5L(t)}{n} + \frac{2L_1(t)}{n} \left( \frac{2L_2(t)}{n - X(t)} - \frac{L_1(t)}{n - X(t)} - 1 \right) \\ &\quad + \frac{2L_2(t)}{n} \left( 1 + \frac{2L_2(t)}{n - X(t)} - \frac{L_1(t)}{n - X(t)} \right) - \frac{L_1(t)}{n} \\ &\quad + \left( 1 - \frac{X(t)}{n} \right) \left( \frac{2L_2(t)}{n - X(t)} - \frac{L_1(t)}{n - X(t)} \right) + O(\log n/n) \end{aligned} \quad (2)$$

$$\begin{aligned} \mathbb{E}[\Delta L_2(t) \mid H_t] &= \frac{L_1(t)}{n} - \left(1 - \frac{X(t)}{n}\right) \frac{2L_2(t)}{n - X(t)} - \frac{2L_1(t)}{n} \frac{2L_2(t)}{n - X(t)} \\ &\quad - \frac{2L_2(t)}{n} \left(1 + \frac{2L_2(t)}{n - X(t)}\right) + O(\log n/n). \end{aligned} \quad (3)$$

The proof is obtained by examining how the landing of  $u_t$  affects the random variables under study. For instance, for  $X(t)$ , observe that  $\Delta X(t)$  is 1 when  $u_{t+1}$  lands on an unsaturated vertex, or adjacent to a red vertex; and is 0 otherwise. Combining with the probabilities of the above two events yields (1). The proofs for (2) and (3) are similar and the details can be found in Appendix A.

In order to analyze **FullyRandomized**, we shall employ the differential equation method [15]. This method is commonly used in probabilistic combinatorics to analyze random processes that evolve step by step. The step changes must be small in relation to the entirety of the discrete structure. For instance, in our application, this refers to adding one edge at a time to the graph on  $[n]$  vertices. The method allows us to derive tight bounds on the associated random variables which hold a.a.s. at every step of the random process. We refer the reader to [4] for a gentle introduction to the methodology, and to Theorem 16 of Appendix C for a statement of the method which be sufficient for our purposes. The execution of **FullyRandomized** starts at some random step  $t_0$ , which we will prove is a.a.s. asymptotic to  $s_0 n$  for some constant  $0 \leq s_0 < 1$ . Let  $X(t_0)$  denote the number of vertices on  $P_t$  after the execution of **DegreeGreedy**. We shall prove that there exists some constant  $\hat{x}(s_0)$  such that  $|X(t_0)/n - \hat{x}(s_0)| \leq \lambda$  for some  $\lambda = o(1)$ . If  $N$  is set to 0, then  $t_0 = s_0 = X(0) = \hat{x}(0) = 0$ .

Let us now fix a sufficiently small constant  $\varepsilon > 0$ , and define the bounded domain

$$\mathcal{D}_\varepsilon := \{(s, x, \ell_1, \ell_2) : -1 < s < 3, -1 < x < 1 - \varepsilon, |\ell_1| < 2, |\ell_2| < 2\}.$$

Consider the system of differential equations in variable  $s$  with functions  $x = x(s)$ ,  $\ell_1 = \ell_1(s)$ , and  $\ell_2 = \ell_2(s)$ :

$$x' = 1 - x + 2(\ell_1 + \ell_2) \quad (4)$$

$$\ell_1' = x - 5(\ell_1 + \ell_2) + \ell_1 \left( \frac{2\ell_2 - \ell_1}{1 - x} - 1 \right) + 2\ell_2 \left( 1 + \frac{2\ell_2 - \ell_1}{1 - x} \right) - \ell_1 + 2\ell_2 - \ell_1 \quad (5)$$

$$\ell_2' = \ell_1 - 2\ell_2 - 2\ell_1 \left( \frac{2\ell_2}{1 - x} \right) - 2\ell_2 \left( 1 + \frac{2\ell_2}{1 - x} \right). \quad (6)$$

The right-hand side (r.h.s.) of each of the above equations is Lipchitz on the domain  $\mathcal{D}_\varepsilon$ . Define

$$T_{\mathcal{D}_\varepsilon} = \min\{t \geq 0 : (t/n, X(t)/n, L_1(t)/n, L_2(t)/n) \notin \mathcal{D}_\varepsilon\}.$$

Now, the “Initial Condition” of Theorem 16 is satisfied with values  $(s_0, \hat{x}(s_0), 0, 0)$  and some  $\lambda = o(1)$ . Moreover, the “Trend Hypothesis” and “Boundedness Hypothesis” are satisfied with some  $\delta = O(\log n/n)$ ,  $\beta = O(\log n)$  and  $\gamma = o(n^{-1})$ , by Lemmas 3 and 4. Thus, for every  $\delta > 0$ ,  $X(t) = nx(t/n) + o(n)$ ,  $L_1(t) = n\ell_1(t/n) + o(n)$  and  $L_2(t) = n\ell_2(t/n) + o(n)$  uniformly for all  $t_0 \leq t \leq (\sigma(\varepsilon) - \delta)n$ , where  $x$ ,  $\ell_1$  and  $\ell_2$  are the unique solution to (4)–(6) with initial conditions  $x(s_0) = \hat{x}(s_0)$ ,  $\ell_1(s_0) = \ell_2(s_0) = 0$ , and  $\sigma(\varepsilon)$  is the supremum of  $s$  to which the solution can be extended before reaching the boundary of  $\mathcal{D}_\varepsilon$ . For  $N = 0$ ,  $s_0 = 0$  and the initial conditions are simply  $x(0) = \ell_1(0) = \ell_2(0) = 0$ . This immediately yields the following.

► **Lemma 5** (Concentration of FullyRandomized's Random Variables). *For every  $\delta > 0$ , a.a.s. for all  $t_0 \leq t \leq (\sigma(\varepsilon) - \delta)n$ ,*

$$\max\{|X(t) - x(t/n)n|, |L_1(t) - \ell_1(t/n)n|, |L_2(t) - \ell_2(t/n)n|\} = o(n).$$

As  $\mathcal{D}_\varepsilon \subseteq \mathcal{D}_{\varepsilon'}$  for every  $\varepsilon > \varepsilon'$ ,  $\sigma(\varepsilon)$  is monotonely nondecreasing as  $\varepsilon \rightarrow 0$ . Thus,

$$\alpha^* := \lim_{\varepsilon \rightarrow 0+} \sigma(\varepsilon) \tag{7}$$

exists. It is obvious that  $|L_1(t)/n|$  and  $|L_2(t)/n|$  are both bounded by 1 for all  $t$  and thus, when  $t/n$  approaches  $\alpha^*$ , either  $X(t)/n$  approaches 1 or  $t/n$  approaches 3. Formally, we have the following proposition.

► **Proposition 6.** *For every  $\varepsilon > 0$ , there exists  $\delta > 0$  such that a.a.s. one of the following holds.*

- $X(t) > (1 - \varepsilon)n$  for all  $t \geq (\alpha^* - \delta)n$ ;
- $\alpha^* = 3$ .

The ordinary differential equations (4)–(6) do not have an analytical solution. In both cases  $N = 0$  and  $N = 100$ , numerical solutions show that  $\alpha^* < 2.1$ . (For  $N = 0$ ,  $\alpha^* \approx 2.07721$ .) Thus, by the end of the execution of **FullyRandomized**, there are  $\varepsilon n$  unabsorbed vertices remaining, for some  $\varepsilon = o(1)$ .

## 2.4 A Clean-up Algorithm

Suppose that we are presented a path  $P$  on  $(1 - \varepsilon)n$  vertices of  $[n]$ , where  $0 < \varepsilon = \varepsilon(n) < 1/1000$ . The assumption on  $\varepsilon$  is a mild but convenient assumption. We will apply the argument for  $\varepsilon = o(1)$ . In this section, we provide an algorithm for the semi-random graph process which absorbs the remaining  $\varepsilon n$  vertices into  $P$  to form a Hamiltonian path, after which a Hamiltonian cycle can be constructed. The whole procedure takes  $O(\sqrt{\varepsilon}n + n^{3/4} \log^2 n) = o(n)$  further steps in the semi-random graph process. Moreover, the algorithm is self-contained in that it only uses the edges of  $P$  in its execution.

► **Lemma 7** (Clean-up Algorithm). *Let  $0 < \varepsilon = \varepsilon(n) < 1/1000$ , and suppose that  $P$  is a path on  $(1 - \varepsilon)n$  vertices of  $[n]$ . Then, given  $P$  initially, there exists a strategy for the semi-random graph process which builds a Hamiltonian cycle from  $P$  in  $O(\sqrt{\varepsilon}n + n^{3/4} \log^2 n)$  steps a.a.s.*

► **Remark 8.** The constant hidden in the  $O(\cdot)$  notation does not depend on  $\varepsilon$ . The strategy used in the clean-up algorithm is similar to that in **FullyRandomized** but the analysis is done in a much less accurate way, as we only need to prove an  $o(n)$  bound on the number of steps required to absorb  $\varepsilon n$  vertices. The proof is presented in Appendix A.

By setting  $N = 0$  we immediately get an algorithm which a.a.s. constructs a Hamiltonian cycle in  $\hat{\alpha}n$  steps, where  $\hat{\alpha} \leq 2.07721$ . To obtain the better bound in Theorem 1, we set  $N = 100$ , and the execution of **DegreeGreedy** will be analysed in the next subsection.

► **Theorem 9.**  $\tau_{\text{HAM}} \leq \hat{\alpha} \leq 2.07721$ , where  $\hat{\alpha}$  is defined in (7) with initial conditions for (4)–(6) set by  $x(0) = \ell_1(0) = \ell_2(0) = 0$ .

**Proof.** This follows by Proposition 6, the numerical value of  $\alpha^*$ , and Lemma 7. ◀



## 2.5 A Degree-Greedy Algorithm

Let us suppose that after  $t \geq 0$  steps, we have constructed the graph  $G_t$  which contains the path  $P_t$ . As before, our algorithm uses path augmentations, and we colour the edges of  $G_t$  to help keep track of when these augmentations can be made. We now use two colours, namely red and blue, to distinguish between edges which are added randomly (red) and greedily (blue). Our blue edges will be chosen so as to minimize the number of blue edges destroyed by path augmentations in future rounds.

We say that  $x \in V(P_t)$  is **blue**, provided it is adjacent to a single blue edge of  $G_t$ , and no red edge. Similarly,  $x \in V(P_t)$  is **red**, provided it is adjacent to a single red edge of  $G_t$ , and no blue edge. Finally, we say that  $x \in V(P_t)$  is **magenta (mixed)**, provided it is adjacent to a single red edge, and a single blue red. We denote the blue vertices, red vertices, and magenta (mixed) vertices by  $\mathcal{B}_t, \mathcal{R}_t$  and  $\mathcal{M}_t$ , respectively, and define  $\mathcal{L}_t := \mathcal{B}_t \cup \mathcal{R}_t \cup \mathcal{M}_t$  to be the **coloured** vertices. By definition,  $\mathcal{B}_t, \mathcal{R}_t$  and  $\mathcal{M}_t$  are disjoint. Once again, we denote our unsaturated vertices by  $U_t$ , and also maintain a set of **permissible** vertices  $\mathcal{Q}_t$  which indicate which saturated vertices are allowed to be coloured blue. Specifically, using the same reasoning as before, we ensure the following:

- (i)  $|\mathcal{Q}_t| = |V(P_t)| - 5|\mathcal{L}_t|$ .
- (ii) If  $\mathcal{L}_t \neq \emptyset$ , then each  $x \in \mathcal{Q}_t$  satisfies  $d_{P_t}(x, \mathcal{L}_t) \geq 3$ .

Upon the arrival of  $u_{t+1}$ , there are five main cases our algorithm must handle. The first two cases involve extending the length of the path, whereas the latter three describe what to do when it is not possible to extend the path in the current round.

1. If  $u_{t+1}$  lands within  $U_t$ , then greedily extend  $P_t$ .
2. If  $u_{t+1}$  lands at path distance one from  $x \in \mathcal{L}_t$ , then augment  $P_t$  via a coloured edge of  $x$ , where a blue edge is taken over a red edge if possible.
3. If  $u_{t+1}$  lands in  $\mathcal{Q}_t$ , then choose  $v_{t+1}$  u.a.r. amongst those vertices of  $U_t$  with *minimum* blue degree. The edge  $u_{t+1}v_{t+1}$  is then coloured blue, and a single blue vertex is created.
4. If  $u_{t+1}$  lands in  $\mathcal{R}_t$ , then choose  $v_{t+1}$  u.a.r. amongst those vertices of  $U_t$  with minimum blue degree. The edge  $u_{t+1}v_{t+1}$  is then coloured blue, and a single red vertex is converted to a magenta (mixed) vertex.
5. If  $u_{t+1}$  lands in  $\mathcal{B}_t$ , then choose  $v_{t+1}$  u.a.r. amongst  $U_t$  and colour  $u_{t+1}v_{t+1}$  red. This case converts a blue vertex to a magenta vertex.

In all the remaining cases, we choose  $v_{t+1}$  uniformly at random, and interpret the algorithm as *passing* on the round. As in **FullyRandomized**, we ensure that all of the algorithm's coloured vertices are at path distance at least 3 from each other, and we define a coloured vertex to be **well spaced** in the same way. Note that red vertices are only created when the blue edges of magenta vertices are uncoloured as a side effect of path extensions and augmentations (see lines (4) and (14) below). We now formally describe step  $t + 1$  of the algorithm upon receiving  $u_{t+1}$ :

■ **Algorithm DegreeGreedy** Step  $t + 1$ .

---

```

1: if $u_{t+1} \in U_t$ then ▷ greedily extend the path
2: Let v_{t+1} be an arbitrarily chosen endpoint of P_t .
3: Set $V(P_{t+1}) = V(P_t) \cup \{u_{t+1}\}$, $E(P_{t+1}) = E(P_t) \cup \{u_{t+1}v_{t+1}\}$.
4: Uncolour all of the edges adjacent to u_{t+1} .
5: else if $d(u_{t+1}, \mathcal{L}_t) = 1$ then ▷ path augment via coloured vertices
6: Let $x \in \mathcal{L}_t$ be the (unique) coloured vertex adjacent to u_{t+1}
7: if x is red then
8: Denote $xy \in E(G_t)$ the red edge of x , where $y \in U_t$.
9: else ▷ x is blue or magenta
10: Denote $xy \in E(G_t)$ the blue edge of x , where $y \in U_t$.
11: end if
12: Set $v_{t+1} = y$.
13: Set $V(P_{t+1}) = V(P_t) \cup \{y\}$ and $E(P_{t+1}) = E(P_t) \cup \{u_{t+1}y, xy\} \setminus \{u_{t+1}x\}$.
14: Uncolour all of the edges adjacent to y .
15: else if $u_{t+1} \in \mathcal{Q}_t \cup \mathcal{R}_t$ then ▷ construct coloured vertices
16: Choose v_{t+1} u.a.r. from the vertices of U_t of minimum blue degree.
17: Colour $u_{t+1}v_{t+1}$ blue. ▷ create a blue or magenta vertex
18: Set $P_{t+1} = P_t$.
19: else if $u_{t+1} \in \mathcal{B}_t$ then
20: Choose v_{t+1} u.a.r. from U_t .
21: Colour the edge $u_{t+1}v_{t+1}$ red. ▷ create a magenta vertex
22: Set $P_{t+1} = P_t$.
23: else ▷ pass on using edge $u_{t+1}v_{t+1}$
24: Choose v_{t+1} u.a.r. from $[n]$.
25: Set $P_{t+1} = P_t$.
26: end if
27: Update \mathcal{Q}_{t+1} such that $|\mathcal{Q}_{t+1}| = |V(P_{t+1})| - 5|\mathcal{L}_{t+1}|$. ▷ update permissible vertices

```

---

For each  $t \geq 0$ , define the random variables  $X(t) := |V(P_t)|$ ,  $B(t) := |\mathcal{B}_t|$ ,  $R(t) := |\mathcal{R}_t|$ ,  $M(t) := |\mathcal{M}_t|$  and  $L(t) := |\mathcal{L}_t| = B(t) + R(t) + M(t)$ . For each  $q \geq 0$  and  $t \geq 0$ , define  $D_q(t)$  to be the number of unsaturated vertices adjacent to precisely  $q$  blue edges. We define the stopping time  $\tau_q$  to be the smallest  $t \geq 0$  such that  $D_j(t) = 0$  for all  $j < q$ , and  $D_q(t) > 0$ . It is obvious that  $\tau_q$  is well-defined and is non-decreasing in  $q$ . By definition,  $\tau_0 = 0$ . Let us refer to **phase**  $q$  as those  $\tau_{q-1} \leq t < \tau_q$ . Observe that during phase  $q$ , each unsaturated vertex has blue degree  $q - 1$  or  $q$ .

Suppose that  $\tau_{q-1} \leq t < \tau_q$ . It will be convenient to denote  $D(t) := D_{q-1}(t)$ . Given  $k_1, k_2 \geq 0$ , we say that  $y \in U_t$  is of **type**  $(k_1, k_2)$ , provided it is adjacent to  $k_1$  blue edges within  $\mathcal{B}_t$  and  $k_2$  blue edges within  $\mathcal{M}_t$ . Similarly,  $x \in \mathcal{B}_t \cup \mathcal{M}_t$  is of type  $(k_1, k_2)$ , provided its (unique) *blue* edge connects to an unsaturated vertex of type  $(k_1, k_2)$ . We denote the number of unsaturated vertices of type  $(k_1, k_2)$  by  $C_{k_1, k_2}(t)$ , the blue vertices of type  $(k_1, k_2)$  by  $B_{k_1, k_2}(t)$ , and the magenta (mixed) vertices of type  $(k_1, k_2)$  by  $M_{k_1, k_2}(t)$ . Observe that  $B_{k_1, k_2}(t) = k_1 \cdot C_{k_1, k_2}(t)$  and  $M_{k_1, k_2}(t) = k_2 \cdot C_{k_1, k_2}(t)$ . Moreover,  $D_j(t) = \sum_{\substack{k_1, k_2: \\ k_1 + k_2 = j}} C_{k_1, k_2}(t)$ .

In Section 3, we inductively define the functions  $x, r$  and  $c_{k_1, k_2}$  for  $k_1 + k_2 \geq 0$ , as well as a constant  $\sigma_q \geq 0$ , such that the following lemma holds:

► **Lemma 10.** *A.a.s.  $\tau_q \sim \sigma_q n$  for every  $0 \leq q \leq N$ .<sup>2</sup> Moreover, at step  $\tau_q$ , a.a.s.*

$$\begin{aligned} X(\tau_q) &\sim x(\sigma_q)n, & R(\tau_q) &\sim r(\sigma_q)n, \\ C_{k_1, k_2}(\tau_q) &\sim c_{k_1, k_2}(\sigma_q)n & \text{for all } (k_1, k_2) \text{ where } k_1 + k_2 = q. \end{aligned}$$

Although the method in the proof of Lemma 10 is similar to that of Lemmas 3, 4, 5 and Proposition 6, the analysis is much more intricate and involved. Before proving Lemma 10, we explain how we use it to prove Theorem 16.

**Proof of Theorem 1.** Set  $N = 100$ . By Lemma 10, the execution of **DegreeGreedy** ends at some step  $t_0 \sim \sigma_N n$ . Moreover,  $X(t_0) \sim x(\sigma_N)n$ . Numerical computation shows that  $\sigma_N \approx 2.00189$ . Next, the algorithm executes **FullyRandomized**. Let  $\alpha^*$  be as defined in (7) where the initial conditions to the differential equations (4)–(6) are set by  $s_0 = \sigma_N$ ,  $x(s_0) = x(\sigma_N) \approx 0.99991$ , and  $\ell_1(s_0) = \ell_2(s_0) = 0$ . Numerical computations show that  $\alpha^* \approx 2.01678$ . By Proposition 6 and the fact that  $\alpha^* < 3$ , the execution of the first two stages (**DegreeGreedy** and **FullyRandomized**) finishes at some step  $(\alpha^* + o(1))n$ , and the number of unsaturated vertices remaining is  $o(n)$ . Finally, the clean-up algorithm constructs a Hamiltonian cycle with an additional  $o(n)$  steps by Lemma 7. The theorem follows. ◀

### 3 Proving Lemma 10

We once again must first argue that our random variables cannot change drastically in one round during phase  $q$ .

► **Lemma 11** (Lipschitz Condition – **DegreeGreedy**).

*If  $|\Delta C(t)| := \max_{\substack{k_1, k_2 \in \mathbb{N} \cup \{0\}: \\ k_1 + k_2 \in \{q-1, q\}}} |\Delta C_{k_1, k_2}(t)|$ , then with probability  $1 - O(n^{-1})$ ,*

$$\max\{|\Delta X(t)|, |\Delta C(t)|, |\Delta R(t)|\} = O(\log n)$$

*for all  $\tau_{q-1} \leq t < \tau_q$  with  $n - X(t) = \Omega(n)$ .*

**Proof.** Since  $q \leq N$  is a constant which does not depend on  $n$ , we can apply the same argument to bound the red edges of each  $\Delta C_{k_1, k_2}(t)$  as in Lemma 3, and then union bound over all  $k_1, k_2 \geq 0$  such that  $k_1 + k_2 \in \{q-1, q\}$ . ◀

We now state the conditional expected differences of our random variables. For space considerations, we defer their derivations to the full version of the paper [10].

Let  $H_t$  denote the history of the above random variables during the first  $t$  rounds. where we assume that  $\tau_{q-1} \leq t < \tau_q$  is such that  $n - X(t) = \Omega(n)$ . Firstly, observe that once again:

$$\mathbb{E}[\Delta X(t) \mid H_t] = 1 - \frac{X(t)}{n} + \frac{2L(t)}{n} + O(1/n) \quad (8)$$

We now consider  $\Delta R(t)$ :

$$\begin{aligned} \mathbb{E}[\Delta R(t) \mid H_t] &= \frac{M(t)}{n} - \frac{R(t)}{n} - \frac{2(B(t) + M(t))}{n} \frac{R(t)}{(n - X(t))} \\ &\quad + \sum_{\substack{j, h: \\ j+h \in \{q-1, q\}}} \frac{2h(M_{j, h}(t) + B_{j, h}(t))}{n} \\ &\quad - \frac{2R(t)}{n} \left( 1 + \frac{R(t)}{n - X(t)} - \frac{M(t)}{n - X(t)} \right) - \frac{R(t)}{n} + O(1/n) \end{aligned} \quad (9)$$

<sup>2</sup> For functions  $f = f(n)$  and  $g = g(n)$ ,  $f \sim g$  is shorthand for  $f = (1 + o(1))g$ .

## 29:12 Hamiltonian Cycles in the Semi-Random Graph Process

Consider  $\Delta C_{k_1, k_2}(t)$  and first assume that  $k_1 + k_2 = q - 1$ :

$$\begin{aligned} \mathbb{E}[\Delta C_{k_1, k_2}(t) \mid H_t] &= \frac{M_{k_1-1, k_2+1}(t)}{n} \cdot \mathbf{1}_{k_1 > 0} - \frac{C_{k_1, k_2}(t)}{n} - \frac{M_{k_1, k_2}(t)}{n} \\ &\quad + \frac{2(B(t) + M(t))}{n} \left( \frac{M_{k_1-1, k_2+1}(t)}{n - X(t)} \cdot \mathbf{1}_{k_1 > 0} - \frac{M_{k_1, k_2}(t)}{n - X(t)} \right) \\ &\quad - \frac{2(B_{k_1, k_2}(t) + M_{k_1, k_2}(t))}{n} \\ &\quad + \frac{2R(t)}{n} \left( \frac{M_{k_1-1, k_2+1}(t)}{n - X(t)} \cdot \mathbf{1}_{k_1 > 0} - \frac{M_{k_1, k_2}(t)}{n - X(t)} - \frac{C_{k_1, k_2}(t)}{n - X(t)} \right) \\ &\quad - \frac{(X(t) - 5L(t))}{n} \frac{C_{k_1, k_2}(t)}{D(t)} \\ &\quad + \frac{B_{k_1+1, k_2-1}(t)}{n} \cdot \mathbf{1}_{k_2 > 0} - \frac{R(t)}{n} \frac{C_{k_1, k_2}(t)}{D(t)} - \frac{B_{k_1, k_2}(t)}{n} + O(1/n) \end{aligned} \quad (10)$$

When  $k_1 + k_2 = q$ , two terms from the above expression are modified slightly, and have their signs reversed:

$$\begin{aligned} \mathbb{E}[\Delta C_{k_1, k_2}(t) \mid H_t] &= \frac{M_{k_1-1, k_2+1}(t)}{n} \cdot \mathbf{1}_{k_1 > 0} - \frac{C_{k_1, k_2}(t)}{n} - \frac{M_{k_1, k_2}(t)}{n} \\ &\quad + \frac{2(B(t) + M(t))}{n} \left( \frac{M_{k_1-1, k_2+1}(t)}{n - X(t)} \cdot \mathbf{1}_{k_1 > 0} - \frac{M_{k_1, k_2}(t)}{n - X(t)} \right) \\ &\quad - \frac{2(B_{k_1, k_2}(t) + M_{k_1, k_2}(t))}{n} \\ &\quad + \frac{2R(t)}{n} \left( \frac{M_{k_1-1, k_2+1}(t)}{n - X(t)} \cdot \mathbf{1}_{k_1 > 0} - \frac{M_{k_1, k_2}(t)}{n - X(t)} - \frac{C_{k_1, k_2}(t)}{n - X(t)} \right) \\ &\quad + \frac{(X(t) - 5L(t))}{n} \frac{C_{k_1-1, k_2}(t)}{D(t)} \\ &\quad + \frac{B_{k_1+1, k_2-1}(t)}{n} \cdot \mathbf{1}_{k_2 > 0} + \frac{R(t)}{n} \frac{C_{k_1, k_2-1}(t)}{D(t)} - \frac{B_{k_1, k_2}(t)}{n} + O(1/n) \end{aligned} \quad (11)$$

We are now ready to prove Lemma 10. Firstly, when  $q = 0$ , by definition  $\tau_0 = 0$ , and so  $\sigma_0 := 0$  trivially satisfies the conditions of Lemma 10. Let us now assume that  $q \geq 1$  and for each of  $0 \leq i \leq q - 1$  we have defined  $\sigma_i$  and functions  $x, r$  and  $c_{j, h}$  on  $[0, \sigma_i]$  for each  $j, h \geq 0$  with  $j + h = i$ , and Lemma 10 holds for all  $0 \leq i \leq q - 1$ . We shall define  $\sigma_q$  which satisfies  $\sigma_q > \sigma_{q-1}$ , extend each  $x, r$  and  $c_{j, h}$  to  $[0, \sigma_q]$ , and define new functions  $c_{k_1, k_2}$  on  $[0, \sigma_q]$  for  $k_1 + k_2 = q$ . We shall then prove that these functions satisfy the assertion of Lemma 10 with respect to  $\tau_q$  and  $\sigma_q$ , which will complete the proof of the lemma.

Fix a sufficiently small constant  $\varepsilon > 0$ , and define the bounded domain

$$\mathcal{D}_\varepsilon := \left\{ (s, x, r, (c_{j, h})_{j+h \in \{q-1, q\}}) : \sigma_{q-1} - 1 < s < 3, |x| < 1 - \varepsilon, |r| < 2, |c_{j, h}| < 2, \varepsilon < \sum_{j, h: j+h=q-1} c_{j, h} < 2 \right\}.$$

It will be convenient to define auxiliary functions to simplify our equations below. Specifically, set  $b_{k_1, k_2} = k_1 \cdot c_{k_1, k_2}$  and  $m_{k_1, k_2} := k_2 \cdot c_{k_1, k_2}$ , as well as  $b = \sum_{j+h \in \{q-1, q\}} c_{j, h}$  and  $m = \sum_{j+h \in \{q-1, q\}} m_{j, h}$ . Finally, set  $d = \sum_{j+h=q-1} c_{j, h}$ . Observe the following system of differential equations:

$$\begin{aligned} x' &= 1 - x + 2 \\ r' &= m - r - \frac{2(b+m)r}{1-x} + \sum_{\substack{j, h: \\ j+h \in \{q-1, q\}}} 2h(m_{j, h} + b_{j, h}) \end{aligned} \quad (12)$$

$$-2r \left( 1 + \frac{r}{1-x} - \frac{m}{1-x} \right) - r \quad (13)$$

If  $k_1 + k_2 = q - 1$ , then:

$$\begin{aligned} c'_{k_1, k_2} &= m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - c_{k_1, k_2} - m_{k_1, k_2} \\ &\quad + 2(b + m) \left( \frac{m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - m_{k_1, k_2}}{1-x} \right) - 2(b_{k_1, k_2} + m_{k_1, k_2}) \\ &\quad + 2r \left( \frac{m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - m_{k_1, k_2} - c_{k_1, k_2}}{1-x} \right) \\ &\quad - (x - 5\ell) \frac{c_{k_1, k_2}}{d} + b_{k_1+1, k_2-1} \cdot \mathbf{1}_{k_2 > 0} - r \frac{c_{k_1, k_2}}{d} - b_{k_1, k_2} \end{aligned} \quad (14)$$

Otherwise, if  $k_1 + k_2 = q$ , then:

$$\begin{aligned} c'_{k_1, k_2} &= m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - c_{k_1, k_2} - m_{k_1, k_2} \\ &\quad + 2(b + m) \left( \frac{m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - m_{k_1, k_2}}{1-x} \right) - 2(b_{k_1, k_2} + m_{k_1, k_2}) \\ &\quad + 2r \left( \frac{m_{k_1-1, k_2+1} \cdot \mathbf{1}_{k_1 > 0} - m_{k_1, k_2} - c_{k_1, k_2}}{1-x} \right) \\ &\quad + (x - 5\ell) \frac{c_{k_1-1, k_2}}{d} + b_{k_1+1, k_2-1} \cdot \mathbf{1}_{k_2 > 0} + r \frac{c_{k_1, k_2-1}}{d} - b_{k_1, k_2} \end{aligned} \quad (15)$$

The right-hand side (r.h.s.) of each of the above equations is Lipchitz on the domain  $\mathcal{D}_\varepsilon$ , as  $d$  is bounded below by  $\varepsilon$ . Define

$$T_{\mathcal{D}_\varepsilon} := \min\{t \geq 0 : (t/n, X(t)/n, R(t)/n, (C_{k_1, k_2}(t)/n)_{k_1+k_2 \in \{q, q-1\}}) \notin \mathcal{D}_\varepsilon\}$$

By the inductive assumption, the “Initial Condition” of Theorem 16 is satisfied for some  $\lambda = o(1)$  and values  $\sigma_{q-1}, x(\sigma_{q-1}), r(\sigma_{q-1})$  and  $c_{j,h}(\sigma_{q-1})$ , where  $c_{j,h}(\sigma_{q-1}) := 0$  for  $j+h = q$ . Moreover, the “Trend Hypothesis” is satisfied with  $\delta = O(1/n)$ , by the expected differences of (8)-(11). Finally, the “Boundedness Hypothesis” is satisfied with  $\beta = O(\log n)$  and  $\gamma = O(n^{-1})$  by Lemma 11. Thus, by Theorem 16, for every  $\delta > 0$ , a.a.s.  $X(t) = nx(t/n) + o(n)$ ,  $R(t) = nr(t/n) + o(n)$  and  $C_{k_1, k_2}(t) = nc_{k_1, k_2}(t/n) + o(n)$  uniformly for all  $\sigma_{q-1}n \leq t \leq (\sigma(\varepsilon) - \delta)n$ , where  $x, \ell_1$  and  $c_{k_1, k_2}$  are the unique solution to (12)-(15) with the above initial conditions, and  $\sigma(\varepsilon)$  is the supremum of  $s$  to which the solution can be extended before reaching the boundary of  $\mathcal{D}_\varepsilon$ . This immediately yields the following lemma.

► **Lemma 12** (Concentration of DegreeGreedy’s Random Variables). *For every  $\delta > 0$ , a.a.s. for all  $\tau_{q-1} \leq t \leq (\sigma(\varepsilon) - \delta)n$  and  $k_1, k_2 \geq 0$  such that  $k_1 + k_2 \in \{q, q-1\}$ ,*

$$\max\{|X(t) - x(t/n)n|, |R(t) - r(t/n)n|, |C_{k_1, k_2}(t) - c_{k_1, k_2}(t/n)n|\} = o(n).$$

As  $\mathcal{D}_\varepsilon \subseteq \mathcal{D}_{\varepsilon'}$  for every  $\varepsilon > \varepsilon'$ ,  $\sigma(\varepsilon)$  is monotonely nondecreasing as  $\varepsilon \rightarrow 0$ , and so  $\sigma_q := \lim_{\varepsilon \rightarrow 0+} \sigma(\varepsilon)$  exists. Moreover, the derivatives of the functions  $x, r$ , and  $c_{k_1, k_2}$  are uniformly bounded on  $(\sigma_{q-1}, \sigma_q)$ , which implies that the functions must be uniformly continuous. The latter condition implies that the functions are (uniquely) continuously extendable to  $[\sigma_{q-1}, \sigma_q]$ , and so the following limits exist:

$$x(\sigma_q) := \lim_{s \rightarrow \sigma_q-} x(s) \quad (16)$$

$$r(\sigma_q) := \lim_{s \rightarrow \sigma_q-} r(s) \quad (17)$$

$$c_{k_1, k_2}(\sigma_q) := \lim_{s \rightarrow \sigma_q-} c_{k_1, k_2}(s). \quad (18)$$

Random variables  $|R(t)/n|$  and  $|C_{k_1, k_2}(t)/n|$  for  $k_1 + k_2 \in \{q, q-1\}$  are both bounded by 1 for all  $t$ . Thus, when  $t/n$  approaches  $\sigma_q$ ,  $X(t)/n$  approaches 1, or  $t/n$  approaches 3, or  $D(t)/n := \sum_{\substack{j, h: \\ j+h=q-1}} C_{j, h}(t)/n$  approaches 0. Formally, we have the following proposition:

► **Proposition 13.** *For every  $\varepsilon > 0$ , there exists  $\delta > 0$  such that a.a.s. one of the following holds.*

- $D(t) < \varepsilon n$  for all  $t \geq (\sigma_q - \delta)n$ ;
- $X(t) > (1 - \varepsilon)n$  for all  $t \geq (\sigma_q - \delta)n$ ;
- $\sigma_q = 3$ .

The ordinary differential equations (12)–(15) again do not have an analytical solution. However, numerical solutions show that  $\sigma_q < 3$ , and  $x(\sigma_q) < 1$ . Thus, after executing **DegreeGreedy** for  $t = \sigma_q n + o(n)$  steps, there are  $D(t) < \varepsilon n$  vertices of type  $q-1$  remaining for some  $\varepsilon = o(1)$ . At this point, by observing the numerical solution (16)–(18) at  $\sigma_q$ , we know that there exists some absolute constant  $0 < p < 1$  such that  $(X(t) - 5L(t))/n \geq p$ , where we recall that  $L(t)$  counts the total number of coloured vertices at time  $t$ . Hence, at each step, some vertex of type  $q-1$  becomes of type  $q$  with probability at least  $p$ . Thus, by applying Chernoff's bound, one can show that a.a.s. after another  $O(\varepsilon n/p) = o(n)$  rounds, all vertices of type  $q-1$  are destroyed. It follows that a.a.s.  $|\tau_q/n - \sigma_q| = o(1)$ , and so Lemma 10 is proven.

## 4 Proof of Theorem 2

Suppose  $G_t$  has a Hamiltonian cycle  $H_t = H$  after  $t \geq 0$  steps. Recall that for the (directed) semi-random edge  $(u_i, v_i)$ , we refer to  $u_i$  as its square and  $v_i$  as its circle. We begin with the following observations:

1.  $H$  uses exactly  $n$  squares;
2.  $H$  uses at most 2 squares on each vertex;
3. Suppose  $(u_i, v_i)$  is an edge of  $G_t$ , and  $v_i$  received at least two squares. Then, either  $H$  uses at most one square on  $v_i$ , or  $H$  does not contain the edge  $(u_i, v_i)$ .

The first two observations above are obvious. For 3, notice that if  $H$  uses exactly 2 squares on  $v_i$ , then these 2 squares correspond to 2 edges in  $H$  that are incident to  $v_i$ . Moreover, neither of these edges can be  $(u_i, v_i)$ , as  $u_i$  is the square of  $(u_i, v_i)$ . Thus, the edge  $(u_i, v_i)$  cannot be used by  $H$  as  $v_i$  has degree 2 in  $H$ .

Define  $Z_x$  as the number of squares on vertex  $x \in [n]$ , the observation 2 above indicates the consideration of the random variable

$$Z = \sum_{x=1}^n (\mathbf{1}_{Z_x=1} + 2 \cdot \mathbf{1}_{Z_x \geq 2}) = 2n - \sum_{x=1}^n (2 \cdot \mathbf{1}_{Z_x=0} + \mathbf{1}_{Z_x=1}),$$

which counts the total number of squares that can possibly contribute to  $H$ , truncated at 2 for each vertex. Observation 3 above indicates the consideration of the following two sets of structures:

Let  $\mathcal{W}_1$  be the set of pairs of vertices  $(x, y)$  at time  $t$  such that

- (a)  $x$  receives its first square in some step  $i < t$ , and  $y$  receives the corresponding circle in the same step;
- (b) no more squares land on  $x$  after step  $i$ ;
- (c) at least two squares land on  $y$  after step  $i$ .

Let  $\mathcal{W}_2$  be the set of pairs of vertices  $(x, y)$  at time  $t$  such that

- (a)  $x$  receives its first square in some step  $i < t$ , and  $y$  receives the corresponding circle in the same step;

- (b) exactly one more square lands on  $x$  either before or after step  $i$ ;
- (c) at least two squares land on  $y$  after step  $s$ .

Note that for every  $(x, y) \in \mathcal{W}_1$ , at most 2 squares on  $x$  and  $y$  together can be used in  $H$ , although  $x$  and  $y$  together contribute 3 to the value of  $Z$ . Similarly, for every  $(x, y) \in \mathcal{W}_2$ , at most 3 squares on  $x$  and  $y$  together can be used in  $H$ , although  $x$  and  $y$  together contribute 4 to the value of  $Z$ .

Therefore, the total number of squares contributing to  $H$  is at most  $Z - |\mathcal{W}_1| - |\mathcal{W}_2| + W$ , where  $W$  accounts for double counting, which sometimes happens when there are  $(x_1, y_1), (x_2, y_2) \in \mathcal{W}_1 \cup \mathcal{W}_2$  where  $\{x_1, y_1\} \cap \{x_2, y_2\} \neq \emptyset$ . More precisely, let

$$\begin{aligned}\mathcal{T}_1 &= \{((x_1, y_1), (x_2, y_2)) \in \mathcal{W}_1 \times \mathcal{W}_2 : y_1 = x_2\} \\ \mathcal{T}_2 &= \{((x_1, y_1), (x_2, y_2)) \in \mathcal{W}_2 \times \mathcal{W}_2 : y_1 = x_2\}.\end{aligned}$$

Then,  $W := |\mathcal{T}_1| + |\mathcal{T}_2|$ .<sup>3</sup>

The random variable  $Z$  is well understood. From the limiting Poisson distribution of the number of squares in a single vertex, we immediately get that a.a.s.  $Z \sim (2 - 2e^{-s} - e^{-s}s)n$  for  $s := t/n$ .

We will estimate the expectation of  $|\mathcal{W}_1|, |\mathcal{W}_2|, |\mathcal{T}_1|, |\mathcal{T}_2|$  as well as the concentration of these random variables. However, concentration may fail if the semi-random process uses a strategy which places many circles on a single vertex. Intuitively, placing many circles on a single vertex is not a good strategy for quickly building a Hamiltonian cycle, as it wastes many edges. To formalise this idea, let  $\mu := \sqrt{n}$  (indeed, choosing any  $\mu$  such that  $\mu \rightarrow \infty$  and  $\mu = o(n)$  will work). We say that a strategy for the semi-random process is  **$\mu$ -well-behaved** up until step  $t$ , if no vertex receives more than  $\mu$  circles in the first  $t$  steps. In [11, Definition 3.2 – Proposition 3.4], it was proven that it is sufficient to consider  $\mu$ -well-behaved strategies in the first  $t = O(n)$  steps for establishing a lower bound on the number of steps needed to build a perfect matching. These definitions and proofs can be easily adapted for building Hamilton cycles in an obvious way. We thus omit the details and only give a high-level explanation below.

The key idea is that within  $t = O(n)$  steps of any semi-random process, the number of vertices of in-degree greater than  $\mu$  is at most  $O(n/\mu) = o(n)$ . Therefore, if a Hamiltonian cycle  $C$  is built in  $t$  steps, then the subgraph  $H$  of  $C$  induced by the set  $S$  of vertices of in-degree at most  $\mu$  in  $G_t$  is a collection of paths spanning all vertices in  $S$  which must also contain  $n - O(n/\mu) = (1 - o(1))n$  edges. We call such a pair  $(S, H)$  an **approximate Hamiltonian cycle**. It follows from the above argument that it takes at least as long time to build a Hamiltonian cycle as to build an approximate Hamiltonian cycle. It is then easy to show by a coupling argument that if a strategy builds an approximate Hamiltonian cycle in  $t = O(n)$  steps, then there exists a well-behaved strategy that builds an approximate Hamiltonian cycle in  $t$  steps as well. Note that observations 2–3 hold for approximate Hamiltonian cycles, and 1 holds for approximate Hamiltonian cycles with  $n$  replaced by  $(1 - o(1))n$ . Thus, no approximate Hamiltonian cycles can be built until step  $Z - |\mathcal{W}_1| - |\mathcal{W}_2| + W \geq (1 - o(1))n$ . We now estimate the sizes of  $\mathcal{W}_1$ ,  $\mathcal{W}_2$ ,  $\mathcal{T}_1$ , and  $\mathcal{T}_2$  in the semi-random process when executing a well-behaved strategy  $\mathcal{S}$ . Crucially, the sizes of these sets do *not* rely on the decisions made by  $\mathcal{S}$ . Recall that  $(G_s^{\mathcal{S}})_{s \geq 0}$  denotes the sequence of graphs produced by  $\mathcal{S}$ .

<sup>3</sup> Note that the cases where  $((x_1, y_1), (x_2, y_2)) \in \mathcal{W}_1 \times \mathcal{W}_1$  such that  $y_1 = y_2$  and  $((x_1, y_1), (x_2, y_2)) \in \mathcal{W}_2 \times \mathcal{W}_2$  such that  $y_1 = x_2$  do not cause double counting.



► **Lemma 14.** Suppose  $\mathcal{S}$  is  $\mu$ -well-behaved. For every  $t = \Theta(n)$ , the following a.a.s. holds in  $G_t^{\mathcal{S}}$ ,

$$Z - |\mathcal{W}_1| - |\mathcal{W}_2| + W \sim f(s)n,$$

where  $s := t/n$  and  $f(s)$  is defined in Theorem 2.

**Proof of Theorem 2.** Recall that  $\beta$  is the positive root of  $f(s) = 1$ . Then, for every  $\varepsilon > 0$ ,  $Z - |\mathcal{W}_1| - |\mathcal{W}_2| + W \leq (1 - O(\varepsilon))n$  a.a.s. in  $G_{(\beta-\varepsilon)n}^{\mathcal{S}}$  for any  $\mu$ -well-behaved  $\mathcal{S}$ . Therefore,  $\tau_{\text{HAM}} \geq \beta$ . ◀

## 5 Conclusion and Open Problems

We have made significant progress on reducing the gap between the previous best upper and lower bounds on  $\tau_{\text{HAM}}$ . That being said, we do not believe that any of our new bounds are tight. For instance, in the case of our lower bound, one could study the appearance of more complicated substructures which prevent any strategy from building a Hamiltonian cycle. One way to likely improve the upper bound would be to analyze an adaptive algorithm whose decisions are all made greedily. Rather, in the terminology of **DegreeGreedy**, when a (second) square lands on a blue vertex, the edge is greedily chosen amongst unsaturated vertices of minimum blue degree (opposed to u.a.r.). Unfortunately, it seems challenging to analyze this algorithm via the differential equation method, but it is likely that this algorithm will lead to an improved upper bound on  $\tau_{\text{HAM}}$  of less than 2.

Another direction is to understand which graph properties exhibit **sharp thresholds**. It is known that for basic properties, such as minimum degree  $k \geq 1$ , sharp thresholds do exist [3]. Moreover, in [2] it was shown that if  $H$  is a spanning graph with max degree  $\Delta = \omega(\log n)$ , then the appearance of  $H$  takes  $(\Delta/2 + o(\Delta))n$  rounds, and  $H$  (deterministically) cannot be constructed in fewer than  $\Delta n/2$  rounds. However, in general it remains open as to whether or not a sharp threshold exists when  $H$  is *sparse* (i.e.,  $\Delta = O(\log n)$ ). Very recently, Surya and the second author [13], developed a general machinery for proving the existence of sharp thresholds in adaptive random graph processes. Applied to the semi-random graph process, they show that sharp thresholds exist for the property of being Hamiltonian as well as to containing a perfect matching. This provides some evidence that sharp thresholds *do* exist when  $\Delta = O(\log n)$ , and we leave this an interesting open problem.

---

## References

- 1 Natalie Behague, Trent Marbach, Paweł Prałat, and Andrzej Ruciński. Subgraph games in the semi-random graph process and its generalization to hypergraphs. *arXiv preprint*, 2022. doi:10.48550/ARXIV.2105.07034.
- 2 Omri Ben-Eliezer, Lior Gishboliner, Dan Hefetz, and Michael Krivelevich. Very fast construction of bounded-degree spanning graphs via the semi-random graph process. *Proceedings of the 31st Symposium on Discrete Algorithms (SODA'20)*, pages 728–737, 2020.
- 3 Omri Ben-Eliezer, Dan Hefetz, Gal Kronenberg, Olaf Parczyk, Clara Shikhelman, and Miloš Stojaković. Semi-random graph process. *Random Structures & Algorithms*, 56(3):648–675, 2020.
- 4 Patrick Bennett and Andrzej Dudek. A gentle introduction to the differential equation method and dynamic concentration. *CoRR*, 2020. doi:10.48550/ARXIV.2007.01994.
- 5 J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

- 6 Tom Bohman and Alan Frieze. Hamilton cycles in 3-out. *Random Structures & Algorithms*, 35(4):393–417, 2009.
- 7 Sofiya Burova and Lyuben Lichev. The semi-random tree process, 2022. doi:10.48550/ARXIV.2204.07376.
- 8 Alan M Frieze. Maximum matchings in a class of random graphs. *Journal of Combinatorial Theory, Series B*, 40(2):196–212, 1986.
- 9 Pu Gao, Bogumił Kamiński, Calum MacRury, and Paweł Prałat. Hamilton cycles in the semi-random graph process. *European Journal of Combinatorics*, 99:103423, 2022. doi:10.1016/j.ejc.2021.103423.
- 10 Pu Gao, Calum MacRury, and Paweł Prałat. A fully adaptive strategy for hamiltonian cycles in the semi-random graph process. *CoRR*, abs/2205.02350, 2022. doi:10.48550/ARXIV.2205.02350.
- 11 Pu Gao, Calum MacRury, and Paweł Prałat. Perfect matchings in the semi-random graph process. *SIAM Journal on Discrete Mathematics*, in press, 2022.
- 12 Michał Karoński, Ed Overman, and Boris Pittel. On a perfect matching in a random digraph with average out-degree below two. *Journal of Combinatorial Theory, Series B*, 143, March 2020. doi:10.1016/j.jctb.2020.03.004.
- 13 Calum MacRury and Erlang Surya. Sharp thresholds in adaptive random graph processes. *CoRR*, 2022.
- 14 Lutz Warnke. On wormald’s differential equation method. *CoRR*, abs/1905.08928, 2019. doi:10.48550/ARXIV.1905.08928.
- 15 Nicholas C Wormald. The differential equation method for random graph processes and greedy algorithms. *Lectures on approximation and randomized algorithms*, 73:155, 1999.

## A Proofs of Lemmas 4 and 7

**Proof of Lemma 4.** As discussed, FullyRandomized ensures that at time  $t$  there are at most 2 red vertices which are not well-spaced. Thus, since our expected differences each allow for a  $O(\log n/n)$  term, without loss of generality, we can assume that all our red vertices are well-spaced. Note that all our explanations below assume that we have conditioned on  $H_t$ . We focus on the second and third expected differences, where we make use of the following crucial observation:

1. Conditional on  $H_t$ , the circles of the red edges of  $\mathcal{L}_t$  are distributed u.a.r. amongst the unsaturated vertices  $U_t$ .

Note that were we to condition on the full history, i.e.,  $G_0, \dots, G_t$ , then these circles would be determined by the history of the process, and so the only randomness in the expectations would be over the draw of  $u_{t+1}$ . By averaging over this additional randomness, we are able to get the claimed expected differences.

Consider now the second expected difference and assume that  $u_{t+1}$  lands on an unsaturated vertex. Firstly, observe that this event occurs with probability  $1 - X(t)/n$ . On the other hand, all the red edges adjacent to  $u_{t+1}$  will be uncoloured after the path augmentation involving  $u_{t+1}$  is made. Now, because of 1., there are  $\frac{L_1(t)}{n-X(t)}$  red edges belonging to  $\mathcal{L}_t^1$  which are adjacent to  $u_{t+1}$  in expectation. After the path augmentation involving  $u_{t+1}$ , these edges are uncoloured and so  $\frac{L_1(t)}{n-X(t)}$  one-red vertices are destroyed in expectation. Now, in expectation there are also  $\frac{2L_2(t)}{n-X(t)} + O(\log n/n)$  red edges adjacent to  $u_{t+1}$  which belong to *distinct* two-red vertices. To see this, fix a two-red vertex  $x \in \mathcal{L}_t^2$  and observe that because of 1., precisely one red edge of  $x$  is adjacent to  $u_{t+1}$  with probability  $\frac{2}{n-X(t)} - \frac{1}{(n-X(t))^2}$ . Since  $n - X(t) \geq n/\log n$  by assumption, this probability is  $\frac{2}{n-X(t)} + O((\log n/n)^2)$ , and so the  $\frac{2L_2(t)}{n-X(t)} + O(\log n/n)$  term follows after summing over all the vertices of  $\mathcal{L}_t^2$ . Now, after the

path augmentation involving  $u_{t+1}$ , these red edges are uncoloured. Since these red edges belonged to distinct two-red vertices, the path augmentation creates  $\frac{2L_2(t)}{n-X(t)} + O(\log n/n)$  new one-red vertices in expectation. These two cases explain the  $\left(1 - \frac{X(t)}{n}\right) \left(\frac{2L_2(t)}{n-X(t)} - \frac{L_1(t)}{n-X(t)}\right)$  term.

Let us now consider when  $u_{t+1}$  lands on a saturated vertex and  $d_{P_t}(u_{t+1}, \mathcal{L}_t) = 1$ , where  $x$  is the unique red vertex adjacent to  $u_{t+1}$ . If  $x$  is a one-red vertex, then let  $r$  be such that  $xr$  is the red edge of  $x$ . Observe that after the augmentation,  $xr$  will be uncoloured, and  $x$  will no longer be a red vertex. Moreover, in expectation there are  $\frac{L_1(t)}{n-X(t)} + O(\log n/n)$  other red edges belonging to  $\mathcal{L}_t^1$  which will be uncoloured. Thus,  $1 + \frac{L_1(t)}{n-X(t)} + O(\log n/n)$  one-red vertices will be destroyed in expectation. On the other hand, there are  $\frac{2L_2(t)}{n-X(t)} + O(\log n/n)$  red edges adjacent to  $r$  which belong to distinct two-red vertices in expectation. Thus,  $\frac{2L_2(t)}{n-X(t)} + O(\log n/n)$  two-red vertices will become one-red vertices in expectation after augmenting via  $xr$  and  $u_{t+1}r$ . Since  $u_{t+1}$  lands next to a one-red vertex with probability,  $\frac{2L_1(t)}{n}$ , this explains the  $\frac{2L_1(t)}{n} \left(\frac{2L_2(t)}{n-X(t)} - \frac{L_1(t)}{n-X(t)} - 1\right)$  term. An analogous argument explains the  $\frac{2L_2(t)}{n} \left(1 + \frac{2L_2(t)}{n-X(t)} - \frac{L_1(t)}{n-X(t)}\right)$  term.

Consider when  $u_{t+1}$  lands in  $\mathcal{Q}_t$ . Observe that this occurs with probability  $\frac{|\mathcal{Q}_t|}{n} = \frac{X(t)-5L(t)}{n}$ . In this case,  $v_{t+1}$  is chosen u.a.r. amongst  $U_t$  and  $u_{t+1}v_{t+1}$  is coloured red. Thus,  $u_{t+1}$  becomes a red vertex, and so  $L_1(t)$  increases by 1 and we get  $\Delta L_1(t) = 1$ . This explains the  $\frac{X(t)-5L(t)}{n}$  term.

The final case to consider is when  $u_{t+1}$  lands on a saturated vertex, and  $u_{t+1} \in \mathcal{L}_t^1$ . Observe that this occurs with probability  $\frac{L_1(t)}{n}$ . Moreover, the algorithm will then choose  $v_{t+1}$  u.a.r. amongst  $U_t$  and colour the edge  $u_{t+1}v_{t+1}$  red. After this move,  $u_{t+1}$  will be converted from a one-red vertex to a two-red vertex, and so  $\Delta L_1(t) = 1$ . This explains the  $\frac{-L_1(t)}{n}$  term.

By combining the contributions from all of the above cases, we get the second expected difference. The third expected difference follows via an analogous argument.  $\blacktriangleleft$

**Proof of Lemma 7.** Let  $j_0 = \varepsilon n$ . For each  $k \geq 1$ , let  $j_k = (1/2)j_{k-1}$  if  $j_{k-1} > n^{1/4}$ , and let  $j_k = j_{k-1} - 1$  otherwise. Clearly,  $j_k$  is a decreasing function of  $k$ . Let  $\tau_1$  be the smallest natural number  $k$  such that  $j_k \leq n^{1/4}$ . Let  $\tau$  be the natural number  $k$  such that  $j_k = 0$ . Obviously,  $\tau_1 = O(\log n)$  and  $\tau = O(n^{1/4})$ .

We use a cleaning-up algorithm, which runs in iterations. The  $k$ -th iteration repeatedly absorbs  $j_{k-1} - j_k$  vertices into  $P$ , leaving  $j_k$  unsaturated vertices in the end. The  $k$ -th iteration of the cleaning-up algorithm works as follows.

- (i) (*Initialising*): Uncolour all vertices in the graph;
- (ii) (*Building reservoir*): Let  $m_k := \sqrt{\varepsilon}(1/2)^{k/2}n$  for  $k \leq \tau_1$  and  $m_k := n^{1/2}$  if  $\tau_1 < k \leq \tau$ . Add  $m_k$  semi-random edges as follows. If  $u_t$  lands on an unsaturated vertex, a red vertex or a neighbour of a red vertex in  $P$ , then let  $v_t$  be chosen arbitrarily. This edge  $u_t v_t$  will not be used in our construction. Otherwise, colour  $u_t$  red and choose an arbitrary  $v_t$  among those unsaturated vertices with the minimum number of red neighbours. Colour  $u_t v_t$  red. Note that each red vertex is adjacent to exactly one red edge;
- (iii) (*Absorbing via path augmentations*): Add semi-random edges as follows. Suppose that  $u_t$  lands on  $P$  and at least one neighbour of  $u_t$  on  $P$  is red. (Otherwise,  $v_t$  is chosen arbitrarily, and this edge will not be used in our construction.) Let  $x$  be such red vertex (if  $u_t$  has two neighbours on  $P$  that are red, then select one of them arbitrarily). Let  $y$  be the neighbour of  $x$  where  $xy$  is red, and let  $v_t = y$ . Extend  $P$  by deleting the edge

$xu_t$  and adding the edges  $xy$  and  $yu_t$ . Uncolour all red edges incident to  $y$  and all red neighbours of  $y$  (which, of course, includes vertex  $x$ ).

Notice that, in each iteration,  $m_k \geq n^{1/2}$ . Indeed, this is obviously true for  $\tau_1 < k \leq \tau$ . On the other hand, if  $k \leq \tau_1$ , then  $j_k = \varepsilon n(1/2)^k$  and so  $m_k = \sqrt{n j_k} \geq \sqrt{n}$  (in fact,  $m_k = \Omega(n^{5/8})$ ).

Let  $T_k$  denote the length of the  $k$ -th iteration of the cleaning-up algorithm. It remains to prove that a.a.s.  $\sum_{k \leq \tau} T_k = O(\sqrt{\varepsilon}n)$ . Let  $R_k$  be the number of red vertices obtained after step (ii) of iteration  $k$ . Obviously,  $R_k \leq m_k$ . On the other hand, each  $u_t$  is coloured red with probability at least  $1 - j_{k-1}/n - 3m_k/n \geq 1 - \varepsilon - 3\sqrt{\varepsilon} \geq 0.95$ . Hence,  $R_k$  can be stochastically lower bounded by the binomial random variable  $\text{Bin}(m_k, 0.95)$ . By the Chernoff bound, with probability at least  $1 - n^{-1}$ ,  $R_k \geq 0.9m_k$ , as  $m_k \geq n^{1/2}$ .

First, we consider iterations  $k \leq \tau_1$ . Let  $\tilde{R}_k$  be the number of red vertices at the end of step (iii). Note that the minimum degree property of step (ii) ensures each unsaturated vertex is adjacent to at most  $R_k/j_{k-1} + 1 \leq m_k/j_{k-1} + 1$  red vertices. Moreover, exactly  $j_{k-1} - j_k = (1/2)j_{k-1}$  vertices are absorbed in step (iii). As a result,

$$\tilde{R}_k \geq R_k - \left( \frac{m_k}{j_{k-1}} + 1 \right) \cdot \frac{j_{k-1}}{2} \geq 0.9m_k - \frac{m_k}{2} - \frac{j_{k-1}}{2} \geq 0.3m_k,$$

as  $j_{k-1} = 2j_k \leq 2\sqrt{\varepsilon}m_k \leq 0.1m_k$ . It follows that throughout step (iii), there are at least  $0.3m_k$  red vertices. Thus, for each semi-random edge added to the graph, the probability that a path extension can be performed is at least  $0.3m_k/n = 0.3\sqrt{\varepsilon}(1/2)^{k/2}$ . Again, by the Chernoff bound, with probability at least  $1 - n^{-1}$ , the number of semi-random edges added in step (iii) is at most

$$2(j_{k-1} - j_k) \cdot \frac{2^{k/2}}{0.3\sqrt{\varepsilon}} \leq 7\sqrt{\varepsilon}(1/2)^{k/2}n.$$

Combining the number of semi-random edges added in step (ii), it follows that with probability at least  $1 - n^{-1}$ ,  $T_k \leq m_k + 7\sqrt{\varepsilon}(1/2)^{k/2}n = 8\sqrt{\varepsilon}(1/2)^{k/2}n$ .

Next, consider iterations  $\tau_1 < k \leq \tau$ . In each iteration, exactly one unsaturated vertex gets absorbed. The number of semi-random edges added in step (ii) is  $m_k = n^{1/2}$ . We have argued that with probability at least  $1 - n^{-1}$ ,  $R_k \geq 0.9m_k$ . Thus, for each semi-random edge added to the graph, the probability that a path extension can be performed is at least  $0.9m_k/n = 0.9n^{-1/2}$ . By the Chernoff bound, with probability at least  $1 - n^{-1}$ , the number of semi-random edges added in step (iii) is at most  $n^{1/2} \log^2 n$ . Thus, with probability at least  $1 - n^{-1}$ ,  $T_k \leq n^{1/2} + n^{1/2} \log^2 n \leq 2n^{1/2} \log^2 n$ .

Taking the union bound over all  $k \leq \tau$ , since  $\tau = O(n^{1/4})$ , it follows that a.a.s.

$$\sum_{k \leq \tau} T_k \leq \sum_{k \leq \tau_1} 8\sqrt{\varepsilon}(1/2)^{k/2}n + \sum_{\tau_1 < k \leq \tau} 2n^{1/2} \log^2 n = O(\sqrt{\varepsilon}n + n^{3/4} \log^2 n)$$

We have shown that a.a.s. by adding  $O(\sqrt{\varepsilon}n + n^{3/4} \log^2 n)$  additional semi-random edges we can construct a Hamiltonian path  $P$ . To complete the job and turn it into a Hamiltonian cycle, let  $u$  and  $v$  denote the left and, respectively, the right endpoint of  $P$ . First, add  $n^{1/2}$  semi-random edges  $u_t v_t$  where  $v_t$  is always  $u$ , discarding any multiple edges that could possibly be created. For each such semi-random edge  $u_t u$ , colour the left neighbour of  $u_t$  on  $P$  blue. Next, add  $n^{1/2} \log^2 n$  semi-random edges  $u_t v_t$  where  $v_t$  is always  $v$ . Suppose that some  $u_t = x$  is blue. Then, a Hamiltonian cycle is obtained by deleting  $xy$  from  $P$  and adding the edges  $xv$  and  $uy$ , where  $y$  is the right neighbour of  $x$  on  $P$ . By Chernoff bound, a.a.s. a semi-random edge added during the second round hits a blue vertex, completing the proof.  $\blacktriangleleft$

## B Proof of Lemma 14

► **Lemma 15.** *Suppose  $\mathcal{S}$  is  $\mu$ -well-behaved up until time  $t = \Theta(n)$ . A.a.s. the following holds for  $G_t^{\mathcal{S}}$ :*

$$|\mathcal{W}_1| \sim n \sum_{i \leq t} \frac{1}{n} \left(1 - \frac{1}{n}\right)^t \sum_{i \leq j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{j_2} \quad (19)$$

$$|\mathcal{W}_2| \sim n \sum_{i_1 \leq i_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \left( \sum_{i_1 \leq j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{j_2} + \sum_{i_2 < j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{j_2} \right) \quad (20)$$

$$\begin{aligned} |\mathcal{T}_1| &\sim n \sum_{i \leq t} \frac{1}{n} \left(1 - \frac{1}{n}\right)^t \sum_{i \leq j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \\ &\quad \times \left( \sum_{j_1 \leq h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} + \sum_{j_2 < h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} \right) \end{aligned} \quad (21)$$

$$\begin{aligned} |\mathcal{T}_2| &\sim n \sum_{i_1 < i_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \sum_{i_1 \leq j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \\ &\quad \times \left( \sum_{j_1 \leq h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} + \sum_{j_2 < h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} \right) \\ &\quad + \sum_{i_1 < i_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \sum_{i_2 \leq j_1 < j_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^t \\ &\quad \times \left( \sum_{j_1 \leq h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} + \sum_{j_2 < h_1 < h_2 \leq t} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{h_2} \right). \end{aligned} \quad (22)$$

**Proof.** We prove (19) and briefly explain the expressions in (20)–(22) whose proofs are similar to that of (19). Fix a vertex  $x \in [n]$  and a square  $u_i$  for  $i \leq t$ . The probability that  $u_i$  lands on  $x$  in step  $i$  is  $1/n$ . Condition on this event. The probability that  $x$  receives no squares in any steps other than  $i$  is  $(1 - 1/n)^{t-1} \sim (1 - 1/n)^t$ . Let  $y$  be the vertex which the strategy chooses to pair with  $u_i$  with. Fix any two integers  $i < j_1 < j_2 \leq t$ , the probability that  $y$  receives its first two squares at times  $j_1$  and  $j_2$  is  $n^{-2}(1 - 1/n)^{j_2-2} \sim n^{-2}(1 - 1/n)^{j_2}$ . Summing over all possible values of  $i, j_1, j_2$  and multiplying by  $n$ , the number of choices for  $x$ , gives  $\mathbb{E}|\mathcal{W}_1|$ .

For concentration of  $|\mathcal{W}_1|$  we prove that  $\mathbb{E}|\mathcal{W}_1|^2 \sim (\mathbb{E}|\mathcal{W}_1|)^2$ . For any pair of  $((x_1, y_1), (x_2, y_2))$  in  $\mathcal{W}_1 \times \mathcal{W}_1$ , either  $x_1, y_1, x_2, y_2$  are pairwise distinct, or  $y_1 = y_2$ . It is easy to see that the expected number of pairs where  $x_1, y_1, x_2, y_2$  are pairwise distinct is

$$n^2 \sum_{\substack{i_1 \leq t \\ i_2 \leq t}} \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{2(t-1)} \sum_{\substack{i_1 \leq j_1 < j_2 \leq t \\ i_2 \leq h_1 < h_2 \leq t}} \frac{1}{n^4} \left(1 - \frac{1}{n}\right)^{j_2-2+h_2-2} \sim (\mathbb{E}|\mathcal{W}_1|)^2.$$

The expected number of pairs where  $y_1 = y_2$  is at most  $\mu n$  as there are most  $n$  choices for  $x_1$  and given  $(x_1, y_1)$ , there can be at most  $\mu$  choices for  $(x_2, y_2)$  since  $\mathcal{S}$  is  $\mu$ -well-behaved. Since  $\mu = o(n)$ ,  $\mu n = o(n^2)$  which is  $o((\mathbb{E}|\mathcal{W}_1|)^2)$ . Thus we have verified that  $\mathbb{E}|\mathcal{W}_1|^2 \sim (\mathbb{E}|\mathcal{W}_1|)^2$  and thus by the second moment method, a.a.s.  $|\mathcal{W}_1| \sim \mathbb{E}|\mathcal{W}_1|$ .

The proofs for the expectation and concentration of  $|\mathcal{W}_2|$ ,  $|\mathcal{T}_1|$  and  $|\mathcal{T}_2|$  are similar. We briefly explain the expressions in (20)–(22):

In (20),  $i_1$  and  $i_2$  denote the two steps at which  $x$  receives a square. Since there are two squares on  $x$ , there are two choices of circles, namely  $v_{i_1}$  and  $v_{i_2}$ . The two summations over  $(j_1, j_2)$  accounts for the two choices of  $v_{i_1}$  and  $v_{i_2}$ , depending on which is to be covered by two squares. Thus,  $j_1$  and  $j_2$  denote the steps where the first two squares on  $v_{i_1}$  or  $v_{i_2}$  arrive.

In (21),  $i$  denotes the step where  $x_1$  receives its only square;  $j_1$  and  $j_2$  denote the two steps where  $y_1 = x_2$  receives its two squares. Hence, there are two choices for  $y_2$ , and  $h_1$  and  $h_2$  denote the two steps of the first two squares  $y_2$  receives.

In (22),  $i_1$  and  $i_2$  denote the two steps where  $x_1$  receives its two squares – hence there are two choices for  $y_1$ . Integers  $j_1$  and  $j_2$  denote the two steps where  $y_1 = x_1$  receives its two squares – hence there are two choices for  $y_2$ . Finally,  $h_1$  and  $h_2$  denote the steps where  $y_2$  receives its first two squares.  $\blacktriangleleft$

From Lemma 15, we deduce that for  $t = sn$ ,

$$\begin{aligned}
|\mathcal{W}_1| &\sim ne^{-s} \int_0^s dx \int_x^s dy_1 \int_{y_1}^s e^{-y_2} dy_2 = ne^{-s} \left( 1 - \frac{e^{-s}s^2}{2} - e^{-s}s - e^{-s} \right) \\
|\mathcal{W}_2| &\sim ne^{-s} \int_0^s dx_1 \int_{x_1}^s dx_2 \left( \int_{x_1}^s dy_1 \int_{y_1}^s e^{-y_2} dy_2 + \int_{x_2}^s dy_1 \int_{y_1}^s e^{-y_2} dy_2 \right) \\
&= ne^{-s} \left( s - e^{-s}s^2 - \frac{e^{-s}s^3}{2} - e^{-s}s \right) \\
|\mathcal{T}_1| &\sim ne^{-2s} \int_0^s dx \int_x^s dy_1 \int_{y_1}^s dy_2 \left( \int_{y_1}^s dz_1 \int_{z_1}^s e^{-z_2} dz_2 + \int_{y_2}^s dz_1 \int_{z_1}^s e^{-z_2} dz_2 \right) \\
&= ne^{-2s} \left( -1 + s - \frac{e^{-s}s^3}{3} - \frac{e^{-s}s^2}{2} - \frac{e^{-s}s^4}{8} + e^{-s} \right) \\
|\mathcal{T}_2| &\sim ne^{-2s} \int_0^s dx_1 \int_{x_1}^s dx_2 \int_{x_1}^s dy_1 \int_{y_1}^s dy_2 \left( \int_{y_1}^s dz_1 \int_{z_1}^s e^{-z_2} dz_2 + \int_{y_2}^s dz_1 \int_{z_1}^t e^{-z_2} dz_2 \right) \\
&\quad + ne^{-2s} \int_0^s dx_1 \int_{x_1}^s dx_2 \int_{x_2}^s dy_1 \int_{y_1}^s dy_2 \left( \int_{y_1}^s dz_1 \int_{z_1}^s e^{-z_2} dz_2 + \int_{y_2}^s dz_1 \int_{z_1}^s e^{-z_2} dz_2 \right) \\
&= ne^{-2s} \left( -s + s^2 - e^{-s}s \left( \frac{s^4}{8} + \frac{s^3}{3} + \frac{s^2}{2} - 1 \right) \right)
\end{aligned}$$

It follows now that  $Z - |\mathcal{W}_1| - |\mathcal{W}_2| + W \sim f(s)n$  where recall that

$$f(s) = 2 + e^{-3s}(s+1) \left( 1 - \frac{s^2}{2} - \frac{s^3}{3} - \frac{s^4}{8} \right) + e^{-2s} \left( 2s + \frac{5s^2}{2} + \frac{s^3}{2} \right) - e^{-s}(3+2s).$$

## C The Differential Equation Method

In this section, we provide a self-contained *non-asymptotic* statement of the differential equation method. The statement combines [14, Theorem 2], and its extension [14, Lemma 9], in a form convenient for our purposes, where we modify the notation of [14] slightly. In particular, we rewrite [14, Lemma 9] in a less general form in terms of a stopping time  $T$ . We need only check the “Boundedness Hypothesis” (see below) for  $0 \leq t \leq T$ , which is exactly the setting of Lemmas 3 and 11.

Suppose we are given integers  $a, n \geq 1$ , a bounded domain  $\mathcal{D} \subseteq \mathbb{R}^{a+1}$ , and functions  $(F_k)_{1 \leq k \leq a}$  where each  $F_k : \mathcal{D} \rightarrow \mathbb{R}$  is  $L$ -Lipschitz-continuous on  $\mathcal{D}$  for  $L \geq 0$ . Moreover, suppose that  $R \in [1, \infty)$  and  $S \in (0, \infty)$  are *any* constants which satisfy  $\max_{1 \leq k \leq a} |F_k(x)| \leq R$  for all  $x = (s, y_1, \dots, y_a) \in \mathcal{D}$  and  $0 \leq s \leq S$ .

► **Theorem 16** (Differential Equation Method, [14]). *Suppose we are given  $\sigma$ -fields  $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \dots$ , and for each  $t \geq 0$ , random variables  $((Y_k(t))_{1 \leq k \leq a})$  which are  $\mathcal{F}_t$ -measurable. Define  $T_{\mathcal{D}}$  to be the minimum  $t \geq 0$  such that*

$$(t/n, Y_1(t)/n, \dots, Y_a(t)/n) \notin \mathcal{D}.$$

*Let  $T \geq 0$  be an (arbitrary) stopping time<sup>4</sup> adapted to  $(\mathcal{F}_t)_{t \geq 0}$ , and assume that the following conditions hold for  $\delta, \beta, \gamma \geq 0$  and  $\lambda \geq \delta \min\{S, L^{-1}\} + R/n$ :*

(i) *The “Initial Condition”: For some  $(0, \hat{y}_1, \dots, \hat{y}_a) \in \mathcal{D}$ ,*

$$\max_{1 \leq k \leq a} |Y_k(0) - \hat{y}_k n| \leq \lambda n.$$

(ii) *The “Trend Hypothesis”: For each  $t \leq \min\{T, T_{\mathcal{D}} - 1\}$ ,*

$$|\mathbb{E}[Y_k(t+1) - Y_k(t) \mid \mathcal{F}_t] - F_k(t/n, Y_1(t)/n, \dots, Y_a(t)/n)| \leq \delta.$$

(iii) *The “Boundedness Hypothesis”: With probability  $1 - \gamma$ ,*

$$|Y_k(t+1) - Y_k(t)| \leq \beta,$$

*for each  $t \leq \min\{T, T_{\mathcal{D}} - 1\}$ :*

*Then, with probability at least  $1 - 2a \exp\left(\frac{-n\lambda^2}{8S\beta^2}\right) - \gamma$ , we have that*

$$\max_{0 \leq t \leq \min\{T, \sigma n\}} \max_{1 \leq k \leq a} |Y_k(t) - y_k(t/n)n| < 3\lambda \exp(LS)n, \quad (23)$$

*where  $(y_k(s))_{1 \leq k \leq a}$  is the unique solution to the system of differential equations*

$$y'_k(s) = F_k(s, y_1(s), \dots, y_a(s)) \quad \text{with } y_k(0) = \hat{y}_k \text{ for } 1 \leq k \leq a, \quad (24)$$

*and  $\sigma = \sigma(\hat{y}_1, \dots, \hat{y}_a) \in [0, S]$  is any choice of  $\sigma \geq 0$  with the property that  $(s, y_1(s), \dots, y_a(s))$  has  $\ell^\infty$ -distance at least  $3\lambda \exp(LS)$  from the boundary of  $\mathcal{D}$  for all  $s \in [0, \sigma]$ .*

► **Remark 17.** Standard results for differential equations guarantee that (24) has a unique solution  $(y_k(s))_{1 \leq k \leq a}$  which extends arbitrarily close to the boundary of  $\mathcal{D}$ .

► **Remark 18.** The proof of Theorem 16 works for any choice of  $R \in [1, \infty)$  and  $T \in (0, \infty)$  which satisfy  $\max_{1 \leq k \leq a} |F_k(x)| \leq R$  for all  $x = (s, y_1, \dots, y_a) \in \mathcal{D}$  and  $0 \leq s \leq T$ .

<sup>4</sup> The stopping time  $T \geq 0$  is **adapted** to  $(\mathcal{F}_t)_{t \geq 0}$ , provided the event  $\{\tau = t\}$  is  $\mathcal{F}_t$ -measurable for each  $t \geq 0$ .



# Cover and Hitting Times of Hyperbolic Random Graphs

Marcos Kiwi   

Department of Industrial Engineering and Center for Mathematical Modeling,  
Universidad de Chile, Santiago, Chile

Markus Schepers   

Institut für Medizinische Biometrie, Epidemiologie und Informatik,  
Johannes-Gutenberg-University Mainz, Germany

John Sylvester   

School of Computing Science, University of Glasgow, UK

---

## Abstract

We study random walks on the giant component of Hyperbolic Random Graphs (HRGs), in the regime when the degree distribution obeys a power law with exponent in the range  $(2, 3)$ . In particular, we focus on the expected times for a random walk to hit a given vertex or visit, i.e. cover, all vertices. We show that up to multiplicative constants: the cover time is  $n(\log n)^2$ , the maximum hitting time is  $n \log n$ , and the average hitting time is  $n$ . The first two results hold in expectation and a.a.s. and the last in expectation (with respect to the HRG).

We prove these results by determining the effective resistance either between an average vertex and the well-connected “center” of HRGs or between an appropriately chosen collection of extremal vertices. We bound the effective resistance by the energy dissipated by carefully designed network flows associated to a tiling of the hyperbolic plane on which we overlay a forest-like structure.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains; Theory of computation  $\rightarrow$  Random network models; Mathematics of computing  $\rightarrow$  Stochastic processes

**Keywords and phrases** Random walk, hyperbolic random graph, cover time, hitting time, average hitting time, target time, effective resistance, Kirchhoff index

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.30

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2207.06956> [49]

**Funding** *Marcos Kiwi:* gratefully acknowledges the support of grant GrHyDy ANR-20-CE40-0002 and BASAL funds for centers of excellence from ANID-Chile grants ACE210010 and FB210005.

*Markus Schepers:* gratefully acknowledges the support of the DIAMANT PhD travel grant for a 2-week research stay in Santiago de Chile.

*John Sylvester:* was supported by EPSRC project EP/T004878/1: Multilayer Algorithmics to Leverage Graph Structure and the ERC Starting Grant 679660 (DYNAMIC MARCH).

## 1 Introduction

In 2010, Krioukov et al. [52] proposed the Hyperbolic Random Graph (HRG) as a model of “real-world” networks such as the Internet (also referred to as complex networks). Early results via non-rigorous methods indicated that HRGs exhibited several key properties empirically observed in frequently studied networks (such as networks of acquaintances, citation networks, networks of autonomous systems [13], etc.). Many of these properties were later established formally, among these are power-law degree distribution [37], short graph distances [1, 46] (a.k.a. small world phenomena), and strong clustering [17, 31, 37]. Many



© Marcos Kiwi, Markus Schepers, and John Sylvester;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 30; pp. 30:1–30:19



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

other fundamental parameters of the HRG model have been studied since its introduction (see the related work section), however notable exceptions are key quantities concerning the behaviour of random walks. This paper is a first step in redressing this situation. The random walk is the quintessential random process, and studies of random walks have proven relevant for algorithm design and analysis; this coupled with the aforementioned appealing aspects of the HRG model motivates this research.

The (simple) random walk is a stochastic process on the vertices of a graph, which at each time step uniformly samples a neighbour of the current vertex as its next state [3, 56]. A key property of the random walk is that, for any connected graph, the expected time it takes for the walk to visit a given vertex (or to visit all vertices) is polynomial in the number of vertices in the graph. These times are known as the hitting and cover times, respectively. This ability of a random walk to explore an unknown connected graph efficiently using a small amount of memory was, for example, used to solve the undirected  $s - t$  connectivity problem in logarithmic space [4]. Other properties such as the ability to sample a vertex independently of the start vertex after a polynomial (often logarithmic) number of steps (mixing time) helped random walks become a fundamental primitive in the design of randomized and approximation algorithms [59]. In particular, random walks have been applied in tasks such as load balancing [68], searching [35], resource location [44], property testing [26, 53, 54], graph parameter estimation [8] and biological applications [38].

One issue to keep in mind when working with HRGs is that for the most relevant range of parameters of the model (that one for which it exhibits the properties observed in “real-world” networks) the graphs obtained are disconnected with probability that tends to 1 as the order of the graph goes to infinity. Quantities such as average hitting time and commute time are not meaningful for disconnected graphs (i.e., they are trivially equal to infinity). However, again for the range of parameters we are interested in, Bode, Fountoulakis and Müller [11] showed that it is very likely the graph has a component of linear size. This result was then complemented by the first author and Mitsche [46] who showed that all other connected components were of size at most polylogarithmic in the order of the graph. This justifies referring to the linear size component as the *giant component*. With this work being among the first study of characteristics of simple random walks in HRGs, it is thus natural and relevant to understand their behavior in the giant component of such graphs. This is the main challenge we undertake in this paper.

Among our main contributions are the determination of the order of the hitting and cover times of random walks on the giant component of HRGs. To achieve this, we appeal to a connection of the former to effective resistances in the graph [56, Section 9]. The effective resistance is a metric, and the resistances between all pairs of vertices uniquely determines the graph [40]. The effective resistance has also found applications to graph clustering [5], spectral sparsification [69], graph convolutional networks [2], and flow-based problems in combinatorial optimization [6, 18, 61].

## 1.1 Main Results

Our main contributions are to determine several quantities related to random walks on the largest connected component  $\mathcal{C}_{\alpha,\nu}(n)$  of the (Poissonized) hyperbolic random graph  $\mathcal{G}_{\alpha,\nu}(n)$ . We refer to this component as the *giant* and note that it is known to have  $\Theta(n)$  vertices a.a.s. [11]. The primary probability space we will be working in is the one induced by the HRG and we use  $\mathbb{P}$  for the associated measure. We also deal with the expected stopping times of random walks, and we use bold type (e.g.  $\mathbf{E}$ ) for the expectation with respect to the

random walk on a fixed graph. We say that a sequence of events (w.r.t. the HRG) holds *asymptotically almost surely (a.a.s.)* if it occurs with probability going to 1 as  $n \rightarrow \infty$ . We give brief descriptions of the objects we study here, for full definitions see Section 2.

The *effective resistance*  $\mathcal{R}(x \leftrightarrow y)$  between two vertices  $x, y$  of a graph  $G$  is the energy dissipated by a unit current flow from  $x$  to  $y$ . Due to a connection with simple random walks, we consider effective resistance in the case when all edges have unit resistances, see Section 2.5 for a formal definition. The sum of all resistances in  $G$  is the *Kirchhoff index*  $\mathcal{K}(G)$ , this has found uses in centrality [57], noisy consensus problems [67], and social recommender systems [73]. Our first result shows the expected effective resistance between two vertices of the giant chosen uniformly at random is bounded, and gives the expected order of the Kirchhoff index.

► **Theorem 1.** *For any  $\frac{1}{2} < \alpha < 1$  and  $\nu > 0$ , if  $\mathcal{C} := \mathcal{C}_{\alpha, \nu}(n)$ , then*

$$\mathbb{E}(\mathcal{K}(\mathcal{C})) = \Theta(n^2), \quad \text{and} \quad \mathbb{E}\left(\frac{1}{|V(\mathcal{C})|^2} \sum_{u, v \in V(\mathcal{C})} \mathcal{R}(u \leftrightarrow v)\right) = \Theta(1).$$

The upper bounds in Theorem 1 are established by exploiting the well known relation between effective resistance and energy dissipated by network flows. The two results in this theorem are very closely related but do not directly imply each other as  $|V(\mathcal{C})|$ , the size of the center component, is a random variable that is not independent from the resistances.

Our construction of low energy flows relies on a tiling of the hyperbolic plane. In this regard, it bears some similarity to how various authors have obtained estimates of the size of the giant and upper bounds on the diameter of the HRG [64]. However, when constructing a desirable flow one often needs multiple paths (as opposed to just one when bounding the diameter) or else the energy dissipated by the flow could be too large to get a tight bound on the effective resistance. Abdullah et al. [1] showed that hyperbolic random graphs of expected size  $\Theta(n)$  have typical distances of length  $\Theta(\log \log n)$  (within the same component), in contrast we show that typical resistances are  $\Theta(1)$ . The diameter of the HRG when  $\frac{1}{2} < \alpha < 1$  was only recently determined precisely [64], though the lower bound, non-tight upper bounds, and the diameter for other values of  $\alpha$ , were established earlier [46, 33]. The tight  $\mathcal{O}(\log n)$  upper bound for the diameter of the giant of the HRG when  $\frac{1}{2} < \alpha < 1$  [64] was proved using a coupling with the Fountoulakis-Müller upper half-plane HRG model [30] and is also based in a tiling-construction. The tiling on which we rely to construct flows is closely related to the Fountoulakis-Müller tiling of the half-plane model. In fact, our tiling is approximately equal to the latter (see the discussion in the last paragraph of the detailed description of our tiling in Section 3.1).

The *target time*  $t_{\odot}(G)$  of a graph  $G$  (also known as *Kemeny's constant*) is the expected time for a random walk to travel between two vertices chosen independently from the stationary distribution  $\pi$ , see Section 2.4. When considering a random walk on a graph, the stationary distribution is arguably the most natural measure on the vertices. Thus the target time should be considered as the “average” hitting time. We show that on the giant of the HRG this notion of average hitting time is of order  $n$  in expectation.

► **Theorem 2.** *For any  $\frac{1}{2} < \alpha < 1$  and  $\nu > 0$ , if  $\mathcal{C} := \mathcal{C}_{\alpha, \nu}(n)$ , then  $\mathbb{E}(t_{\odot}(\mathcal{C})) = \Theta(n)$ .*

The *hitting time* of a vertex  $v$  from a vertex  $u$  in a graph  $G$  is the expected time it take a random walk started from  $u$  to first visit  $v$ . Let  $t_{\text{hit}}(G)$  denote the *maximum hitting time*, this is the maximum over all pairs of vertices  $u, v$  in  $V(G)$  of the hitting time of  $u$  from  $v$ . We let the *cover time*  $t_{\text{cov}}(G)$  be the expected time for the walk to visit all vertices of  $G$  (taken from a worst case start vertex), see Section 2.4. We show that both of these quantities concentrate on the giant of the HRG.

► **Theorem 3.** *For any  $\frac{1}{2} < \alpha < 1$  and  $\nu > 0$ , if  $\mathcal{C} := \mathcal{C}_{\alpha,\nu}(n)$ , then a.a.s. and in expectation*

$$t_{\text{hit}}(\mathcal{C}) = \Theta(n \log n) \quad \text{and} \quad t_{\text{cov}}(\mathcal{C}) = \Theta(n \log^2 n).$$

The result above also establishes that the maximum resistance between two vertices of the giant is  $\Theta(\log n)$  a.a.s., compared to  $\Theta(1)$  for a typical pair by Theorem 1. This discrepancy between the maximum and the average resistances is also seen in graph distances in the giant, as the maximum and average distances are  $\Theta(\log n)$  [64] and  $\Theta(\log \log n)$  [1] a.a.s., respectively. Interestingly, there are enough (polynomially many) pairs of vertices with resistance matching the maximum to ensure that the cover time is a factor  $\Theta(\log^2 n)$  larger than the average hitting time, many random graphs (e.g., connected Erdős-Rényi, preferential attachment) are expanders and do not have this feature.

Stating additional contributions of this paper, as well as providing more detail about those already stated, requires a bit more terminology and notation, which we introduce below after discussing the related literature.

## 1.2 Further Related Work and Our Techniques

Over the last two decades, the cover time of many random graph models has been determined. These networks include the binomial random graph [20, 22], random geometric graph [23], preferential attachment model [21], configuration model [25], random digraphs [24] and the binomial random intersection graph [10]. These results were all proven using Cooper and Frieze’s *first visit lemma*, see the aforementioned papers or [62]. This result is based on expressing the probability that a vertex has been visited up-to a given time by a function of the return probabilities. One (simplified) condition required to easily apply the first visit lemma is that  $t_{\text{rel}} \cdot \max_{v \in V} \pi(v) = o(1)$ , where  $\pi$  is the stationary distribution and  $t_{\text{rel}}(G) := \frac{1}{1-\lambda_2}$  is the *relaxation time* of  $G$ , and  $\lambda_2$  is the second-largest eigenvalue of the transition matrix of the (lazy) random walk on  $G$ . However, inserting the best known bounds on  $t_{\text{rel}}$  and  $\max_{v \in V} \pi(v)$  for the HRG, by [47] and [37] respectively, gives  $t_{\text{rel}} \cdot \max_{v \in V} \pi(v) \leq (n^{2\alpha-1} \log n) \cdot n^{\frac{1}{2\alpha}-1+o(1)}$  which is not  $o(1)$  for any  $\frac{1}{2} \leq \alpha \leq 1$ .

Another key ingredient of the first visit lemma is good bounds on the expected number of returns to a vertex before the walk mixes, i.e.  $\sum_{t=0}^{t_{\text{mix}}} P_{v,v}^t$  where  $P_{x,y}^t$  is the probability a (lazy) random walk from  $x$  is at vertex  $y$  after exactly  $t$  steps. Obtaining such bounds in the HRG appears challenging due to the large mixing time and irregular local structure of the HRG. This also effects arguably the most natural approach to obtaining bounds on the average hitting time, that is applying the formula  $\pi(v)E_\pi[\tau_v] = \sum_{t=0}^{\infty} [P_{v,v}^t - \pi(v)]$ , see [3, Lemma 2.11], as this involves the same sum (which only needs to be considered up to relaxation/mixing time).

Given the perceived difficulty in determining the cover time using the return probabilities as described above, the approach taken in this paper is to determine the hitting and cover times via the effective resistances  $\{\mathcal{R}(u \leftrightarrow v)\}_{u,v \in V}$ . There is an intimate connection between reversible Markov chains and electrical networks as certain quantities in each setting are determined by the same harmonic equations. Classically this connection has been exploited to determine whether random walks on infinite graphs are transient or recurrent [60, Chapter 2], and more recently the effective resistance metric has been understood to relate the blanket times of random walks on finite graphs to the Gaussian free field [28]. The main connection we shall use is that the commute time (sum of hitting times in either direction) between two vertices is equal to the number of edges times the effective resistance between the two points [16, 71]. This result has been used to bound hitting and cover times in several random graph models, notably in the binomial random graph [41, 70] and the geometric random graph [7]. Luxburg et al. [72] recently refined a previous bound of Lovász [59] to give

$$\left| \mathcal{R}(u \leftrightarrow v) - \frac{1}{d(u)} - \frac{1}{d(v)} \right| \leq \frac{t_{\text{rel}} + 2}{d_{\min}} \left( \frac{1}{d(u)} + \frac{1}{d(v)} \right), \quad (1)$$

for any non-bipartite graph  $G$  and  $u, v \in V(G)$ , where  $d(v)$  is the degree of the vertex  $v$  and  $d_{\min} = \min_{v \in V} d(v)$  is the minimum degree. For the HRG with parameter  $\frac{1}{2} < \alpha < 1$ , with high probability,  $t_{\text{rel}} \geq n^{2\alpha-1}/(\log n)^{1+o(1)}$  [47] and the average degree is constant - thus (1) does not give a good bound.

Since their introduction in 2010 [52], hyperbolic random graphs have been studied by various authors. Apart from the results already mentioned (power-law degree distribution, short graph distances, strong clustering, giant component, spectral gap and diameter), connectivity was investigated by Bode et al. [12]. Further results exist on the number of  $k$ -cliques and the clique number [32], the existence of perfect matchings and Hamilton cycles [29], the tree-width [9] and sub-tree counts [65]. Two models, commonly considered closely related to the hyperbolic random graphs, are scale-free percolation [27] and geometric inhomogeneous random graphs [14].

Few random processes on HRGs have been rigorously studied. Among the notable exceptions is the work by Linker et al. [58] which studies the contact processes in the HRG model, bootstrap percolation by Candellero and Fountoulakis [15] and Marshall et al. [63], and, for geometric inhomogeneous random graphs, by Koch and Lengler [50]. Komjáthy and Lodewijks [51] studied first passage percolation on scale free spatial network models.

To the best of our knowledge, the only work that explicitly studies random walks that deals with (a more general model of) HRGs is the work by Cipriani and Salvi [19] on mixing time of scale-free percolation. However, some aspects of simple random walks have been analyzed on infinite versions of HRGs. Specifically, Heydenreich et al. study transience and recurrence of random walks in the scale-free percolation model [39] (also known as heterogeneous long-range percolation) which is a “lattice” version of the HRG model. For similar investigations, but for more general graphs on Poisson point processes, see [36]. Additionally, the first author, Linker, and Mitsche [45] have studied a dynamic variant of the HRG generated by stationary Brownian motions.

## 2 Preliminaries

In this section we introduce notation, define some objects and terms we will be working with, and collect, for future reference, some known results concerning them. We adopt some conventions in Section 2.1, we recall a large deviations bound in Section 2.2, then we formally define the HRG model in Section 2.3, we discuss random walks in Section 2.4 and electrical networks in Section 2.5.

### 2.1 Conventions

Throughout, we use standard notions and notation concerning the asymptotic behavior of sequences. If  $(a_n)_{n \in \mathbb{N}}, (b_n)_{n \in \mathbb{N}}$  are two sequences of real numbers, we write  $a_n = \mathcal{O}(b_n)$  to denote that for some constant  $C > 0$  and  $n_0 \in \mathbb{N}$  it holds that  $|a_n| \leq C|b_n|$  for all  $n \geq n_0$ . Also, we write  $a_n = \Omega(b_n)$  if  $b_n = \mathcal{O}(a_n)$ , and  $a_n = \Theta(b_n)$  if  $a_n = \mathcal{O}(b_n)$  and  $a_n = \Omega(b_n)$ .

Unless stated otherwise, all asymptotics are as  $n \rightarrow \infty$  and all other parameters are assumed fixed. Expressions given in terms of other variables that depend on  $n$ , for example  $\mathcal{O}(R)$ , are still asymptotics with respect to  $n$ . As we are interested in asymptotics, we only claim and prove inequalities for  $n$  sufficiently large. So, for simplicity, we always assume  $n$  sufficiently large. For example, we may write  $n^2 > 5n$  without requiring  $n > 5$ .

An event, more precisely a family of events parameterized by  $n \in \mathbb{N}$ , is said to hold *with high probability (w.h.p.)*, if for every  $c > 0$  the event holds with probability at least  $1 - \mathcal{O}(n^{-c})$ .

We shall follow standard notation, such as denoting the vertex and edge sets of  $G$  by  $V(G)$  and  $E(G)$ , respectively. We use  $d_G(u, v)$  to denote the graph distance between two vertices  $u, v \in V(G)$ , let  $N(v) := \{u \in V \mid d_G(u, v) = 1\}$  denote the neighbourhood of a vertex, and let  $d(v) := |N(v)|$ .

## 2.2 Poisson Random Variables

We will be working with a Poissonized model, where the number of points within a given region is Poisson-distributed. Thus, we will need some elementary results for Poisson random variable. The first is a (Chernoff) large deviation bound.

► **Lemma 4.** *Let  $P$  have a Poisson distribution with mean  $\mu$ . The following holds*

- (i)  $\mathbb{P}(P \leq \frac{1}{2}\mu) \leq e^{-\frac{1}{8}\mu}$ .
- (ii) *If  $\delta \geq e^{\frac{3}{2}}$ , then  $\mathbb{P}(P \geq \delta\mu) \leq e^{-\frac{1}{2}\delta\mu}$ .*

Several times, when bounding various expectations, we use the following crude but useful bound on the raw moments of Poisson random variables.

► **Lemma 5.** *Let  $X$  be a Poisson random variable with mean  $\mu$ . Then, for any real  $\kappa \geq 1$ , we have  $\mathbb{E}(X^\kappa) \leq \mu^\kappa \cdot (40 \cdot \min\{\frac{\kappa}{5\mu}, 1\})^\kappa$ .*

## 2.3 The HRG model

We represent the hyperbolic plane (of constant Gaussian curvature  $-1$ ), denoted  $\mathbb{H}^2$ , by points in  $\mathbb{R}^2$ . Elements of  $\mathbb{H}^2$  are referred to by the polar coordinates  $(r, \theta)$  of their representation as points in  $\mathbb{R}^2$ . The point with coordinates  $(0, 0)$  will be called the *origin* of  $\mathbb{H}^2$  and denoted  $O$ . When alluding to a point  $u \in \mathbb{H}^2$  we denote its polar coordinates by  $(r_u, \theta_u)$ . The hyperbolic distance  $d_{\mathbb{H}^2}(u, v)$  between two points  $u, v \in \mathbb{H}^2$  is determined via the Hyperbolic Law of Cosines as the unique solution of

$$\cosh d_{\mathbb{H}^2}(u, v) = \cosh r_u \cosh r_v - \sinh r_u \sinh r_v \cos(\theta_u - \theta_v).$$

In particular, the hyperbolic distance between the origin and a point  $u \in \mathbb{H}^2$  equals  $r_u$ . For a point  $p \in \mathbb{H}^2$  the ball of radius  $\rho > 0$  centered at  $p$  will be denoted  $B_p(\rho)$ , i.e.,

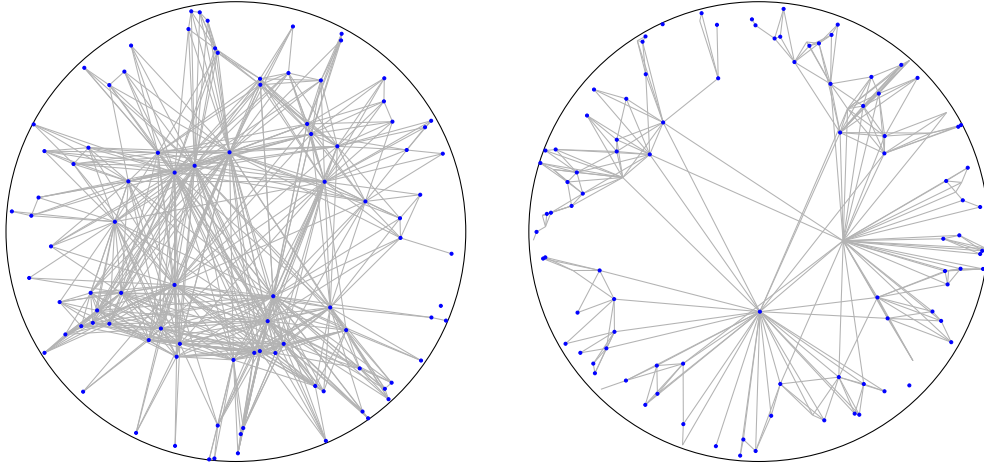
$$B_p(\rho) := \{q \in \mathbb{H}^2 \mid d_{\mathbb{H}^2}(p, q) < \rho\}.$$

We will work in the Poissonized version of the HRG model which we describe next. For a positive integer  $n$  and positive constant  $\nu$  we consider a Poisson point process on the hyperbolic disk centered at the origin  $O$  and of radius  $R := 2 \ln(n/\nu)$ . The intensity function at polar coordinates  $(r, \theta)$  for  $0 \leq r < R$  and  $\theta \in \mathbb{R}$  equals

$$\lambda(r, \theta) := \nu e^{\frac{R}{2}} f(r, \theta) = n f(r, \theta) \tag{2}$$

where  $f(r, \theta)$  is the joint density function of independent random variables  $\theta$  and  $r$ , with  $\theta$  chosen uniformly at random in  $[0, 2\pi)$  and  $r$  chosen according to the following density function:

$$f(r) := \frac{\alpha \sinh(\alpha r)}{\cosh(\alpha R) - 1} \cdot \mathbf{1}_{[0, R)}(r) \quad \text{where } \mathbf{1}_{[0, R)}(\cdot) \text{ is the indicator of } [0, R).$$



■ **Figure 1** Instances of  $\mathcal{G}_{\alpha, \nu}(n)$  for  $n = 100$ ,  $\nu \approx 1.832$ ,  $\alpha = 0.6$  (left) and  $\alpha = 0.9$  (right).

The parameter  $\alpha > 0$  controls the distribution: For  $\alpha = 1$ , the distribution is uniform in  $\mathbb{H}^2$ , for smaller values the vertices are distributed more towards the center of  $B_O(R)$  and for bigger values more towards the border. (See Figure 1 for an illustration of instances of  $\mathcal{G}_{\alpha, \nu}(n)$  for two distinct values of  $\alpha$ .)

We shall need the following useful approximation to the density  $f(\cdot)$ .

► **Lemma 6** ([33, Equation (3)]).  $f(r) = \alpha e^{-\alpha(R-r)} \cdot (1 + \Theta(e^{-\alpha R} + e^{-2\alpha r})) \cdot \mathbf{1}_{[0, R)}(r)$ .

We denote the point set of the Poisson process by  $V$  and we identify elements of  $V$  with the vertices of a graph whose edge set  $E$  is the collection of vertex pairs  $uv$  such that  $d_{\mathbb{H}^2}(u, v) < R$ . The probability space over graphs  $(V, E)$  thus generated is denoted by  $\mathcal{G}_{\alpha, \nu}(n)$  and referred to as the HRG. Note in particular that  $\mathbb{E}|V| = n$  since

$$\int_{B_O(R)} \lambda(r, \theta) d\theta dr = \nu e^{\frac{R}{2}} \int_0^\infty f(r) dr = n.$$

The parameter  $\nu$  controls the average degree of  $\mathcal{G}_{\alpha, \nu}(n)$  which, for  $\alpha > \frac{1}{2}$ , is  $(1 + o(1)) \frac{2\alpha^2 \nu}{(\alpha - 1/2)^2}$  (see [37, Theorem 2.3]).

The Hyperbolic Law of Cosines turns out to be complicated to work with when computing distances in hyperbolic space. Instead, it is more convenient to consider the maximum angle  $\theta_R(r_u, r_v)$  that two points  $u, v \in B_O(R)$  can form with the origin  $O$  and still be within (hyperbolic) distance at most  $R$  provided  $u$  and  $v$  are at distance  $r_u$  and  $r_v$  from the origin, respectively.

► **Remark 7.** Replacing in (7) the terms  $d_{\mathbb{H}^2}(u, v)$  by  $R$  and  $\theta_u - \theta_v$  by  $\theta_R(r_u, r_v)$ , taking partial derivatives on both sides with respect to  $r_u$  and some basic arithmetic gives that the mapping  $r_u \mapsto \theta_R(r_u, r_v)$  is continuous and strictly decreasing in the interval  $[0, R)$ . Since  $\theta_R(r_u, r_v) = \theta_R(r_v, r_u)$ , the same is true of the mapping  $r_v \mapsto \theta_R(r_u, r_v)$ . (See [48, Remark 2.1] for additional details.)

The following estimate of  $\theta_R(r, r')$ , due to Gugelmann, Panagiotou and Peter is very useful and accurate (especially when  $R - (r + r')$  goes to infinity with  $n$ ).

► **Lemma 8** ([37, Lemma 6]). *If  $0 \leq r \leq R$  and  $r + r' \geq R$ , then  $\theta_R(r, r') = 2e^{\frac{1}{2}(R-r-r')}(1 + \Theta(e^{R-r-r'}))$ .*



We will need estimates for measures of regions of the hyperbolic plane, and specifically for the measure of balls. We denote by  $\mu(S)$  the measure of a set  $S \subseteq \mathbb{H}^2$ , i.e.,  $\mu(S) := \int_S f(r, \theta) dr d\theta$ . The following approximation of the measures of the ball of radius  $\rho$  centered at the origin and the ball of radius  $R$  centered at  $p \in B_O(R)$ , both also due to Gugelmann et al., will be used frequently in our analysis.

► **Lemma 9** ([37, Lemma 7]). *For  $\alpha > \frac{1}{2}$ ,  $p \in B_O(R)$  and  $0 \leq r \leq R$  we have*

$$\mu(B_O(r)) = e^{-\alpha(R-r)}(1 + o(1)),$$

$$\mu(B_p(R) \cap B_O(R)) = \frac{2\alpha e^{-\frac{1}{2}r_p}}{\pi(\alpha - \frac{1}{2})} (1 + \mathcal{O}(e^{-(\alpha - \frac{1}{2})r_p} + e^{-r_p})).$$

Next, we state a result that is implicit in [11] (later refined in [30]) concerning the size and the “geometric location” of the giant component of  $\mathcal{G}_{\alpha, \nu}(n)$ . First, observe that the set of vertices of  $\mathcal{G}_{\alpha, \nu}(n)$  that are inside  $B_O(\frac{R}{2})$  are within distance at most  $R$  of each other. Hence, they form a clique and in particular belong to the same connected component. The graph induced by the connected component of  $\mathcal{G}_{\alpha, \nu}(n)$  to which the vertices in  $B_O(\frac{R}{2})$  belong will be referred to as the *center component* of  $\mathcal{G}_{\alpha, \nu}(n)$ .

► **Theorem 10** ([11, Theorem 1.4], [48, Theorem 1.1]). *If  $\frac{1}{2} < \alpha < 1$ , then a.a.s. the center component of  $\mathcal{G}_{\alpha, \nu}(n)$  has size  $\Theta(n)$  and all other connected components of  $\mathcal{G}_{\alpha, \nu}(n)$  are of size polylogarithmic in  $n$ .*

The previous result partly explains why we focus our attention on simple random walks in the center component of  $\mathcal{G}_{\alpha, \nu}(n)$ . In the following remark we justify formally why we, henceforth, consider both the giant and the center component as being the same component, and consequently denote both of them by  $\mathcal{C}_{\alpha, \nu}(n)$ .

► **Remark 11.** Let  $\mathcal{S}_n$  be the event that the giant (the largest component) is equal to the center component, then  $\mathbb{P}(\mathcal{S}_n) = 1 - e^{-\Omega(n)}$  by [11]. It follows immediately that all of our results holding a.a.s. for the center component also hold for the giant component. For statements of the form  $\mathbb{E}(X(\mathcal{C}))$ , where  $X(G)$  is a function of a graph satisfying  $1 \leq X(G) \leq |V(G)|^\kappa$  for some fixed  $\kappa > 0$ , for example (non-trivial) cover/hitting times, the results also carry over. That is, if  $\mathcal{C}'$  is the giant of the HRG then  $\mathbb{E}(X(\mathcal{C}')) = (1 + o(1))\mathbb{E}(X(\mathcal{C}))$ . To see this, since  $\mathbb{E}(|V(\mathcal{G})|^\kappa) = \mathcal{O}(n^\kappa)$  by Lemma 5, we have the following by Cauchy–Schwartz,

$$\mathbb{E}(X(\mathcal{C}')) = \mathbb{E}(X(\mathcal{C}')\mathbf{1}_{\mathcal{S}_n}) + \sqrt{\mathbb{E}|V(\mathcal{C}')|^{2\kappa} \cdot \mathbb{P}(\mathcal{S}_n^c)} \leq \mathbb{E}(X(\mathcal{C})) + n^\kappa e^{-\Omega(n)} = (1 + o(1))\mathbb{E}(X(\mathcal{C})).$$

This also holds with the roles of  $\mathcal{C}'$  and  $\mathcal{C}$  reversed, giving the result.

The condition  $\alpha > \frac{1}{2}$  guarantees that  $\mathcal{G}_{\alpha, \nu}(n)$  has constant average degree depending on  $\alpha$  and  $\nu$  only [37, Theorem 2.3]. If on the contrary  $\alpha \leq \frac{1}{2}$ , then the average degree grows with  $n$ . If  $\alpha > 1$ , the largest component of  $\mathcal{G}_{\alpha, \nu}(n)$  is sublinear in  $n$  [11, Theorem 1.4]. For  $\alpha = 1$  whether the largest component is of size linear in  $n$  depends on  $\nu$  [11, Theorem 1.5]. Hence, the parameter range where  $\frac{1}{2} < \alpha < 1$  is where HRGs are always sparse, exhibit power law degree distribution with exponent between 2 and 3 and the giant component is, a.a.s., unique and of linear size. All these are characteristics ascribed to so called “social” or “complex networks” which HRGs purport to model. Our main motivation is to contribute to understand processes that take place in complex networks, many of which, as already discussed in the introduction, either involve or are related to simple random walks on such networks. Thus, we restrict our study exclusively to the case where  $\frac{1}{2} < \alpha < 1$ , but in order to avoid excessive repetition, we omit this condition from the statements we establish.

The last known property of HRGs that we recall is that, w.h.p. the center component has a linear in  $n$  number of edges.

► **Lemma 12** ([47, Lemma 15]). For  $\frac{1}{2} < \alpha < 1$ , w.h.p.  $|E(\mathcal{C}_{\alpha,\nu}(n))|$  is  $O(n)$ .

To conclude this section we point out that everything we do throughout this paper can be easily adapted to the case where  $\mathbb{H}^2$  has Gaussian curvature  $-\eta^2$  instead of  $-1$  and all stated results hold provided  $\alpha$  is replaced by  $\alpha/\eta$  everywhere.

## 2.4 Random Walks

The simple random walk  $(X_t)_{t \geq 0}$  on a graph  $G = (V, E)$  is a discrete time random process on  $V$  which at each time moves to a neighbour of the current vertex  $u \in V$  uniformly with probability  $1/d(u)$ . We use  $\mathbf{P}(\cdot) := \mathbf{P}^G(\cdot)$  to denote the law of the random walk on a graph  $G$  (as opposed to  $\mathbb{P}$  for the random graph). For a probability distribution  $\mu$  on  $V$  we let  $\mathbf{P}_\mu^G(\cdot) := \mathbf{P}^G(\cdot \mid X_0 \sim \mu)$  be the conditional probability of the walk on  $G$  started from a vertex sampled from  $\mu$ . In the case where the walk starts from a single vertex  $u \in V$  we write  $u$  instead of  $\mu$ , for example  $\mathbf{E}_u^G(\cdot) := \mathbf{E}^G(\cdot \mid X_0 = u)$ . We shall drop the superscript  $G$  when the graph is clear from the context. We now define the *cover time*  $t_{\text{cov}}(G)$  of  $G$  by

$$t_{\text{cov}}(G) := \max_{u \in V} \mathbf{E}_u^G(\tau_{\text{cov}}), \quad \text{where} \quad \tau_{\text{cov}} := \inf \left\{ t \geq 0 : \bigcup_{i=0}^t \{X_i\} = V(G) \right\}.$$

Similarly, for  $u, v \in V$  we let  $\mathbf{E}_u(\tau_v)$ , where  $\tau_v := \inf\{t \geq 0 \mid X_t = v\}$ , be the *hitting time* of  $v$  from  $u$ . We shall use  $\pi$  to denote the *stationary distribution* of a single random walk on a connected graph  $G$ , given by  $\pi(v) := \frac{d(v)}{2|E|}$  for  $v \in V$ . Let

$$t_{\text{hit}}(G) := \max_{u, v \in V} \mathbf{E}_u^G(\tau_v), \quad \text{and} \quad t_{\odot}(G) := \sum_{u, v \in V(G)} \mathbf{E}_u^G(\tau_v) \pi(u) \pi(v),$$

denote the *maximum hitting time* and the *target time*, respectively. We define each of these times to be 0 if  $G$  is either the empty graph or consists of just a single vertex. The target time  $t_{\odot}(G)$ , also given by  $\mathbf{E}_\pi^G(\tau_\pi)$ , is the expected time for a random walk to travel between two vertices sampled independently from the stationary distribution [56, Section 10.2]. We will consider the random walk on the center component  $\mathcal{C} := \mathcal{C}_{\alpha,\nu}(n)$  of the HRG and so each of the expected stopping times  $t_{\text{cov}}(\mathcal{C})$ ,  $t_{\text{hit}}(\mathcal{C})$  and  $t_{\odot}(\mathcal{C})$  will in fact be random variables. We shall also refer to  $\mathbf{E}_u(\tau_v) + \mathbf{E}_v(\tau_u)$  as the *commute time* between the vertices  $u, v$ .

## 2.5 Electrical Networks & Effective Resistance

An electrical network,  $N := (G, C)$ , is a graph  $G$  and an assignment of conductances  $C : E(G) \rightarrow \mathbb{R}^+$  to the edges of  $G$ . For an undirected graph  $G$  we define  $\vec{E}(G) := \{\vec{xy} \mid xy \in E(G)\}$  as the set of all possible oriented edges for which there is an edge in  $G$ . For some  $S, T \subseteq V(G)$ , a *flow* from  $S$  to  $T$  (or  $(S, T)$ -*flow*, for short) on  $N$  is a function  $f : \vec{E}(G) \rightarrow \mathbb{R}$  satisfying  $f(\vec{xy}) = -f(\vec{yx})$  for every  $xy \in E(G)$  as well as Kirchoff's node law for every vertex apart from those that belong to  $S$  and  $T$ , i.e.

$$\sum_{u \in N(v)} f(\vec{uv}) = 0 \quad \text{for each } v \in V \setminus (S \cup T).$$

A flow from  $S$  to  $T$  is called a *unit flow* if, in addition, its strength is 1, that is,

$$\sum_{s \in S} \sum_{u \in N(s)} f(\vec{su}) = 1.$$

We say that the  $(S, T)$ -flow is *balanced* if

$$\sum_{u \in N(s)} f(\vec{s}u) = \sum_{u \in N(s')} f(\vec{s}'u) \text{ and } \sum_{u \in N(t)} f(u\vec{t}) = \sum_{u \in N(t')} f(u\vec{t}') \text{ for all } s, s' \in S \text{ and } t, t' \in T.$$

When  $S = \{s\}$  and  $T = \{t\}$  we write  $(s, t)$ -flow instead of  $(\{s\}, \{t\})$ -flow. Note that  $(s, t)$ -flows are always balanced. Given two flows  $f$  and  $g$  on  $N := (G, C)$ , we say that  $g$  can be concatenated to  $f$  if  $f + g$  is a flow on  $N$  and for every oriented edge  $\vec{e} \in \vec{E}$  either  $f(\vec{e})$  and  $g(\vec{e})$  are both 0, or they have opposite signs, so  $(f(\vec{e}) + g(\vec{e}))^2 \leq (f(\vec{e}))^2 + (g(\vec{e}))^2$ .

For the network  $N := (G, C)$  and a flow  $f$  on  $N$  define the *energy dissipated* by  $f$ , denoted  $\mathcal{E}(f)$ , by

$$\mathcal{E}(f) := \sum_{e \in \vec{E}(G)} \frac{f(e)^2}{2C(e)}, \quad (3)$$

For future reference, we state the following easily verified fact:

► **Claim 13.** Let  $N := (G, C)$  be an electrical network and  $S, M, T \subseteq V(G)$ . If  $f$  and  $g$  are balanced  $(S, M)$  and  $(M, T)$  flows on  $N$ , respectively, and  $g$  can be concatenated to  $f$ , then  $f + g$  is a balanced  $(S, T)$ -flow on  $N$  and  $\mathcal{E}(f + g) \leq \mathcal{E}(f) + \mathcal{E}(g)$ . Moreover, if  $f$  and  $g$  are unit flows, so is  $f + g$ .

For  $S, T \subseteq V(G)$ , the *effective resistance* between  $S$  and  $T$ , denoted  $\mathcal{R}_C(S \leftrightarrow T)$ , is

$$\mathcal{R}_C(S \leftrightarrow T) := \inf \{ \mathcal{E}(f) \mid f \text{ is a unit flow from } S \text{ to } T \}. \quad (4)$$

The set of conductances  $C$  defines a reversible Markov chain [60, Chapter 2]. In this paper we shall fix  $C(e) = 1$  for all  $e \in E(G)$  as this corresponds to a simple random walk. In this case, we write  $\mathcal{R}_G(S \leftrightarrow T)$  (or  $\mathcal{R}(S \leftrightarrow T)$  if the graph is clear) instead of  $\mathcal{R}_C(S \leftrightarrow T)$ . The following is well known.

► **Proposition 14** ([56, Corollary 10.8]). *The effective resistances form a metric space on any graph, in particular they satisfy the triangle inequality.*

Choosing a single shortest path  $P$  between any two vertices  $s, t$  (if one exists) in a network (with  $C(e) = 1$  for each  $e \in E$ ) and routing a unit flow down the edges of  $P$  we obtain, straight from the definition (4) of  $\mathcal{R}(s \leftrightarrow t)$ , the following basic but useful result.

► **Lemma 15** ([16, Lemma 3.2]). *For any graph  $G$  and  $s, t \in V(G)$ , we have  $\mathcal{R}(s \leftrightarrow t) \leq d_G(s, t)$ .*

Another very useful tool is Rayleigh's monotonicity law (RML).

► **Theorem 16** (Rayleigh's Monotonicity Law [56, Theorem 9.12]). *Let  $\{C(e)\}_{e \in E}$  and  $\{C'(e)\}_{e \in E}$  be sets of conductances on the edges of the same graph  $G = (V, E)$ . If  $C(e) \geq C'(e)$  for all  $e \in E$ , then*

$$\mathcal{R}_C(S \leftrightarrow T) \leq \mathcal{R}_{C'}(S \leftrightarrow T) \quad \text{for all } S, T \subseteq V.$$

The *Kirchhoff index*  $\mathcal{K}(G)$  of a graph  $G$  is the sum of resistances in the graph, that is

$$\mathcal{K}(G) = \sum_{u, v \in V(G)} \mathcal{R}(u \leftrightarrow v).$$

The Kirchhoff index has applications in mathematical chemistry, see [66] and citing papers.

The following result allows us to relate hitting times to effective resistance.

► **Lemma 17** ([56, Proposition 10.6]). *For any graph  $G$  and any pair of vertices  $u, v \in V(G)$  we have*

$$\mathbf{E}_u(\tau_v) + \mathbf{E}_v(\tau_u) = 2|E(G)| \cdot \mathcal{R}(u \leftrightarrow v). \quad (\text{Commute time identity})$$

### 3 Overview of the Proofs of the Main Results

Owing to space we give a brief (heavily abridged, non-rigorous) overview of the ideas used in the proofs of the main theorems from Section 1.1. Proofs of all claims can be found in the full version of this paper [49].

#### 3.1 Theorems 1 & 2: Average Effective Resistance & Target Time

To control the resistance  $\mathcal{R}(s \leftrightarrow t)$  between two vertices  $s$  and  $t$  of the center component  $\mathcal{C}_{\alpha,\nu}(n)$  we bound the energy dissipated by a carefully designed  $(s, t)$ -flow  $f_{s,t}$  associated to a tiling of the hyperbolic plane on which we overlay a forest-like structure. We shall first describe the tiling, then the flow, before explaining how to bound the resistance from this flow.

##### Tiling

We define a set of tiles  $\{T_{i,j}\}_{i,j \geq 0}$  of the hyperbolic plane  $\mathbb{H}^2$  centered around the origin  $O$ . This tiling is spherical in nature (see Figure 2) and, roughly speaking, tile  $T_{i,j}$  is  $i$  tiles from the origin (we say it is at *level*  $i$ ) and it is the  $j^{\text{th}}$  tile, at level  $i$ , when going clockwise from a fixed ray emanating from  $O$ , at 0 degrees “north”. A region of  $\mathbb{H}^2$  between two rays emanating from the origin  $O$  will be called a *sector*, and refer to the angle of the sector as the (smallest) angle between the two rays.

There will be a distinguished collection of tiles, called *root* tiles  $\{T_{0,j}\}_{j \in N_0}$ , corresponding to the elements of the equipartition of  $B_O(\frac{R}{2})$  into  $N_0 = \Theta(1)$  sectors, hence each sector has angle  $\theta_0 = 2\pi/N_0$ . We then define three sequences  $N_i$ ,  $\theta_i$  and  $h_i$ , for  $i \in \mathbb{N}$  where  $N_i := 2^i N_0$ ,  $\theta_i := 2\pi/N_i$ , and (very roughly speaking)  $h_0 = R/2$  and  $h_i \approx R/2 + i \cdot \ln 2$ . The rest of the tiling is specified by setting  $T_{i,j}$  to be the region at distance between  $h_{i-1}$  and  $h_i$  from  $O$  that lies in the smallest sector between rays at angles  $\theta_i \cdot j$  and  $\theta_i \cdot (j+1)$ . Thus each sector has central angle  $\theta_i$  and there is a total of  $N_i = 2N_{i-1}$  sectors for a given  $i$ , see Figure 2.

We say that  $T_{i,j}$  is the *parent* tile of both  $T_{i+1,2j}$  and  $T_{i+1,2j+1}$  and refer to the latter two tiles as *children* of tile  $T_{i,j}$  (root tiles are assumed to be their own parent). For a tile  $T_{i,j}$  we refer to the tiles of height  $i' \leq i$  that intersect a ray from  $O$  that passes through  $T_{i,j}$  as the *ancestors* of  $T_{i,j}$ . A tile  $T$  will be said to be a *descendant* of another tile  $T'$  if the latter is an ancestor of the former. Given a tile  $T_{i,j}$  let  $\{T_{i,j}^0, T_{i,j}^1\}$  be the “natural” equipartition of  $T_{i,j}$  into two tiles along a ray from  $O$  through the center of  $T_{i,j}$ . We refer to  $T_{i,j}^0, T_{i,j}^1$  as the *half-tiles* of  $T_{i,j}$  and say  $T_{i,j}^0$  is the *twin* of  $T_{i,j}^1$ , and vice versa. Given a tile  $T$  we call  $H$  the *parent half-tile* of  $T$  if it is a half-tile of the parent of  $T$  and a line segment from the origin to any point in the interior of  $T$  intersects  $H$ .

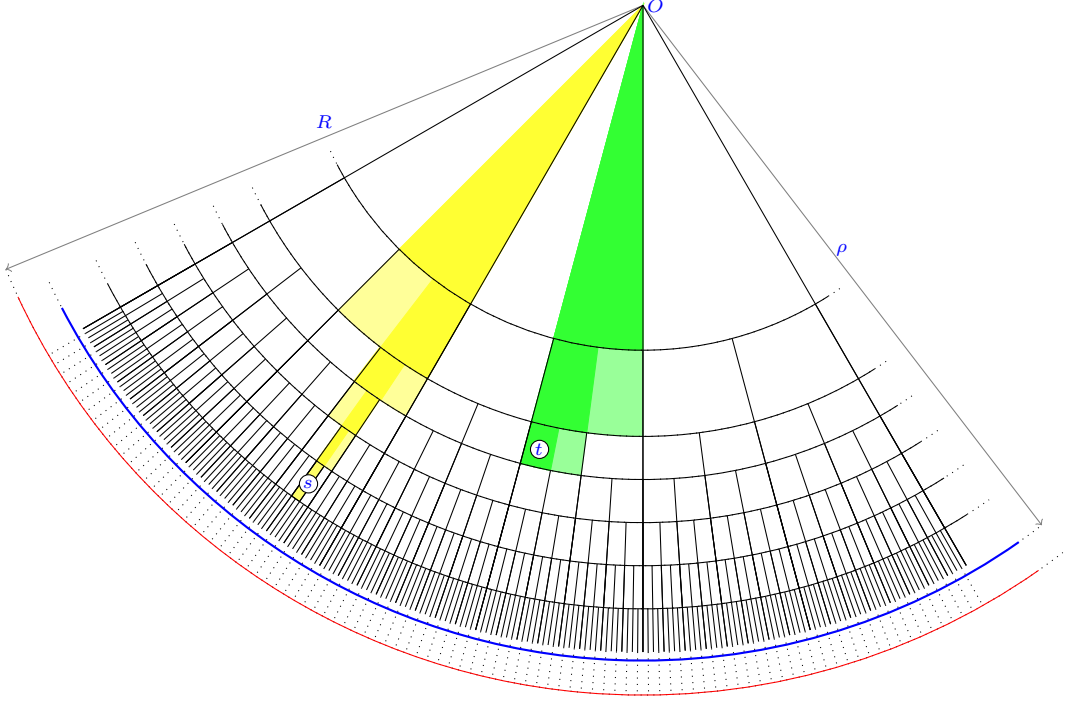
Recall that two points at distance at most  $R$  are connected by an edge of  $\mathcal{G}_{\alpha,\nu}(n)$ . The sequences  $N_i, \theta_i$  and  $h_i$  are chosen in such a way so that the tiling satisfies the following two properties.

- (i) Two points in a given tile are within distance at most  $R$  of each other.
- (ii) Any point in a tile is within distance at most  $R$  from any point in its parent half-tile.

These two properties allow us too describe a flow based on how it moves between blocks, rather than getting bogged down with specific vertices.

##### Comparison with a Tiling Due To Fountoulakis and Müller [30]

In this article we work in the so called *Gans model* [34] or *native model* [52] of hyperbolic space, in contrast to [30] where the *Poincaré half-plane model* is used. Although the two tilings are approximately equal, ours is a partition of  $\mathbb{R}^2$  instead of the half-plane, i.e., each



■ **Figure 2** (a) Partial illustration of a tiling of  $B_O(R)$  (not at scale). (b) Depiction of flow between vertices  $s$  and  $t$  with no common ancestor tile. Coloured regions contain vertices through which flow from  $s$  to  $t$  is routed. Flow is pushed radially towards the origin  $O$  from a yellow shaded tile to its parent half-tile. Flow is pushed in an angular direction from dark to light yellow shaded half-tiles. The direction of flow is reversed in green shaded region.

tiling partitions the set used to represent hyperbolic space. Since both representations are alternative models of  $\mathbb{H}^2$ , both tilings can be cast in one or the other. Doing so, one gets that the similarity of both tilings increases the further towards infinity their tiles are, i.e., further from the origin in the Gans model and closer to the half-plane boundary in the Poincaré half-plane model. For points close to the origin in the former or far from the boundary in the latter representation, both tilings differ significantly, although this is irrelevant for the analyses performed either here or in [30, 64]. However, working with our tiling avoids having to perform, as in [30], a coupling between the HRG and a point process in the half-plane and also avoids some of the approximations incurred by working with the idealized model of Müller and Staps [64]. We believe this explains why we can guarantee that most of our results hold with probability  $1 - \mathcal{O}(n^{-c})$  for all  $c > 0$  instead of the same probability but just for some  $c > 0$ .

### Definition of the Flow

We now sketch the construction of our unit  $(s, t)$ -flow, between two distinct vertices  $s$  and  $t$  of  $\mathcal{G}_{\alpha, \nu}(n)$ . The energy dissipated by this flow yields bounds on the effective resistance between  $s$  and  $t$ .

The flow  $f_{s,t}$  is built up as a sum of five separate flows, that is

$$f_{s,t} := f_s + f_s^t + f^{s,t} + f_t^s + f_t. \quad (5)$$

The rough idea is to find the tile  $T$  that is a common ancestor of  $s$  and  $t$  at furthest distance from  $O$  and create flows to this tile from  $s$ , and from this tile to  $t$ . If  $s$  and  $t$  have no common ancestor then the flows meet up in the center  $B_O(\frac{R}{2})$  (which is a clique), with the centre essentially taking the place of  $T$ . The first term  $f_s$  in (5) spreads flow from  $s$  evenly across the half-tile it is contained in, likewise  $f_t$  collects flow from vertices in the half-tile containing  $t$  and sends it to  $t$ . The term  $f_s^t$  moves flow towards the centre by first moving flow from the current half-tile to its twin, then from the current (full) tile to its parent half-tile, and repeating like this until reaching the half-tile of the common ancestor (or a root half-tile in the center  $B_O(\frac{R}{2})$ ). The term  $f_s^t$  does the same in reverse, taking flow out from the center to  $t$ , see the yellow and green parts of Figure 2 for an illustration. Finally the term  $f^{s,t}$  moves the flow from the half-tile in the common ancestor in the ray that intersects  $s$  to its twin (or across root tiles), this flow is zero if  $s$  and  $t$  lie in a ray from  $O$ .

The main ideas of this construction are that it spreads the flow over as evenly as possible over the vertices in each tile. Its modular construction also makes it easy to analyse, in particular when bounding the energy dissipated. One can show that if each half-tile encountered in the above sketched construction of  $f_{s,t}$  is non-empty, then  $f_{s,t}$  indeed gives a valid  $(s, t)$ -flow. Moreover, since this flow is balanced and the parts can be concatenated, we have

$$\mathcal{E}(f_{s,t}) \leq \mathcal{E}(f_s) + \mathcal{E}(f_s^t) + \mathcal{E}(f^{s,t}) + \mathcal{E}(f_t^s) + \mathcal{E}(f_t).$$

### Validity and Energy of the Flow

At this juncture we turn our attention to determining conditions under which  $f_{s,t}$  exists (i.e., every tile contains a vertex), and more importantly is a good flow in the sense that it dissipates a small amount of energy. Clearly, the larger the number of vertices in each half-tile, the smaller the energy dissipated by the flow.

We say that a half-tile  $H$  is *sparse* if the number of vertices it contains is fewer than half the ones expected, i.e., if  $|V \cap H| < \frac{1}{2}\mathbb{E}|V \cap H|$ . We say that a tile  $T$  is *faulty* if either one of its two associated half-tiles is sparse. For  $C > 0$  a large constant to be determined, let

$$\rho := R - \frac{\ln(C\frac{R}{\nu})}{1 - \alpha}. \quad (6)$$

Using standard arguments concerning Poisson point processes we argue that,

$$\text{w.h.p., none of the tiles } T \text{ contained in } B_O(\rho) \text{ are faulty.} \quad (7)$$

We say that a tile  $T$  is *robust* if none of its ancestors (including itself) is faulty. Thus, (7) implies that w.h.p. every tile  $T$  contained in  $B_O(\rho)$  is robust. The condition  $T \subseteq B_O(\rho)$  cannot be relaxed significantly, so we will have to settle for a weaker statement. For  $C' > 0$ , let

$$\rho' := R - \frac{\ln(\frac{2C'}{\nu})}{1 - \alpha}. \quad (8)$$

Thus some points in  $B_O(\rho')$  are only a constant distance from the boundary. We show that

$$\text{a tile contained in } B_O(\rho') \text{ has at least a constant probability of being robust.} \quad (9)$$

We finally establish the following:

$$\text{if } s \text{ and } t \text{ belong to robust tiles then } f_{s,t} \text{ is a unit } (s, t)\text{-flow and } \mathcal{E}(f_{s,t}) = \mathcal{O}(1). \quad (10)$$

### Bounding Average Resistance and Target Time

Together, (9) and (10) show that a constant fraction of pairs of points in  $B_O(\rho')$  have bounded resistance between them. We are yet far from done, as a significant fraction of the vertices of the giant component of  $\mathcal{G} := \mathcal{G}_{\alpha,\nu}(n)$  fall outside the ball  $B_O(\rho')$ . For any vertex  $w \in V(\mathcal{G})$  let  $\Upsilon(w)$  be the smallest sector containing  $w$  whose closure contains vertices in  $V(\mathcal{G}) \cap (B_O(\rho') \setminus B_O(\rho' - \ln 2))$  both clockwise and anti-clockwise of  $w$  that reside in robust tiles. We then prove the following.

Let  $w$  be a vertex in the giant and  $w' \in V(\mathcal{G}) \cap B_O(\rho')$  be a robust vertex which is closest (in graph distance) to  $w$ . Then,  $d(w, w') \leq 1 + |V(\mathcal{G}) \cap \Upsilon(w)|$ . (11)

By the triangle inequality (as  $\mathcal{R}(\cdot \leftrightarrow \cdot)$  is a metric), and (11) and (10), for any  $s, t$  in the giant

$$\mathcal{R}(s \leftrightarrow t) \leq |V(\mathcal{G}) \cap \Upsilon(s)| + |V(\mathcal{G}) \cap \Upsilon(t)| + \mathcal{O}(1). \quad (12)$$

Armed with (12) we can now bound the resistance between  $s$  and  $t$  by bounding the number of vertices in the smallest sectors containing  $s$  and  $t$  defined by rays through robust vertices. We prove some further (more technical) results in the spirit of (9) that give improved bounds on the probability of a vertex being robust as a function of its location. Using these bounds, together with some elaborate arguments discussed in the full version of this article [49], for a vertex  $w \in V(\mathcal{G}) \setminus B_O(\rho)$  we can bound the tails of  $|V(\mathcal{G}) \cap \Upsilon(w)|$  by a stretched exponential function. That is, roughly speaking, we can show that there exists constants  $\kappa_1, \kappa_2 > 0$  such that for any  $w \in V(\mathcal{G}) \setminus B_O(\rho)$  and  $p \in \mathbb{N} \setminus \{0\}$ ,

$$\mathbb{P}(|V(\mathcal{G}) \cap \Upsilon(w)| \geq \kappa_1 \cdot 2^p) \leq \exp(-2^{\kappa_2 \cdot p}). \quad (13)$$

Applying the Campbell-Mecke formula [55, Theorem 4.4], a powerful tool from point process theory expressing expectations of functions of point processes as integrals of their mean measure, with (13) we show that  $\mathbb{E}(\sum_{w \in V(\mathcal{G}) \setminus B_O(\rho)} |V(\mathcal{G}) \cap \Upsilon(w)|^\kappa) = \mathcal{O}(n)$  for any fixed real  $\kappa \geq 1$ . Taking  $\kappa = 1$  now gives Theorem 1. With some additional effort and a bound on the raw moments of degrees in the HRG we can also prove Theorem 2 using the commute time identity and Hölders inequality.

### 3.2 Theorem 3: Max Hitting and Cover Times

Due to the well known bound  $\mathbf{E}_x[\tau_y] \leq d(x, y) \cdot 2|E|$ , we obtain  $t_{\text{hit}} = \mathcal{O}(n \log n)$  since a.a.s. the diameter is  $\Theta(\log n)$  and there are  $\Theta(n)$  many edges. A bound of  $\mathcal{O}(n \log^2 n)$  on the cover time then follows from Matthews bound.

The lower bounds are significantly more demanding and the basic Matthews lower bound gives a bound on the cover time that is a polynomial factor off the upper bound. We establish the following result which is implicit in [46]:

w.h.p. there are  $n^{\Omega(1)}$  maximal dangling paths of length at least  $\Omega(\ln n)$  in  $\mathcal{C}_{\alpha,\nu}(n)$ , (14)

and since the resistance between any pair of endpoints of the paths in (14) is  $\Omega(\log n)$ , we deduce that the commute time is  $\Omega(n \log n)$ . A refinement of Matthews bound due to Kahn, Kim, Lovász and Vu [42, Theorem 1.3] then gives the desired bound on the cover time from which the claimed lower bound on the hitting time immediately follows.



## 4 Concluding remarks

In this paper we determined the expected order of the maximum hitting time, cover time, target time and effective resistance between two uniform vertices, with the first two results also holding a.a.s. (w.r.t. the HRG). Our main finding to take away is that (in expectation) there are order  $\log n$  gaps between each of the quantities. This indicates that most vertices in the giant are well-connected to the center of the graph, but a significant proportion are not.

We restricted our study to the giant component of the graph in the regime  $\frac{1}{2} < \alpha < 1$ , although this is arguably the most interesting regime it would still be interesting to determine the aforementioned quantities on the other smaller components or when  $\alpha \notin (\frac{1}{2}, 1)$ . Another problem we leave open is to discover the leading constants hidden behind our asymptotic notation, if the expression for the clustering coefficient of the HRG [31] is anything to go by these constants may have very rich and complex expressions as functions of  $\alpha$  and  $\nu$ . An interesting problem is to determine the order of meeting time, that is, the expected time it takes two (lazy) random walks to occupy the same vertex when started from the worst case start vertices [43]. Finally, the mixing time of a (lazy) random walk on the giant HRG is known up to polylogarithmic factors by [47]. Closing this gap is of great importance, but it may well be quite challenging.

---

## References

- 1 Mohammed Amin Abdullah, Michel Bode, and Nikolaos Fountoulakis. Typical distances in a geometric model for complex networks. *Internet Math.*, page 38, 2017.
- 2 Tasweer Ahmad, Lianwen Jin, LuoJun Lin, and Guozhi Tang. Skeleton-based action recognition using sparse spatio-temporal GCN with edge effective resistance. *Neurocomputing*, 423:389–398, 2021. doi:10.1016/j.neucom.2020.10.096.
- 3 David Aldous and James Allen Fill. Reversible Markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014. URL: <https://www.stat.berkeley.edu/~aldous/RWG/book.html>.
- 4 Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 218–223. IEEE Computer Society, 1979. doi:10.1109/SFCS.1979.34.
- 5 Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.41.
- 6 Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 20–39. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.11.
- 7 Chen Avin and Gunes Ercal. On the cover time and mixing time of random geometric graphs. *Theor. Comput. Sci.*, 380(1-2):2–22, 2007. doi:10.1016/j.tcs.2007.02.065.
- 8 Anna Ben-Hamou, Roberto I. Oliveira, and Yuval Peres. Estimating graph parameters via random walks with restarts. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1702–1714. SIAM, 2018. doi:10.1137/1.9781611975031.111.
- 9 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

- 10 Mindaugas Bloznelis, Jerzy Jaworski, and Katarzyna Rybarczyk. The cover time of a sparse random intersection graph, 2019. doi:10.48550/ARXIV.1910.09639.
- 11 Michel Bode, Nikolaos Fountoulakis, and Tobias Müller. On the largest component of a hyperbolic model of complex networks. *Electron. J. Comb.*, 22(3):P3.24, 2015. doi:10.37236/4958.
- 12 Michel Bode, Nikolaos Fountoulakis, and Tobias Müller. The probability of connectivity in a hyperbolic model of complex networks. *Random Struct. Algorithms*, 49(1):65–94, 2016.
- 13 Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):1–8, 2010.
- 14 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theoretical Computer Science*, 760:35–54, 2019.
- 15 Elisabetta Candellero and Nikolaos Fountoulakis. Bootstrap percolation and the geometry of complex networks. *Stochastic Processes and their Applications*, 126(1):234–264, 2016.
- 16 Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Comput. Complex.*, 6(4):312–340, 1997. doi:10.1007/BF01270385.
- 17 Jordan Chellig, Nikolaos Fountoulakis, and Fiona Skerman. The modularity of random graphs on the hyperbolic plane, 2021. arXiv:2104.01041.
- 18 Paul F. Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282. ACM, 2011. doi:10.1145/1993636.1993674.
- 19 Alessandra Cipriani and Michele Salvi. Scale-free percolation mixing time, 2021. doi:10.48550/ARXIV.2111.05201.
- 20 Colin Cooper and Alan M. Frieze. The cover time of sparse random graphs. *Random Struct. Algorithms*, 30(1-2):1–16, 2007. doi:10.1002/rsa.20151.
- 21 Colin Cooper and Alan M. Frieze. The cover time of the preferential attachment graph. *J. Comb. Theory B*, 97(2):269–290, 2007. doi:10.1016/j.jctb.2006.05.007.
- 22 Colin Cooper and Alan M. Frieze. The cover time of the giant component of a random graph. *Random Struct. Algorithms*, 32(4):401–439, 2008. doi:10.1002/rsa.20201.
- 23 Colin Cooper and Alan M. Frieze. The cover time of random geometric graphs. *Random Struct. Algorithms*, 38(3):324–349, 2011. doi:10.1002/rsa.20320.
- 24 Colin Cooper and Alan M. Frieze. Stationary distribution and cover time of random walks on random digraphs. *J. Comb. Theory B*, 102(2):329–362, 2012. doi:10.1016/j.jctb.2011.11.001.
- 25 Colin Cooper, Alan M. Frieze, and Eyal Lubetzky. Cover time of a random graph with a degree sequence II: Allowing vertices of degree two. *Random Struct. Algorithms*, 45(4):627–674, 2014. doi:10.1002/rsa.20573.
- 26 Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. *Random Struct. Algorithms*, 55(1):104–124, 2019. doi:10.1002/rsa.20826.
- 27 Maria Deijfen, Remco Van der Hofstad, and Gerard Hooghiemstra. Scale-free percolation. *Annales de l’IHP Probabilités et statistiques*, 49(3):817–838, 2013.
- 28 Jian Ding, James R. Lee, and Yuval Peres. Cover times, blanket times, and majorizing measures. *Ann. of Math. (2)*, 175(3):1409–1471, 2012. doi:10.4007/annals.2012.175.3.8.
- 29 Nikolaos Fountoulakis, Dieter Mitsche, Tobias Müller, and Markus Schepers. Hamilton cycles and perfect matchings in the KPKVB model. *Stochastic Processes and their Applications*, 131:340–358, 2021.
- 30 Nikolaos Fountoulakis and Tobias Müller. Law of large numbers for the largest component in a hyperbolic model of complex networks. *Ann. Appl. Probab.*, 28(1):607–650, 2018. doi:10.1214/17-AAP1314.

- 31 Nikolaos Fountoulakis, Pim van der Hoorn, Tobias Müller, and Markus Schepers. Clustering in a hyperbolic model of complex networks. *Electron. J. Probab.*, 26:Paper No. 13, 132, 2021. doi:10.1214/21-ejp583.
- 32 Tobias Friedrich and Anton Krohmer. Cliques in hyperbolic random graphs. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1544–1552. IEEE, 2015.
- 33 Tobias Friedrich and Anton Krohmer. On the diameter of hyperbolic random graphs. *SIAM J. Discret. Math.*, 32(2):1314–1334, 2018. doi:10.1137/17M1123961.
- 34 David Gans. A new model of the hyperbolic plane. *The American Mathematical Monthly*, 73(3):291–295, 1966.
- 35 Christos Gkantsidis, Milena Mihail, and Amin Saberi. Hybrid search schemes for unstructured peer-to-peer networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 1526–1537. IEEE, 2005. doi:10.1109/INFCOM.2005.1498436.
- 36 Peter Gracar, Markus Heydenreich, Christian Mönch, and Peter Mörters. Recurrence versus transience for weight-dependent random connection models, 2021. arXiv:1911.04350.
- 37 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 573–585. Springer, 2012. doi:10.1007/978-3-642-31585-5\_51.
- 38 Brieuc Guinard and Amos Korman. Tight bounds for the cover times of random walks with heterogeneous step lengths. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 28:1–28:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.28.
- 39 Markus Heydenreich, Tim Hulshof, and Joost Jorritsma. Structures in supercritical scale-free percolation. *Ann. Appl. Probab.*, 27(4):2569–2604, 2017. doi:10.1214/16-AAP1270.
- 40 Jeremy G. Hoskins, Cameron Musco, Christopher Musco, and Babis Tsourakakis. Inferring networks from random walk-based node similarities. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3708–3719, 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/2f25f6e326adb93c5787175dda209ab6-Abstract.html>.
- 41 Johan Jonasson. On the cover time for random walks on random graphs. *Comb. Probab. Comput.*, 7(3):265–279, 1998. URL: <http://journals.cambridge.org/action/displayAbstract?aid=46637>.
- 42 Jeff Kahn, Jeong Han Kim, László Lovász, and Van H. Vu. The cover time, the blanket time, and the Matthews bound. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000*, pages 467–475. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892134.
- 43 Varun Kanade, Frederik Mallmann-Trenn, and Thomas Sauerwald. On coalescence time in graphs: When is coalescing as fast as meeting? In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 956–965. SIAM, 2019. doi:10.1137/1.9781611975482.59.
- 44 David Kempe, Jon M. Kleinberg, and Alan J. Demers. Spatial gossip and resource location protocols. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 163–172. ACM, 2001. doi:10.1145/380752.380796.
- 45 Marcos Kiwi, Amitai Linker, and Dieter Mitsche. Tail bounds for detection times in mobile hyperbolic graphs, 2022. doi:10.48550/ARXIV.2203.00209.

- 46 Marcos Kiwi and Dieter Mitsche. A bound for the diameter of random hyperbolic graphs. In *Proceedings of the Twelfth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2015*, pages 26–39. SIAM, 2015. doi:10.1137/1.9781611973761.3.
- 47 Marcos Kiwi and Dieter Mitsche. Spectral gap of random hyperbolic graphs and related parameters. *Ann. Appl. Probab.*, 28(2):941–989, 2018.
- 48 Marcos Kiwi and Dieter Mitsche. On the second largest component of random hyperbolic graphs. *SIAM J. Discrete Math.*, 33(4):2200–2217, 2019. doi:10.1137/18M121201X.
- 49 Marcos Kiwi, Markus Schepers, and John Sylvester. Cover and hitting times of hyperbolic random graphs, 2022. doi:10.48550/ARXIV.2207.06956.
- 50 Christoph Koch and Johannes Lengler. Bootstrap percolation on geometric inhomogeneous random graphs. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11–15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 147:1–147:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.147.
- 51 Júlia Komjáthy and Bas Lodewijks. Explosion in weighted hyperbolic random graphs and geometric inhomogeneous random graphs. *Stochastic Processes and their Applications*, 130(3):1309–1367, 2020. doi:10.1016/j.spa.2019.04.014.
- 52 D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82(3):036106, 18, 2010. doi:10.1103/PhysRevE.82.036106.
- 53 Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A  $n^{1/2+o(1)}$ -query one-sided tester for minor closed properties on bounded degree graphs. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7–9, 2018*, pages 509–520. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00055.
- 54 Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: A  $\text{poly}(d/\epsilon)$ -query tester for minor-closed properties of bounded degree graphs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23–26, 2019*, pages 559–567. ACM, 2019. doi:10.1145/3313276.3316330.
- 55 G. Last and M. Penrose. *Lectures on the Poisson Process*. IMS Textbooks. Cambridge University Press, 2018.
- 56 David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, Providence, RI, 2009.
- 57 Huan Li and Zhongzhi Zhang. Kirchhoff index as a measure of edge centrality in weighted networks: Nearly linear time algorithms. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10, 2018*, pages 2377–2396. SIAM, 2018. doi:10.1137/1.9781611975031.153.
- 58 Amitai Linker, Dieter Mitsche, Bruno Schapira, and Daniel Valesin. The contact process on random hyperbolic graphs: metastability and critical exponents. *Ann. Probab.*, 49(3):1480–1514, 2021. doi:10.1214/20-aop1489.
- 59 László Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 353–397. János Bolyai Math. Soc., Budapest, 1996.
- 60 Russell Lyons and Yuval Peres. *Probability on trees and networks*, volume 42 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, New York, 2016. doi:10.1017/9781316672815.
- 61 Aleksander Madry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4–6, 2015*, pages 2019–2036. SIAM, 2015. doi:10.1137/1.9781611973730.134.

- 62 Francesco Manzo, Matteo Quattropiani, and Elisabetta Scoppola. A probabilistic proof of Cooper & Frieze’s “First visit time lemma”. *ALEA Lat. Am. J. Probab. Math. Stat.*, 18(2):1739–1758, 2021. doi:10.30757/alea.v18-64.
- 63 Christine Marshall, Colm O’Riordan, and James Cruickshank. Targeting influential nodes for recovery in bootstrap percolation on hyperbolic networks. In *European Network Intelligence Conference*, pages 3–16. Springer, 2017.
- 64 Tobias Müller and Merlijn Staps. The diameter of KPKVB random graphs. *Adv. Appl. Probab.*, 51(2):358–377, 2019. doi:10.1017/apr.2019.23.
- 65 Takashi Owada and D Yogeshwaran. Sub-tree counts on hyperbolic random geometric graphs. *To appear in Advances in Applied Probability*, 2022+.
- 66 José Luis Palacios. Closed-form formulas for Kirchhoff index. *International J. of Quantum Chemistry*, 81(2):135–140, 2001.
- 67 Stacy Patterson and Bassam Bamieh. Consensus and coherence in fractal networks. *IEEE Trans. Control. Netw. Syst.*, 1(4):338–348, 2014. doi:10.1109/TCNS.2014.2357552.
- 68 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 341–350. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.86.
- 69 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011. doi:10.1137/080734029.
- 70 John Sylvester. Random walk hitting times and effective resistance in sparsely connected Erdős-Rényi random graphs. *J. Graph Theory*, 96(1):44–84, 2021. doi:10.1002/jgt.22551.
- 71 Prasad Tetali. Random walks and the effective resistance of networks. *J. Theor. Probab.*, 4(1):101–109, 1991. doi:10.1007/BF01046996.
- 72 Ulrike von Luxburg, Agnes Radl, and Matthias Hein. Hitting and commute times in large random neighborhood graphs. *J. Mach. Learn. Res.*, 15(52):1751–1798, 2014.
- 73 Felix Ming Fai Wong, Zhenming Liu, and Mung Chiang. On the efficiency of social recommender networks. *IEEE/ACM Trans. Netw.*, 24(4):2512–2524, 2016. doi:10.1109/TNET.2015.2475616.




# Adaptive Sketches for Robust Regression with Importance Sampling

Sepideh Mahabadi ✉

Microsoft Research, Redmond, WA, USA

David P. Woodruff ✉

Carnegie Mellon University, Pittsburgh, PA, USA

Samson Zhou ✉ 

Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

We introduce data structures for solving robust regression through stochastic gradient descent (SGD) by sampling gradients with probability proportional to their norm, i.e., importance sampling. Although SGD is widely used for large scale machine learning, it is well-known for possibly experiencing slow convergence rates due to the high variance from uniform sampling. On the other hand, importance sampling can significantly decrease the variance but is usually difficult to implement because computing the sampling probabilities requires additional passes over the data, in which case standard gradient descent (GD) could be used instead. In this paper, we introduce an algorithm that approximately samples  $T$  gradients of dimension  $d$  from nearly the optimal importance sampling distribution for a robust regression problem over  $n$  rows. Thus our algorithm effectively runs  $T$  steps of SGD with importance sampling while using sublinear space and just making a single pass over the data. Our techniques also extend to performing importance sampling for second-order optimization.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Streaming algorithms, stochastic optimization, importance sampling

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.31

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/pdf/2207.07822.pdf>

**Funding** David P. Woodruff and Samson Zhou were supported by a Simons Investigator Award and by the National Science Foundation under Grant No. CCF-1815840.

## 1 Introduction

Given a matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$  with rows  $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^d$  and a measurement/label vector  $\mathbf{b} \in \mathbb{R}^n$ , we consider the standard regression problem

$$\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}) := \sum_{i=1}^n M(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i),$$

where  $M : \mathbb{R} \rightarrow \mathbb{R}^{\geq 0}$  is a function, called a measure function, that satisfies  $M(x) = M(-x)$  and is non-decreasing in  $|x|$ . An  $M$ -estimator is a solution to this minimization problem and for appropriate choices of  $M$ , can combine the low variance of  $L_2$  regression with the robustness of  $L_1$  regression against outliers.

The Huber norm, for example, is defined using the measure function  $H(x) = \frac{x^2}{2\tau}$  for  $|x| \leq \tau$  and  $H(x) = |x| - \frac{\tau}{2}$  for  $|x| > \tau$ , where  $\tau$  is a threshold that governs the interpolation between  $L_2$  loss for small  $|x|$  and  $L_1$  loss for large  $|x|$ . Indeed, it can often be more reasonable to have robust treatment of large residuals due to outliers or errors and Gaussian treatment of



© Sepideh Mahabadi, David P. Woodruff, and Samson Zhou;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 31; pp. 31:1–31:21



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



small residuals [12]. Thus the Huber norm is especially popular and “recommended for almost all situations” [39], because it is the “most robust” [13] due to “the useful computational and statistical properties implied by the convexity and smoothness” [9] of its measure function, which is differentiable at all points.

Since the measure function  $H$  for the Huber norm and more generally, the measure function  $M$  for many common measure functions is convex, we can consider the standard convex *finite-sum form* optimization problem  $\min_{\mathbf{x} \in \mathbb{R}^d} F(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x})$ , where  $f_1, \dots, f_n : \mathbb{R}^d \rightarrow \mathbb{R}$  is a sequence of convex functions that commonly represent loss functions. Whereas gradient descent (GD) performs the update rule  $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla F(\mathbf{x}_t)$  on the iterate  $\mathbf{x}_t$  at iterations  $t = 1, 2, \dots, T$ , stochastic gradient descent (SGD) [32, 27, 26] picks  $i_t \in [n]$  in iteration  $t$  with probability  $p_{i_t}$  and performs the update rule  $\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta_t}{np_{i_t}} \nabla f_{i_t}(\mathbf{x}_t)$ , where  $\nabla f_{i_t}$  is the gradient (or a subgradient) of  $f_{i_t}$  and  $\eta_t$  is some predetermined learning rate. Effectively, training example  $i_t$  is sampled with probability  $p_{i_t}$  and the model parameters are updated using the selected example. The SGD update rule only requires the computation of a single gradient at each iteration and provides an unbiased estimator to the full gradient, compared to GD, which evaluates  $n$  individual gradients in each iteration and is prohibitively expensive for large  $n$ . However, since SGD is often performed with uniform sampling, so that the probability  $p_{i,t}$ <sup>1</sup> of choosing index  $i \in [n]$  at iteration  $t$  is  $p_{i,t} = \frac{1}{n}$  at all times, the variance introduced by the randomness of sampling a specific vector function can be a bottleneck for the convergence rate of this iterative process. Thus, the subject of variance reduction beyond uniform sampling has been well-studied in recent years [33, 18, 11, 31, 40, 10, 25, 35, 19, 21, 34, 30].

A common technique to reduce variance is importance sampling, where the probabilities  $p_{i,t}$  are chosen so that vector functions with larger gradients are more likely to be sampled. One such setting of importance sampling is to set the probability of sampling a gradient with probability proportional to its  $L_2$  norm, so that

$$p_{i,t} = \frac{\|\nabla f_i(\mathbf{x}_t)\|_2}{\sum_{j \in [n]} \|\nabla f_j(\mathbf{x}_t)\|_2}.$$

Under these sampling probabilities, importance sampling gives variance

$$\sigma_{opt,t}^2 = \frac{1}{n^2} \left( \left( \sum_{i=1}^n \|\nabla f_i(\mathbf{x}_t)\|_2 \right)^2 - n^2 \cdot \|\nabla F(\mathbf{x}_t)\|_2^2 \right),$$

where we define the variance of a random vector  $\mathbf{v}$  to be  $\text{Var}(\mathbf{v}) := \mathbb{E}[\|\mathbf{v}\|_2^2] - \|\mathbb{E}[\mathbf{v}]\|_2^2$ , and we define  $\sigma_{opt,t}^2$  to be the variance of the random vector  $\mathbf{v}$  produced at time  $t$  by importance sampling.

By comparison, the probabilities for uniform sampling  $p_{i,t} = \frac{1}{n}$  imply  $\sigma_t^2 = \text{Var}\left(\frac{1}{np_{i,t,t}}\right)$  and thus the variance  $\sigma_{uni,t}^2$  for uniform sampling satisfies

$$\sigma_{uni,t}^2 = \frac{1}{n^2} \left( n \sum_{i=1}^n \|\nabla f_i(\mathbf{x}_t)\|_2^2 - n^2 \cdot \|\nabla F(\mathbf{x}_t)\|_2^2 \right).$$

By the root mean square-arithmetic mean inequality, the variance of importance sampling is always at most the variance of uniform sampling, and can be significantly less. Hence  $\sigma_{opt,t}^2 \leq \sigma_{uni,t}^2$ , so that the variance at each step is reduced, possibly substantially, by performing importance sampling instead of uniform sampling.

<sup>1</sup> In contrast to  $p_{i,t}$ , the term  $p_{i_t}$  denotes the probability associated with the specific index  $i_t$  chosen at time  $t$ .

To see examples where uniform sampling an index performs significantly worse than importance sampling, consider  $\nabla f_i(\mathbf{x}) = \langle \mathbf{a}_i, \mathbf{x} \rangle \cdot \mathbf{a}_i$ . Then for  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n$ :

► **Example 1.** When the non-zero entries of the input  $\mathbf{A}$  are concentrated in a small number of vectors  $\mathbf{a}_i$ , uniform sampling will frequently sample gradients that are small and make little progress, whereas importance sampling will rarely do so. In an extreme case, the input  $\mathbf{A}$  can contain exactly one non-zero vector  $\mathbf{a}_i$  and importance sampling will always output the full gradient, whereas uniform sampling will only find the non-zero row with probability  $\frac{1}{n}$ , so that  $\sigma_{uni,t}^2 = n \cdot \sigma_{opt,t}^2$ .

► **Example 2.** It may be that all rows of  $\mathbf{A}$  have large magnitude, but  $\mathbf{x}$  is nearly orthogonal to most of the rows of  $\mathbf{A}$ , but is well-aligned with row  $\mathbf{a}_r$ . Then  $\langle \mathbf{a}_i, \mathbf{x} \rangle \cdot \mathbf{a}_i$  is small in magnitude for most  $i$ , but  $\langle \mathbf{a}_r, \mathbf{x} \rangle \cdot \mathbf{a}_r$  is large so uniform sampling will often output small gradients while importance sampling will output  $\langle \mathbf{a}_r, \mathbf{x} \rangle \cdot \mathbf{a}_r$  with high probability, so that it can be that  $\sigma_{uni,t}^2 = \Omega(n) \cdot \sigma_{opt,t}^2$ .

► **Example 3.** More generally for a parameter  $\nu \in [0, 1]$ , if a  $\nu$ -fraction of the  $n$  gradient lengths are bounded by  $\mathcal{O}(n)$  while the other  $1 - \nu$  fraction of the  $n$  gradient lengths are bounded by  $\text{poly}(d) \ll n$ , then the variance for uniform sampling satisfies  $\sigma_{uni,t}^2 = \mathcal{O}(\nu n^2) + \text{poly}(d)$  while the variance for importance sampling satisfies  $\mathcal{O}(\nu^2 n^2) + \text{poly}(d)$ .

In fact, it follows from the Cauchy-Schwarz inequality that the importance sampling probability distribution is the *optimal* distribution for variance reduction.

However, computing the probability distribution for importance sampling requires computing the gradients in each round, which creates a “chicken and egg” problem because computing the gradients is too expensive in the first place, or else it is feasible to just run gradient descent. Unfortunately, computing the sampling probabilities in each iteration often requires additional passes over the data, e.g., to compute the gradients in each step, which is generally prohibitively expensive. This problem often prevents importance sampling from being widely deployed.

In this paper, we overcome this problem by introducing efficient sketches for a wide range of  $M$ -estimators that can enable importance sampling *without* additional passes over the data. Using our sketches for various measure functions, we give a time-efficient algorithm that *provably* approximates the optimal importance sampling distribution within a constant factor. Thus we can surprisingly simulate  $T$  steps of SGD with (nearly) the optimal sampling distribution, while only using a single pass over the data, which avoids the aforementioned problem.

► **Theorem 4.** *Given an input matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$  whose rows arrive sequentially in a data stream along with the corresponding labels of a measurement vector  $\mathbf{b} \in \mathbb{R}^d$ , and a measure function  $M$  whose derivative is a continuous union of piecewise constant or linear functions, there exists an algorithm that performs  $T$  steps of SGD with variance within a constant factor of the optimal sampling distribution. The algorithm uses  $\tilde{\mathcal{O}}(nd^2 + Td^2)$  pre-processing time and  $Td^2 \text{ polylog}(Tnd)$  words of space.*

For  $T$  iterations, both GD and optimal importance sampling SGD require  $T$  passes over the data, while our algorithm only requires a single pass over the data and uses sublinear space for  $nd \gg Td^2$ . We remark that although the number  $T$  of iterations for SGD may be large, a major advantage of GD and SGD with importance sampling is a significantly smaller number of iterations than SGD with uniform sampling, e.g., as in Example 1 and Example 2, so we should expect  $n \gg T$ . In particular from known results about the convergence of SGD,

e.g., see Theorem 8, if the diameter of the search space and the gradient lengths  $\|\nabla f_i(x_t)\|_2$  are both bounded by  $\text{poly}(d)$ , then we should expect  $T \propto \text{poly}(d)$  even for uniform sampling. More generally, if  $\nu n$  of the gradients have lengths  $\Theta(n)$ , while the remaining gradients have lengths  $\text{poly}(d) \ll n$ , then Example 3 and Theorem 8 show that the number of steps necessary for convergence for uniform sampling satisfies  $T \propto \mathcal{O}(\nu^2 n^4) + \text{poly}(d)$ , while the number of steps necessary for convergence for importance sampling satisfies  $T \propto \mathcal{O}(\nu^4 n^4)$ . Thus for  $\nu n = \mathcal{O}(n^C)$  for  $C < 1$ , i.e., a sublinear number of gradients have lengths that exceed the input size, we have  $T = \mathcal{O}(n^{4C})$  and hence for  $C < \frac{1}{4}$ , we have roughly  $T = o(n)$  steps are necessary for convergence for SGD with importance sampling.

Finally, we show in the full version of the paper that our techniques can also be generalized to perform importance sampling for second-order optimization.

## 1.1 Our Techniques

In addition to our main conceptual contribution that optimal convergence rate of importance sampling for SGD can surprisingly be achieved (up to constant factors) without the “chicken and egg” problem of separately computing the sampling probabilities, we present a number of technical contributions that may be of independent interest. Our first observation is that if we were only running a single step of importance sampling for SGD, then we just want a subroutine that outputs a gradient  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  with probability proportional to its norm  $\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$ .

**G-sampler.** In particular, we need an algorithm that reads a matrix  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$  and a vector  $\mathbf{x} \in \mathbb{R}^d$  given *after processing* the matrix  $\mathbf{A}$ , and outputs (a rough approximation to) a gradient  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  with probability roughly

$$\frac{\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2}{\sum_{j=1}^n \|G(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j, \mathbf{a}_j)\|_2}.$$

We call such an algorithm a *G-sampler* and introduce such a single-pass, memory-efficient sampler with the following guarantees:

► **Theorem 5.** *Given an  $(\alpha, \varepsilon)$ -smooth gradient  $G$ , there exists an algorithm SAMPLER that outputs a noisy vector  $\mathbf{v}$  such that  $\|\mathbf{v} - \mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)\|_2 \leq \alpha \|\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)\|_2$  and  $\mathbb{E}[\mathbf{v}] = \mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  is  $(1 \pm \mathcal{O}(\varepsilon)) \frac{\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2}{\sum_{j \in [n]} \|G(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j, \mathbf{a}_j)\|_2} + \frac{1}{\text{poly}(n)}$ . The algorithm uses  $d^2 \text{poly}(\log(nT), \frac{1}{\alpha})$  update time per arriving row and  $Td^2 \text{poly}(\log(nT), \frac{1}{\alpha})$  total bits of space.*

We say a gradient  $G$  is  $(\alpha, \varepsilon)$ -smooth if a vector  $\mathbf{u}$  that satisfies  $\|\mathbf{u} - \mathbf{v}\|_2 \leq \alpha \|\mathbf{v}\|_2$  implies that  $(1 - \varepsilon)\|G(\mathbf{v})\|_2 \leq \|G(\mathbf{u})\|_2 \leq (1 + \varepsilon)\|G(\mathbf{v})\|_2$ . In particular, the measure functions discussed in Section 1.2 have gradients that are  $(\mathcal{O}(\varepsilon), \varepsilon)$ -smooth. For example, the subgradient of the Huber estimator is  $\mathbf{a}_i \cdot \text{sgn}(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  for  $|\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i| > \tau$ , which may change sign when  $\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i$  is close to zero, but its norm will remain the same. Moreover, the form  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  necessitates that the gradient can be computed strictly from the two quantities  $\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i$  and  $\mathbf{a}_i$ . Thus Theorem 5 implies that our algorithm can also compute a noisy vector  $\mathbf{v}'$  such that  $\|\mathbf{v}' - G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2 \leq \varepsilon \|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$ .

Observe that an instance of SAMPLER in Theorem 5 can be used to simulate a single step of SGD with importance sampling and thus  $T$  independent instances of SAMPLER provide an oracle for  $T$  steps of SGD with importance sampling. However, this naïve implementation

does not suffice for our purposes because the overall runtime would be  $\tilde{O}(Tnd^2)$  so it would be more efficient to just run  $T$  iterations of GD. Nevertheless, our  $G$ -sampler is a crucial subroutine towards our final algorithm and we briefly describe it here.

An alternative definition of  $G$ -sampler is given in [16]. In their setting, the goal is to sample a coordinate  $i \in [n]$  of a frequency vector  $f$  with probability proportional to  $G(f_i)$ , where  $G$  in their notation is a measure function rather than a gradient. However, because the  $G$ -sampler of [16] is not a linear sketch, their approach cannot be easily generalized to our setting where the sampling probability of each row  $\mathbf{a}_i$  is a function of  $\langle \mathbf{a}_i, \mathbf{x} \rangle$ , but the vector  $\mathbf{x}$  arrives after the stream is already processed.

Furthermore, because the loss function  $f$  may not be scale-invariant, then we should also not expect its gradient to be scale invariant at any location  $\mathbf{x} \in \mathbb{R}^d$ , i.e.,  $\nabla f(C\mathbf{x}) \neq C^p \cdot \nabla f(\mathbf{x})$  for any constants  $p, C > 0$ . Hence, our subroutine SAMPLER cannot use the standard  $L_p$  sampler framework used in [20, 2, 15, 22], which generally rescales each row of  $\mathbf{a}_i$  by the inverse of a uniform or exponential random variable. A somewhat less common design for  $L_p$  samplers is a level set and subsampling approach [24, 17], due to their suboptimal dependencies on the accuracy parameter  $\varepsilon$ . Fortunately, because we require  $\varepsilon = \mathcal{O}(1)$  to achieve a constant factor approximation, we can use the level set and subsampling paradigm as a starting point for our algorithm. Because the algorithms of [24, 17] only sample entries of a vector implicitly defined from a data stream, our  $G$ -sampler construction must (1) sample rows of a matrix implicitly defined from a data stream and (2) permit updates to the sampling probabilities implicitly defined through multiplication of each row  $\mathbf{a}_i$  with a vector  $\mathbf{x}$  that only arrives after the stream is processed.

**$G$ -sampler through level sets and subsampling.** To illustrate our method and simplify presentation here, we consider  $L_2$  regression with gradient  $\mathbf{A}_i \mathbf{x} := \langle \mathbf{a}_i, \mathbf{x} \rangle \cdot \mathbf{a}_i$ , by folding in the measurement vector  $\mathbf{b}$  into a column of  $\mathbf{A}$  – our full algorithm in Section 2 handles both sampling distributions defined with respect to the norm of a general gradient  $G$  in the form of Theorem 5, as well as an independent measurement vector  $\mathbf{b}$ .

We first partition the rows of  $\mathbf{A}$  into separate geometrically growing classes based on their  $L_2$  norms, so that for instance, class  $C_k$  contains the rows  $\mathbf{a}_i$  of  $\mathbf{A}$  such that  $2^k \leq \|\mathbf{a}_i\|_2 < 2^{k+1}$ . We build a separate data structure for each class  $C_k$ , which resembles the framework for  $L_p$  norm estimation [14]. We would like to use the approximate contributions of the level sets  $\Gamma_1, \dots, \Gamma_K$ , with  $K = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$ , toward the total mass  $F_2(S) = \sum_{i=1}^n \|\mathbf{A}_i \mathbf{x}\|_2$ , where a level set  $\Gamma_j$  is informally the set of rows  $\mathbf{a}_i$  with  $\|\mathbf{A}_i \mathbf{x}\|_2 \in \left[\frac{F_2(S)}{(1+\alpha)^{j-1}}, \frac{F_2(S)}{(1+\alpha)^j}\right]$  and the contribution of a level set  $\Gamma_j$  is  $\sum_{i \in \Gamma_j} \|\mathbf{A}_i \mathbf{x}\|_2$ . Then we could first sample a level set  $\Gamma_j$  from a class  $C_k$  and then uniformly select a row  $\mathbf{a}_i$  among those in  $\Gamma_j$ . Indeed, we can run a generalized version of the  $L_2$  heavy-hitter algorithm COUNTSKETCH [7] on the stream  $S$  to identify the level set  $\Gamma_1$ , since its rows will be heavy with respect to  $F_2(S)$ . However, the rows of the level sets  $\Gamma_j$  for large  $j$  may not be detected by COUNTSKETCH. Thus, we create  $L = \mathcal{O}(\log n)$  substreams  $S_1, \dots, S_L$ , so that substream  $S_\ell$  samples each row of  $\mathbf{A}$  with probability  $2^{-\ell+1}$ , and run an instance of COUNTSKETCH on each substream  $S_\ell$  to detect the rows of each level set and thus estimate the contribution of each level set.

**Sampling from level sets with small contribution.** However, there is still an issue – some level sets have contribution that is too small to well-approximate with small variance. For example, if there is a single row with contribution  $\frac{F_2(S)}{(1+\alpha)^j}$ , then it might not survive the subsampling at a level  $S_\ell$  that is used to detect it, in which case it will never be sampled. Alternatively, if it is sampled, it will be rescaled by a large amount, so that its level set will

be sampled with abnormally large probability. Instead of handling this large variance, we instead add a number of dummy rows to each level set, to ensure that their contributions are all “significant” and thus be well-approximated.

Now we have “good” approximations to the contributions of each level set within a class, so we can first select a level set with probability proportional to the approximate contributions of each level set and then uniformly sample a row from the level set. Of course, we may uniformly sample a dummy row, in which case we say the algorithm fails to acquire a sample. We show that the contribution added by the dummy rows is a constant fraction, so this only happens with a constant probability. Thus with  $\mathcal{O}(\log \frac{1}{\delta})$  constant number of independent samples, we can boost the probability of successfully acquiring a sample to  $1 - \delta$  for any  $\delta \in (0, 1]$ . We then set  $\delta = \frac{1}{\text{poly}(n, T, d)}$ .

**Unbiased samples.** Unfortunately, COUNTSKETCH using  $\mathcal{O}(\frac{1}{\alpha^2})$  buckets only guarantees additive  $\alpha L_2(S)$  error to a particular row with constant probability. To achieve the standard “for-all” guarantee across all  $n$  rows, an estimate for each row  $\mathbf{a}_i$  is then output by taking the row with the median length across  $\mathcal{O}(\log n)$  independent instances. However, the median row is no longer unbiased, which could potentially affect the convergence guarantees of SGD. Instead, we use  $d$  separate instances of COUNTSKETCH, so that each instance handles a separate coordinate of the vector. Thus if the goal is to output a noisy estimate to  $\mathbf{a}_i$ , we have a separate COUNTSKETCH report each coordinate  $(\mathbf{a}_i)_j$ , where  $j \in [d]$ . It can be shown that the median of each estimated coordinate is an unbiased estimate to the true value  $(\mathbf{a}_i)_j$  of the coordinate because the probability mass function is symmetric about the true value for each coordinate. Moreover, the error to a single coordinate  $(\mathbf{a}_i)_j$  may be large relative to the value of the coordinate in the case that  $(\mathbf{a}_i)_j$  is not heavy with respect to  $\{(\mathbf{a}_i)_j\}_{i \in [n]}$ . However, we show that the “overall” error to all coordinates of  $\mathbf{a}_i$  is small relative to  $\|\mathbf{a}_i\|_2$ , due to  $\mathbf{a}_i$  being a “heavy” row at the appropriate subsampling level.

**Stochastic gradient descent with importance sampling.** The main problem with the proposed  $G$ -sampler is that it requires reading the entire matrix  $\mathbf{A}$  but it cannot be repeatedly used without incurring dependency issues. In particular, if a sampler at the first iteration of SGD outputs a gradient  $\mathbf{A}_{i_1} \mathbf{x}_1$  that is used to construct  $\mathbf{x}_2$ , then  $\mathbf{x}_2$  is not independent of the sampler and thus the same sampler should not be used to sample  $\mathbf{A}_{i_2} \mathbf{x}_2$ . This suggests that if we want to perform  $T$  steps of SGD with importance sampling, then we would require  $T$  separate data structures, which would require  $Tnd$  time to construct for dense matrices, but then we might as well just perform full gradient descent!

Instead in Section 3, we partition the matrix  $\mathbf{A}$  among multiple buckets and create a sampler for each bucket. Now as long as each bucket should have been sampled a single time, then we will have a fresh sampler with independent randomness for each time a new bucket is sampled. If we perform  $T$  steps of SGD with importance sampling, then roughly  $T$  buckets should suffice, but we cannot guarantee that each bucket is sampled a single time. For example, if only a single  $\mathbf{A}_i$  is non-zero, then whichever bucket  $\mathbf{A}_i$  is assigned to will be sampled every single time.

Now the challenge is identifying the submatrices  $\mathbf{A}_i = \mathbf{a}_i^\top \mathbf{a}_i$  that may be sampled multiple times, since we do not know the values of the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_T$  a priori. Fortunately, we know that  $\|\mathbf{A}_i \mathbf{x}_t\|_2$  can only be large if  $\mathbf{a}_i$  has high sensitivity, where we define the sensitivity for a row  $\mathbf{a}_i$  in  $\mathbf{A}$  to be the quantity  $\max_{\mathbf{x} \in \mathbb{R}^d} \frac{\|\mathbf{a}_i^\top (\langle \mathbf{a}_i, \mathbf{x} \rangle)\|_2}{\sum_{j=1}^n \|\mathbf{a}_j^\top (\langle \mathbf{a}_j, \mathbf{x} \rangle)\|_2}$ . Thus if a block is sampled multiple times, then one of its rows must have large sensitivity.

Hence, we would like to identify the buckets that contain any row with sensitivity at least  $\frac{1}{T}$  and create  $T$  independent samplers for those buckets so that even if the same bucket is sampled all  $T$  times, there will be a fresh sampler available. Crucially, the process of building separate buckets for the rows with the large sensitivities can be identified in just a single pass over the data.

We remark that since each row has sensitivity  $\max_{\mathbf{x} \in \mathbb{R}^d} \frac{\|\mathbf{a}_i^\top \langle \mathbf{a}_i, \mathbf{x} \rangle\|_2}{\sum_{j=1}^n \|\mathbf{a}_j^\top \langle \mathbf{a}_j, \mathbf{x} \rangle\|_2}$ , then it can be shown that the sum of the sensitivities is  $\mathcal{O}(d \log n)$  by partitioning the rows into  $\mathcal{O}(\log n)$  classes  $C_1, C_2, \dots$  of exponentially increasing norm, so that  $\mathbf{a}_i \in C_\ell$  if  $2^\ell \leq \|\mathbf{a}_i\|_2 < 2^{\ell+1}$ . We then note that the sensitivity of each row  $\mathbf{a}_i \in C_\ell$  is upper bounded by  $\max_{\mathbf{x} \in \mathbb{R}^d} \frac{|\langle \mathbf{a}_i, \mathbf{x} \rangle|}{\sum_{\mathbf{a}_j \in C_\ell} |\langle \mathbf{a}_j, \mathbf{x} \rangle|}$ . However, this latter quantity is an  $L_1$  sensitivity, whose sum is known to be bounded by  $\mathcal{O}(d)$ , e.g., [8]. Thus the sum of the sensitivities in each class is at most  $\mathcal{O}(d)$  and so for a matrix  $\mathbf{A}$  whose entries are polynomially bounded by  $n$ , the sum of the sensitivities is at most  $\mathcal{O}(d \log n)$ .

Unfortunately, since the sensitivities sum to  $\mathcal{O}(d \log n)$ , there can be up to  $Td$  rows with sensitivity at least  $\frac{1}{T}$ , so creating  $T$  independent samplers corresponding to each of these rows would yield  $\Omega(T^2 d)$  samplers, which is a prohibitive amount of space. Instead, we simply remove the rows with large sensitivities from the buckets and store them explicitly. We then show this approach still avoids any sampler from being used multiple times across the  $T$  iterations while also enabling the data structure to just use  $\tilde{\mathcal{O}}(Td)$  samplers. Now since we can explicitly consider the rows with sensitivities roughly at least  $\frac{1}{T}$ , then we can use  $\Theta(T)$  buckets in total to ensure that the remaining non-zero entries of  $\mathbf{A}$  are partitioned evenly across buckets that will only require  $\Theta(\log(Td))$  independent samplers.

## 1.2 Applications

In this section, we discuss applications of our result to commonly used loss functions, such as  $L_p$  loss or various  $M$ -estimators, e.g., [9, 8, 37, 29].

**$L_1$  and  $L_2$  regression.** The  $L_p$  regression loss function is defined using  $f_i(\mathbf{x}) = |\mathbf{a}_i^\top \mathbf{x} - b_i|^p$ . The case  $p = 2$  corresponds to the standard least squares regression problem, while  $p = 1$  corresponds to least absolute deviation regression, which is more robust to outliers than least squares, but also less stable and with possibly multiple solutions. For  $p = 1$ , the subgradient is  $\mathbf{a}_i \cdot \text{sgn}(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  while for  $p = 2$ , the subgradient is  $2\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$ .

**Huber estimator.** As previously discussed, Huber loss [13] is commonly used, e.g., [39, 9], to achieve Gaussian properties for small residuals [12] and robust properties for large residuals due to outliers or errors. The Huber estimator is also within a constant factor of other  $M$ -estimators that utilize the advantage of the  $L_1$  loss function to minimize the impact of large errors/outliers and that of the  $L_2$  loss function to be convex, such as the  $L_1$ - $L_2$  estimator and the Fair estimator [4]. Given a threshold  $\tau > 0$ , the Huber loss  $H$  is defined by  $H(x) = \frac{x^2}{2\tau}$  for  $|x| \leq \tau$  and  $H(x) = |x| - \frac{\tau}{2}$  for  $|x| > \tau$ . Thus the subgradient for  $H$  is  $\frac{\mathbf{a}_i}{\tau}(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  for  $|\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i| \leq \tau$  and  $\mathbf{a}_i \cdot \text{sgn}(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  for  $|\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i| > \tau$ .

**Ridge regression.** It is often desirable for a solution  $\mathbf{x}$  to be sparse. The natural approach to encourage sparse solutions is to add a regularization  $\lambda \|\mathbf{x}\|_0$  term to the loss function, for some parameter  $\lambda > 0$ . However, since  $\|\mathbf{x}\|_0$  is not convex, ridge regression is often used as a convex relaxation that encourage sparse solutions. The ridge regression loss function satisfies

$f_i(\mathbf{x}) = (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 + \lambda \|\mathbf{x}\|_2^2$  for each  $i \in [n]$ , so that  $\lambda$  regularizes the penalty term associated with the squared magnitude of  $\mathbf{x}$ . Higher values of  $\lambda$  push the optimal solution towards zero, which leads to lower variance, as a particular coordinate has a smaller effect on the prediction. The gradient for the ridge regression loss function satisfies  $\nabla f_i(\mathbf{x}) = 2\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i) + 2\lambda \mathbf{x}$ .

**Lasso.** Another approach that encourages sparsity is using the  $L_1$  regularization instead of the  $L_2$  regularization. Least absolute shrinkage and selection operator (Lasso) regression uses the loss function  $f_i(\mathbf{x}) = (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 + \lambda \|\mathbf{x}\|_1$ . Whereas the penalty term associated for ridge regression will drive down the Euclidean norm of  $\mathbf{x}$  for larger  $\lambda$ , solutions with large  $L_1$  norm are still possible if the mass of  $\mathbf{x}$  is spread across a large number of coordinates. By contrast, the penalty term associated for Lasso drives down the total magnitude of the coordinates of  $\mathbf{x}$ . Thus, in this sense, Lasso tends to drive coordinates to zero and encourages sparsity, which does not usually happen for ridge regression. The subgradient for the Lasso regression loss function satisfies  $\nabla f_i(\mathbf{x}) = 2\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i) + 2\lambda \text{sgn}(\mathbf{x})$ , where we abuse notation by using  $\text{sgn}(\mathbf{x})$  to denote the coordinate-wise sign of the entries of  $\mathbf{x}$ .

**Group lasso.** [38] proposed Group Lasso as a generalization to Lasso. Suppose the weights in  $\mathbf{x}$  can be grouped into  $m$  groups:  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$ . We group the columns of  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n$  so that  $\mathbf{A}^{(i)}$  is the set of columns that corresponds to the weights in  $\mathbf{x}^{(i)}$ . The Group Lasso function is defined as  $f_i(\mathbf{x}) = (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 + \lambda \sum_{j=1}^m \sqrt{G_j} \|\mathbf{x}^{(j)}\|_2$ , where  $G_j$  represents the number of weights in  $\mathbf{x}^{(j)}$ . Note that Group Lasso becomes Lasso for  $m = n$ .

### 1.3 Preliminaries

For an integer  $n > 0$ , we use  $[n]$  to denote the set  $\{1, 2, \dots, n\}$ . We use boldfaced font for variables that represent either vectors or matrices and plain font to denote variables that represent scalars. We use the notation  $\tilde{O}(\cdot)$  to suppress polylog factors, so that  $f(T, n, d) = \tilde{O}(g(T, n, d))$  implies that  $f(T, n, d) \leq g(T, n, d) \text{polylog}(Tnd)$ . Let  $\mathbf{A} \in \mathbb{R}^{n \times d}$  and  $\mathbf{B} \in \mathbb{R}^{m \times d}$ . We use  $\circ$  to denote vertical concatenation, so that  $\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$ , and  $\otimes$  to denote outer product, so that the  $(i, j)$ -th entry of the matrix  $\mathbf{u} \otimes \mathbf{v} \in \mathbb{R}^{m \times n}$  for  $\mathbf{u} \in \mathbb{R}^m$  and  $\mathbf{v} \in \mathbb{R}^n$  is  $u_i v_j$ . For a vector  $\mathbf{v} \in \mathbb{R}^n$ , we let  $\|\mathbf{v}\|_p^p = \sum_{i=1}^n v_i^p$  and  $\|\mathbf{v}\|_\infty = \max_i |v_i|$ . For a matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$ , we denote the Frobenius norm of  $\mathbf{A}$  by  $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^d A_{i,j}^2}$ . We also use  $\|\mathbf{A}\|_p = \left( \sum_{i=1}^n \sum_{j=1}^d |A_{i,j}|^p \right)^{\frac{1}{p}}$ . For a function  $f$ , we use  $\nabla f$  to denote its gradient.

► **Definition 6.** A function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if  $f(\mathbf{x}) \geq f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle$  for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ .

► **Definition 7.** A continuously differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is  $\mu$ -smooth if

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq \mu \|\mathbf{x} - \mathbf{y}\|_2,$$

for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ . Then it follows, e.g., by Lemma 3.4 in [6], that for every  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ ,

$$|f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle| \leq \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|_2^2.$$

Recall that SGD offers the following convergence guarantees for smooth convex functions:



► **Theorem 8** ([26, 23]). Let  $F$  be a  $\mu$ -smooth convex function and  $\mathbf{x}_{\text{opt}} = \text{argmin } F(\mathbf{x})$ . Let  $\sigma^2$  be an upper bound for the variance of the unbiased estimator across all iterations and  $\bar{\mathbf{x}}_k = \frac{\mathbf{x}_1 + \dots + \mathbf{x}_k}{k}$ . Let each step-size  $\eta_t$  be  $\eta \leq \frac{1}{\mu}$ . Then for SGD with initial position  $\mathbf{x}_0$ , and any value of  $k$ ,

$$\mathbb{E}[F(\bar{\mathbf{x}}_k) - F(\mathbf{x}_{\text{opt}})] \leq \frac{1}{2\eta k} \|\mathbf{x}_0 - \mathbf{x}_{\text{opt}}\|_2^2 + \frac{\eta\sigma^2}{2}.$$

This means that  $k = \mathcal{O}\left(\frac{1}{\varepsilon^2} \left(\sigma^2 + \mu \|\mathbf{x}_0 - \mathbf{x}_{\text{opt}}\|_2^2\right)^2\right)$  iterations suffice to obtain an  $\varepsilon$ -approximate optimal value by setting  $\eta = \frac{1}{\sqrt{k}}$ .

## 2 $G$ -Sampler Algorithm

In this section, we describe our  $G$ -sampler, which reads a matrix  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$  and a vector  $\mathbf{x} \in \mathbb{R}^d$  given after processing the matrix  $\mathbf{A}$ , and outputs a gradient  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  among the  $n$  gradients  $\{G(\langle \mathbf{a}_1, \mathbf{x} \rangle - b_1, \mathbf{a}_1), \dots, G(\langle \mathbf{a}_n, \mathbf{x} \rangle - b_n, \mathbf{a}_n)\}$  with probability roughly  $\frac{\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2}{\sum_{j=1}^n \|G(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j, \mathbf{a}_j)\|_2}$ . However, it is not possible to exactly return  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  using sublinear space; we instead return a vector  $\mathbf{v}$  such that  $\mathbb{E}[\mathbf{v}] = G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  and  $\|\mathbf{v} - G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\| \leq \varepsilon \|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$ . To achieve our  $G$ -sampler, we first require a generalization of the standard  $L_2$ -heavy hitters algorithm COUNTSKETCH [7], which we describe in Section 2.1. We then describe our  $G$ -sampler in full in Section 2.2.

### 2.1 Heavy-Hitters

Before describing our generalization of COUNTSKETCH, we first require the following  $F_G$  estimation algorithm that generalizes both well-known frequency moment estimation algorithm of [1, 36] and symmetric norm estimation algorithm of [3] by leveraging the linear sketches used in those data structures to support “post-processing” with multiplication by any vector  $\mathbf{x} \in \mathbb{R}^d$ .

► **Theorem 9** ([3]). Given a constant  $\varepsilon > 0$  and an  $(\alpha, \varepsilon)$ -smooth gradient  $G$ , there exists a one-pass streaming algorithm ESTIMATOR that takes updates to entries of a matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$ , as well as vectors  $\mathbf{x} \in \mathbb{R}^d$  and  $\mathbf{b} \in \mathbb{R}^d$  that arrive after the stream, and outputs a quantity  $\hat{F}$  such that  $(1 - \varepsilon) \sum_{i \in [n]} \|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2 \leq \hat{F} \leq (1 + \varepsilon) \sum_{i \in [n]} \|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$ . The algorithm uses  $\frac{d^2}{\alpha^2} \text{polylog}(nT)$  bits of space and succeeds with probability at least  $1 - \frac{1}{\text{poly}(n, T)}$ .

We now describe a straightforward generalization of the  $L_2$ -heavy hitter algorithm COUNTSKETCH so that (1) it can find the “heavy rows” of a matrix  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$  rather than the “heavy coordinates” of a vector and (2) it supports post-processing multiplication by a vector  $\mathbf{x} \in \mathbb{R}^d$  that arrives only after  $\mathbf{A}$  is processed. Let  $\mathbf{A}_i = \mathbf{a}_i \otimes \mathbf{a}_i \in \mathbb{R}^{d \times d}$  for all  $i \in [n]$ . We define  $\text{tail}(c)$  to be the  $n - c$  rows that do not include the top  $c$  values of  $\|\mathbf{A}_i \mathbf{x}\|_2$ . For a given  $\varepsilon > 0$ , we say a block  $\mathbf{A}_i$  with  $i \in [n]$  is *heavy* if  $\|\mathbf{A}_i \mathbf{x}\|_2 \geq \varepsilon \sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_i \mathbf{x}\|_2$ .

The standard COUNTSKETCH algorithm for finding the  $L_2$ -heavy hitters among the coordinates of a vector  $\mathbf{v}$  of dimension  $n$  works by hashing the universe  $[n]$  across  $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$  buckets. Each coordinate  $i \in [n]$  is also given a random sign  $\sigma_i$  and so the algorithm maintains the  $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$  signed sums  $\sum \sigma_i x_i$  across all the coordinates hashed to each bucket. Then to estimate  $x_i$ , the algorithm simply outputs  $\sigma_i C_{h(i)}$ , where  $C_{h(i)}$  represents the counter corresponding to the bucket to which coordinate  $i$  hashes. It can be shown that

## 31:10 Adaptive Sketches for Robust Regression with Importance Sampling

$\mathbb{E}[\sigma_i C_{h(i)}] = x_i$ , where the expectation is taken over the random signs  $\sigma$  and the choices of the hash functions. Similarly, the variance of the estimator can be bounded to show that with constant probability, the estimator has additive error  $\mathcal{O}(\varepsilon) \|\mathbf{x}\|_2^2$  to  $x_i$  with constant probability. Thus if  $x_i > \varepsilon \|\mathbf{x}\|_2^2$ , the algorithm will be able to identify coordinate  $i$  as a heavy-hitter (in part by allowing some false positives). We give the algorithm in full in Algorithm 1.

■ **Algorithm 1** Output heavy vectors  $(\langle \mathbf{a}_i, \mathbf{x} \rangle) \mathbf{a}_i$ , where  $\mathbf{x}$  can be a vector that arrives after  $\mathbf{A}$  is processed.

**Input:** Matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$ , vector  $\mathbf{x} \in \mathbb{R}^d$ , accuracy parameter  $\varepsilon > 0$ , failure parameter  $\delta \in (0, 1]$ .

**Output:** Noisy vectors  $\mathbf{a}_i^\top \mathbf{a}_i \mathbf{x}$  with  $\|\mathbf{a}_i^\top \mathbf{a}_i \mathbf{x}\|_2^2 \geq \varepsilon^2 \sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{a}_i^\top \mathbf{a}_i \mathbf{x}\|_2^2$ .

- 1:  $b \leftarrow \Omega\left(\frac{1}{\delta \varepsilon^4}\right)$
- 2: Let  $\mathcal{T}$  contain  $b$  buckets, each initialized to the all zeros  $\mathbb{R}^{d \times d}$  matrix.
- 3: Let  $\sigma_i \in \{-1, +1\}$  be drawn from 4-wise independent family for  $i \in [n]$ .
- 4: Let  $h : [n] \rightarrow [b]$  be 2-wise independent
- 5: **Process  $\mathbf{A}$ :**
- 6: Let  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n$ , where each  $\mathbf{a}_i \in \mathbb{R}^d$ .
- 7: **for** each  $j = 1$  to  $n$  **do**
- 8:      $\mathbf{A}_j \leftarrow \mathbf{a}_j \otimes \mathbf{a}_j$
- 9:     Add  $\sigma_j \mathbf{A}_j$  to the matrix in bucket  $h(j)$ .
- 10: Let  $\mathbf{M}_j$  be the matrix in bucket  $j$  of  $\mathcal{T}$  for  $i \in [r], j \in [b]$ .
- 11: **Process  $\mathbf{x}$ :**
- 12: **for**  $j \in [b]$  **do**
- 13:      $\mathbf{v}_j \leftarrow \mathbf{M}_j \mathbf{x}$
- 14: On query  $k \in [n]$ , report  $\sigma_k \mathbf{v}_{h(k)}$ .

Thus Algorithm 1 can be used to give the following guarantee by taking the median of the norms of  $\mathcal{O}(\log(nT))$  copies, as well as the vector that realizes the median.

► **Lemma 10** ([22]). *There exists an algorithm that uses  $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2 n\right)$  space and outputs a set  $S$  of indices so that with probability  $1 - \frac{1}{\text{poly}(n, T)}$ , for all  $i \in [n]$ ,  $i \in S$  if  $\|\mathbf{A}_i \mathbf{x}\|_2 \geq \varepsilon \sum_{j \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_j \mathbf{x}\|_2$  and  $i \notin S$  if  $\|\mathbf{A}_i \mathbf{x}\|_2 \leq \frac{\varepsilon}{2} \sum_{j \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_j \mathbf{x}\|_2$ . The algorithm uses  $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2(nT)\right)$  space.*

However, the vector that realizes the median of the norms may no longer be an unbiased estimate to each heavy-hitter. Unfortunately, we shall require unbiased estimates to each heavy-hitter, because we will use estimated heavy-hitters as unbiased gradients as part of SGD with importance sampling. Thus we give an additional algorithm so that for each  $i \in S$  reported by Algorithm 1, the algorithm outputs an *unbiased* estimate to the vector  $(\langle \mathbf{a}_i, \mathbf{x} \rangle) \mathbf{a}_i$  with a “small” error, in terms of the total mass  $\sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_i \mathbf{x}\|_2$  excluding the largest  $\frac{2}{\varepsilon^2}$  rows.

To that end, we instead run  $d$  separate instances of COUNTSKETCH to handle the  $d$  separate coordinates of each heavy-hitter  $\mathbf{A}_i \mathbf{x}$ . We show that the median of each estimated coordinate is an unbiased estimate to the coordinate  $(\mathbf{A}_i \mathbf{x})_j$ , since the probability mass function is symmetric about the true value for each coordinate. Furthermore, we show that although the error to a single coordinate  $(\mathbf{A}_i \mathbf{x})_j$  may be large compared to  $|(\mathbf{A}_i \mathbf{x})_j|$ , the error is not large compared to  $\sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_i \mathbf{x}\|_2$ .

► **Lemma 11.** *There exists an algorithm that uses  $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2(nT)\right)$  space and outputs a vector  $\mathbf{y}_i$  for each index  $i \in [n]$  so that  $|\|\mathbf{y}_i\|_2 - \|\mathbf{A}_i \mathbf{x}\|_2| \leq \varepsilon \sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_i \mathbf{x}\|_2$  and  $\mathbb{E}[\mathbf{y}_i] = \mathbf{A}_i \mathbf{x}$  with probability at least  $1 - \frac{1}{\text{poly}(n, T)}$ .*

However if say, we want to identify the heavy gradients  $(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i) \mathbf{a}_i$ , then we create separate data structures for the constant (in  $\mathbf{x}$ ) term  $b_i \mathbf{a}_i$  and the linear term  $(\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}$ , using the same buckets, hash functions, and random signs. For the constant term data structure, we hash the scaled rows  $b_i \mathbf{a}_i$  into  $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$  buckets, so that each bucket contains a vector that represents the signed sum of the (scaled) rows of  $\mathbf{A}$  that hash to the bucket. For the linear term data structure, we hash the outer products  $\mathbf{A}_i := \mathbf{a}_i \otimes \mathbf{a}_i$  into  $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$  buckets, so that each bucket contains a vector that represents the signed sum of the matrices  $\mathbf{A}_i$  that hash to the bucket. Once the vector  $\mathbf{x}$  arrives after  $\mathbf{A}$  is processed, then we can multiply each of the matrices stored by each bucket by  $\mathbf{x}$ . Since the signed sum is a linear sketch, this procedure is equivalent to originally taking the signed sums of the vectors  $\mathbf{A}_i \mathbf{x}$ . Similarly, by linearity, we can then take any linear combination of the two data structures to identify the heavy gradients  $(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i) \mathbf{a}_i$ .

## 2.2 G-Sampler Algorithm

In this section, we first describe our  $G$ -sampler algorithm, where we sample a gradient  $G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  with probability proportional to  $\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$ . Given an accuracy parameter  $\varepsilon > 0$ , let  $\alpha$  be a constant, parametrized by  $\varepsilon$ , so that  $(1 - \varepsilon)F_G(\mathbf{v}) \leq F_G(\mathbf{u}) \leq (1 + \varepsilon)F_G(\mathbf{v})$ , for any  $\mathbf{u}$  with  $\|\mathbf{u} - \mathbf{v}\|_2 \leq \alpha \|\mathbf{v}\|_2$ . As our data structure will be a linear sketch, we focus on the case where we fold the measurement vector  $\mathbf{b}$  into a column of  $\mathbf{A}$ , so that we want to output a gradient  $\mathbf{A}_i \mathbf{x} := (\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}$ .

Our algorithm first partitions the rows of  $\mathbf{A}$  into classes, based on their  $L_2$  norm. For example, if all entries of  $\mathbf{A}$  are integers, then we define class  $C_k := \{\mathbf{a}_i : 2^{k-1} \leq \|\mathbf{a}_i\|_2 < 2^k\}$ . We create a separate data structure for each class. We will use the  $F_G$  estimation algorithm on each class to first sample a particular class. It then remains to sample a particular vector  $(\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}$  from a class.

Depending on the vector  $\mathbf{x}$ , the vectors  $(\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}$  in a certain class  $C_k$  can have drastically different  $L_2$  norm. We define level set  $\Gamma_j$  as the vectors that satisfy  $(1 + \varepsilon)^{j-1} \leq \|(\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}\|_2 < (1 + \varepsilon)^j$ . If we could estimate  $|\Gamma_j|$ , then we could estimate the contribution of each level set  $\Gamma_j$  toward the overall mass  $\sum_{i \in C_k} \|(\mathbf{a}_i \otimes \mathbf{a}_i) \mathbf{x}\|_2$ , so we can then sample a specific level set  $\Gamma_j$  from the class  $C_k$ . To that end, we create  $L = \mathcal{O}(\log n)$  substreams,  $S_1, \dots, S_L$ , so that we sample each row with probability  $\frac{1}{2^{L-1}}$  in substream  $S_\ell$ .

The point is that if the contribution of level set  $\Gamma_j$  is “significant”, then there exists a specific substream  $S_\ell$  in which the vectors of  $\Gamma_j$  will be likely detected by the heavy-hitter algorithm we introduced in Section 2.1, if they are sampled by  $S_\ell$ . We can then use these vectors that are output by the heavy-hitter algorithm to estimate the contribution of level set  $\Gamma_j$ . However, if the contribution of level set  $\Gamma_j$  is not significant, then there may not be any vectors of  $\Gamma_j$  that survive the sampling in substream  $S_\ell$ . Thus we add a number of “dummy rows” to each level set to insist that all level sets are significant, so that we can estimate their contributions.

We then sample a level set  $\Gamma_j$  with probability proportional to its contribution and uniformly select a (noisy) vector from the level set. If the selected vector is one of the original rows of the matrix, then we output the noisy vector. Otherwise, we say the sampler has failed. We show that the sampler only fails with constant probability, so it suffices to run  $\mathcal{O}(\log \frac{1}{\delta})$  independent instances to boost the probability of success to any arbitrary  $1 - \delta$ . The algorithm for selecting a level set  $\Gamma_j$  from a specific class  $C_k$  appears in Algorithm 2.

## 31:12 Adaptive Sketches for Robust Regression with Importance Sampling

We first show that the dummy rows only contribute at constant multiple of the mass  $F_G(S) = \sum_{i=1}^n \|\mathbf{A}_i \mathbf{x}\|_2$ , where we assume for simplicity that all rows of  $\mathbf{A}$  are in the same class.

► **Lemma 12.** *Let  $S$  be the input data stream with subsamples  $S_1, \dots, S_L$ . Let  $\tilde{S}$  be the input data stream with the additional dummy rows and corresponding subsamples  $\tilde{S}_1, \dots, \tilde{S}_L$ . Then  $2F_G(S) \geq F_G(\tilde{S}) \geq F_G(S)$ .*

We would now like to show that with high probability, each of the substreams have exponentially smaller mass  $F_G(S_j)$ . However, this may not be true. Consider a single row  $\mathbf{a}_i$  that contributes a constant fraction of  $F_G(S)$ . Then even for  $j = \log n$ , the probability that  $\mathbf{a}_i$  is sampled is roughly  $\frac{1}{n} \gg \frac{1}{\text{poly}(n)}$ . Instead, we note that COUNTSKETCH satisfies the stronger tail guarantee in Lemma 11. Hence for each  $j \in [K]$ , we define  $S_j^{\text{tail}(t)}$  to be the frequency vector  $S_j$  with its  $t$  largest entries set to zero and we show an exponentially decreasing upper bound on  $F_G(\widetilde{S_j^{\text{tail}(t)}})$ .

► **Lemma 13.** *With high probability, we have that for all  $j \in [K]$ ,  $F_G(\widetilde{S_j^{\text{tail}(t)}}) \leq \frac{F_G(S)}{2^j} \log(nT)$  for  $t = \mathcal{O}\left(\frac{\log n}{\alpha^3}\right)$ .*

We also show that the estimated contribution of each level set (after incorporating the dummy rows) is a  $(1 + \alpha)$ -approximation of the true contribution.

► **Lemma 14.** *With high probability, we have that for all  $j \in [K]$ ,  $(1 - \varepsilon)F_G(\tilde{S}_j) \leq \widetilde{F}_G(\tilde{S}_j) \leq (1 + \varepsilon)F_G(\tilde{S}_j)$ .*

Finally, we show that each row is sampled with the correct distribution and is an unbiased estimate.

► **Lemma 15.** *Suppose that  $2^k < \|\mathbf{a}_i\|_2 \leq 2^{k+1}$  for all  $i \in [n]$ . Then the probability that Algorithm 2 outputs a noisy vector  $\mathbf{v}$  such that  $\|\mathbf{v} - \mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)\|_2 \leq \alpha \|\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)\|_2$  with  $\mathbb{E}[\mathbf{v}] = \mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$  is  $p_{\mathbf{v}} = (1 \pm \mathcal{O}(\varepsilon)) \frac{G(\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i))}{F_G(\tilde{S})} + \frac{1}{\text{poly}(nT)}$ .*

Putting things together, we have the guarantees of Theorem 5 for our  $G$ -sampler.

### 3 SGD Algorithm and Analysis

Before introducing our main SGD algorithm, we recall the following algorithm, that essentially outputs noisy version of the rows with high “importance”. Although SAMPLER outputs a (noisy) vector according to the desired probability distribution, we also require an algorithm that automatically does this for indices  $i \in [n]$  that are likely to be sampled multiple times across the  $T$  iterations. Equivalently, we require explicitly storing the rows with high sensitivities.

► **Theorem 16 ([5]).** *Given a constant  $\varepsilon > 0$ , there exists an algorithm SENS that returns all indices  $i \in [n]$  such that  $\sup_x \frac{|\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)|}{\sum_{j=1}^n |\mathbf{a}_j(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j)|} \geq \frac{1}{200Td}$  for some  $\mathbf{x} \in \mathbb{R}^n$ , along with the vector  $\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)$ . The algorithm requires a single pass over  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n$ , uses  $\tilde{\mathcal{O}}(nd^2 + Td^2)$  runtime and  $\tilde{\mathcal{O}}(Td^2)$  space, and succeeds with probability  $1 - \frac{1}{\text{poly}(n)}$ .*

The quantity  $\sup_x \frac{\|\mathbf{a}_i(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i)\|_2}{\sum_{j=1}^n \|\mathbf{a}_j(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j)\|_2}$  can be considered the *sensitivity* of row  $\mathbf{a}_i$  and can be interpreted as a measure of “importance” of the row  $\mathbf{a}_i$  with respect to the other rows of  $\mathbf{A}$ .

■ **Algorithm 2**  $G$ -sampler for a single class of rows.

---

**Input:** Rows  $\mathbf{a}_1, \dots, \mathbf{a}_n$  of a matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$  with  $2^k \leq \|\mathbf{a}_i\|_2 < 2^{k+1}$  for all  $i \in [n]$ , function  $G$ , accuracy parameter  $\alpha$  for sampling parameter  $\varepsilon$

**Output:** Noisy row  $\mathbf{v}$  with the correct sampling distribution induced by  $G$

- 1:  $\gamma$  uniformly at random from  $[1/2, 1]$ ,  $K \leftarrow \mathcal{O}\left(\frac{\log n}{\alpha}\right)$ ,  $L \leftarrow \mathcal{O}(\log n)$
- 2: **for**  $\ell \in [L]$  **do** ▷Processing stage
- 3:   Form a stream  $S_\ell$  by sampling each row with probability  $2^{-\ell+1}$
- 4:   Run COUNTSKETCH $_\ell^{(1)}$  with threshold  $\mathcal{O}\left(\frac{\alpha^3}{\log n}\right)$  and failure probability  $\frac{1}{\text{poly}(n, T)}$  by creating a table  $A_\ell^{(1)}$  with entries  $\mathbf{a}_j^\top \mathbf{a}_j$  in  $S_\ell$  and a table  $B_\ell^{(1)}$  with entries  $\mathbf{a}_j$  ▷Identify heavy-hitters
- 5:   Run COUNTSKETCH $_\ell^{(2)}$  with threshold  $\mathcal{O}\left(\frac{\alpha^3}{\log n}\right)$  and failure probability  $\frac{1}{\text{poly}(n, T)}$  by creating a table  $A_\ell^{(2)}$  with entries  $\mathbf{a}_j^\top \mathbf{a}_j$  in  $S_\ell$  and a table  $B_\ell^{(2)}$  with entries  $\mathbf{a}_j$  and separately considering coordinates after post-processing ▷Unbiased estimates of heavy-hitters, see Lemma 11
- 6: **for**  $\ell \in [L]$  **do** ▷Post-processing
- 7:   Set  $C_\ell^{(i)} = A_\ell^{(i)} \mathbf{x} + B_\ell^{(i)}$  with post-multiplication by  $\mathbf{x}$  for  $i \in \{1, 2\}$
- 8:   Query  $\widehat{M} \in [M/2, 2M]$ , where  $M = \sum_{i=1}^n \|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2$
- 9:   **for**  $j \in [K]$  **do**
- 10:     **if**  $j > \log_{(1+\alpha)} \frac{\log^2 n}{\alpha^3}$  **then**
- 11:       Add  $\mathcal{O}\left(\frac{(1+\alpha)^j \alpha^3}{\log n}\right)$  dummy rows that each contribute  $\mathcal{O}\left(\frac{\widehat{M}}{(1+\alpha)^j \alpha^2}\right)$  to  $F_G$
- 12:   Let  $H_\ell^{(i)}$  be the heavy rows of  $C_\ell^{(i)}$  for  $i \in \{1, 2\}$  from COUNTSKETCH $_\ell^{(i)}$
- 13: **for**  $j \in [K]$  **do**
- 14:    $L_j \leftarrow \max\left(1, \log \frac{\alpha^2(1+\alpha)^j}{\log n}\right)$
- 15:   Let  $X_j$  be the estimated heavy-hitters  $\mathbf{v}$  from  $H_j^{(2)}$  that are reported by  $H_j^{(1)}$  with  $G(\mathbf{v})$  in  $\left[\frac{8\gamma\widehat{M}}{(1+\alpha)^{j+1}}, \frac{8\gamma\widehat{M}}{(1+\alpha)^j}\right)$
- 16:   **if**  $L_j = 1$  **then**
- 17:      $\widetilde{F}_G(\widetilde{S}_j) \leftarrow \sum_{\mathbf{v} \in X_j} \frac{8\gamma\widehat{M}}{(1+\alpha)^{j+1}}$
- 18:   **else if**  $L_j > 1$  and  $|X_j| > \frac{1}{\alpha^2}$  **then**
- 19:      $\widetilde{F}_G(\widetilde{S}_j) \leftarrow \sum_{\mathbf{v} \in X_j} \frac{8\gamma\widehat{M}}{(1+\alpha)^{j+1}} \cdot 2^{L_j}$
- 20:   **else**
- 21:      $\widetilde{F}_G(\widetilde{S}_j) \leftarrow 0$
- 22: Sample  $j \in [K]$  with probability  $\frac{\widetilde{F}_G(\widetilde{S}_j)}{\sum \widetilde{F}_G(\widetilde{S}_j)}$
- 23: Sample  $\mathbf{v}$  from  $X_j$  with probability  $\frac{1}{|X_j|}$
- 24: **if**  $\mathbf{v}$  is a dummy row **then**
- 25:   **return**  $\perp$
- 26: **else**
- 27:   **return**  $\mathbf{v}$

---

We now proceed to describe our main SGD algorithm. For the finite-sum optimization problem  $\min_{\mathbf{x} \in \mathbb{R}^d} F(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^n G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$ , where each  $G$  is a piecewise function of a polynomial with degree at most 1, recall that we could simply use an instance of SAMPLER as an oracle for SGD with importance sampling. However, naïvely running  $T$  SGD steps

requires  $T$  independent instances, which uses  $Tnd$  runtime by Theorem 5. Thus, as our main theoretical contribution, we use a two level data structure by first implicitly partitioning the rows of matrix  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n$  into  $\beta := \Theta(T)$  buckets  $B_1, \dots, B_\beta$  and creating an instance of ESTIMATOR and SAMPLER for each bucket. The idea is that for a given query  $\mathbf{x}_t$  in SGD iteration  $t \in [T]$ , we first query  $\mathbf{x}_t$  to each of the ESTIMATOR data structures to estimate  $\sum_{i \in B_j} G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)$  for each  $j \in [\beta]$ . We then sample index  $j \in [\beta]$  among the buckets  $B_1, \dots, B_\beta$  with probability roughly  $\frac{\sum_{i \in B_j} G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)}{\sum_{i=1}^n G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)}$ . Once we have sampled index  $j$ , it would seem that querying the instance SAMPLER corresponding to  $B_j$  simulates SGD, since SAMPLER now performs importance sampling on the rows in  $B_j$ , which gives the correct overall probability distribution for each row  $i \in [n]$ . Moreover, SAMPLER has runtime proportional to the sparsity of  $B_j$ , so the total runtime across the  $\beta$  instances of SAMPLER is  $\tilde{O}(nd)$ .

However, an issue arises when the same bucket  $B_j$  is sampled multiple times, as we only create a single instance of SAMPLER for each bucket. We avoid this issue by explicitly accounting for the buckets that are likely to be sampled multiple times. Namely, we show that if  $\frac{G(\langle \mathbf{a}_i, \mathbf{x}_t \rangle - b_i, \mathbf{a}_i)}{\sum_{j=1}^n G(\langle \mathbf{a}_j, \mathbf{x}_t \rangle - b_j, \mathbf{a}_j)} < \mathcal{O}(\frac{1}{T})$  for all  $t \in [T]$  and  $i \in [n]$ , then by Bernstein's inequality, the probability that no bucket  $B_j$  is sampled at least  $2 \log T$  times is at least  $1 - \frac{1}{\text{poly}(T)}$ . Thus we use SENS to separate all such rows  $\mathbf{a}_i$  whose sensitivities violate this property from their respective buckets and explicitly track the SGD steps in which these rows are sampled.

The natural approach would be to create  $T$  samplers for each of the rows with sensitivity at least  $\Omega(\frac{1}{T})$ , ensuring that each of these samplers has access to fresh randomness in each of the  $T$  SGD steps. However since the sensitivities sum to  $\mathcal{O}(d \log n)$ , there can be up to  $\mathcal{O}(Td \log n)$  rows with sensitivity at least  $\Omega(\frac{1}{T})$ , so creating  $T$  samplers for each of these rows could create up to  $\Theta(T^2 d \log n)$  samplers, which is prohibitively expensive in  $T$ . Instead, we simply keep each row with sensitivity at least  $\Omega(\frac{1}{T})$  explicitly, while not including them in the bucket. Due to the monotonicity of sensitivities, the sensitivity of each row may only decrease as the stream progresses. In the case that a row had sensitivity at least  $\Omega(\frac{1}{T})$  at some point, but then no longer exceeds the threshold at some later point, then the row is given as input to the sampler corresponding to the bucket to which the row hashes and then the explicit storage of the row is deleted. This ensures we need only  $\tilde{O}(Td)$  samplers while still avoiding any sampler from being used multiple times across the  $T$  SGD steps. We give the algorithm in full in Algorithm 3.

The key property achieved by Algorithm 3 in partitioning the rows and removing the rows that are likely to be sampled multiple times is that each of the SAMPLER instances are queried at most once.

► **Lemma 17.** *With probability at least  $\frac{98}{100}$ , each  $t \in [T]$  uses a different instance of  $\text{SAMPLER}_j$ .*

Theorem 4 then follows from Lemma 17 and the sampling distribution guaranteed by each subroutine in Lemma 15. In particular, Lemma 17 crucially guarantees that each step  $t \in [T]$  of SGD will receive a vector with fresh independent randomness. Moreover, we have that each (noisy) vector has small variance and is an unbiased estimate of a subgradient sampled from nearly the optimal importance sampling probability distribution.

► **Theorem 4.** *Given an input matrix  $\mathbf{A} \in \mathbb{R}^{n \times d}$  whose rows arrive sequentially in a data stream along with the corresponding labels of a measurement vector  $\mathbf{b} \in \mathbb{R}^d$ , and a measure function  $M$  whose derivative is a continuous union of piecewise constant or linear functions, there exists an algorithm that performs  $T$  steps of SGD with variance within a constant factor of the optimal sampling distribution. The algorithm uses  $\tilde{O}(nd^2 + Td^2)$  pre-processing time and  $Td^2 \text{polylog}(Tnd)$  words of space.*



■ **Algorithm 3** Approximate SGD with Importance Sampling.

**Input:** Matrix  $\mathbf{A} = \mathbf{a}_1 \circ \dots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ , parameter  $T$  for number of SGD steps.

**Output:**  $T$  gradient directions.

- 1: **Preprocessing Stage:**
- 2:  $\beta \leftarrow \Theta(T)$  with a sufficiently large constant in the  $\Theta$ .
- 3: Let  $h : [n] \rightarrow [\beta]$  be a uniformly random hash function.
- 4: Let  $\mathbf{B}_j$  be the matrix formed by the rows  $\mathbf{a}_i$  of  $\mathbf{A}$  with  $h(i) = j$ , for each  $j \in [\beta]$ .
- 5: Create  $\Theta(\log(Td))$  instances  $\text{ESTIMATOR}_j$  and  $\text{SAMPLER}_j$  for each  $\mathbf{B}_j$  with  $j \in [\beta]$  with  $\varepsilon = \frac{1}{2}$ .
- 6: Run  $\text{SENS}$  to find a set  $L_0$  of rows with sensitivity at least  $\Omega(\frac{1}{T})$ .
- 7: **Gradient Descent Stage:**
- 8: Randomly pick starting location  $\mathbf{x}_0$
- 9: **for**  $t = 1$  to  $T$  **do**
- 10:   Let  $q_i$  be the output of  $\text{ESTIMATOR}_j$  on query  $\mathbf{x}_{t-1}$  for each  $i \in [\beta]$ .
- 11:   Sample  $j \in [\beta]$  with probability  $p_j = \frac{q_j}{\sum_{i \in [\beta]} q_i}$ .
- 12:   **if** there exists  $i \in L_0$  with  $h(i) = j$  **then**
- 13:     Use  $\text{ESTIMATOR}_j$ ,  $L_0$ , and  $\text{SAMPLER}_j$  to sample gradient  $\mathbf{w}_t = \widehat{\nabla f_{i_t}(\mathbf{x}_t)}$
- 14:   **else**
- 15:     Use fresh  $\text{SAMPLER}_j$  to sample gradient  $\mathbf{w}_t = \widehat{\nabla f_{i_t}(\mathbf{x}_t)}$
- 16:    $\widehat{p}_{i,t} \leftarrow \frac{\|\mathbf{w}_t\|_2^2}{\sum_{j \in [\beta]} q_j}$
- 17:    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \frac{\eta_t}{np_{i,t}} \cdot \mathbf{w}_t$

**Proof.** Consider Algorithm 3. By Lemma 17, each time  $t \in [T]$  uses a fresh instance of  $\text{SAMPLER}_j$ , so that independent randomness is used. A possible concern is that each instance  $\text{ESTIMATOR}_j$  is not using fresh randomness, but we observe that the  $\text{ESTIMATOR}$  procedures are only used in sampling a bucket  $j \in [\beta]$ ; otherwise the sampling uses fresh randomness whereas the sampling is built into each instance of  $\text{SAMPLER}_j$ . By Theorem 5, each index  $i$  is sampled with probability within a factor 2 of the importance sampling probability distribution. By Theorem 9, we have that  $\widehat{p}_{i,t}$  is within a factor 4 of the probability  $p_{i,t}$  induced by optimal importance sampling SGD. Note that  $\mathbf{w}_t = G(\langle \mathbf{a}_i, \mathbf{x}_t \rangle - b_i, \mathbf{a}_i)$  is an unbiased estimator of  $G(\langle \mathbf{a}_i, \mathbf{x}_t \rangle - b_i, \mathbf{a}_i)$  and  $G(\mathbf{w}_t)$  is a 2-approximation to  $G(\mathbf{x}_t)$  by Theorem 5. Hence, the variance at each time  $t \in [T]$  of Algorithm 3 is within a constant factor of the variance  $\sigma^2 = (\sum_{i=1}^n G(\langle \mathbf{a}_i, \mathbf{x}_t \rangle - b_i, \mathbf{a}_i))^2 - \sum_{i=1}^n G(\langle \mathbf{a}_i, \mathbf{x}_t \rangle - b_i, \mathbf{a}_i)^2$  of optimal importance sampling SGD.

By Theorem 5, Theorem 9, and Theorem 16, the preprocessing time is  $d^2 \text{polylog}(nT)$  for  $\varepsilon = \mathcal{O}(1)$  and  $\beta = \Theta(T)$ , but partitioning the non-zero entries of  $\mathbf{A}$  across the  $\beta$  buckets and the space used by the algorithm is  $\tilde{\mathcal{O}}(Td^2)$ . Once the gradient descent stage of Algorithm 3 begins, it takes  $Td^2 \text{polylog}(n)$  time in each step  $t \in [T]$  to query the  $\beta = \Theta(T)$  instances of  $\text{SAMPLER}$  and  $\text{ESTIMATOR}$ , for total time  $Td^2 \text{polylog}(n)$ . ◀

Finally, we derandomize our algorithm in Appendix B with an extra logarithmic factor in the space complexity by using the following formulation of Nisan's pseudorandom generator:

► **Theorem 18** (Nisan's Pseudorandom Generator). [28] *Let  $\mathcal{A}$  be an algorithm that uses  $S = \Omega(\log n)$  space and  $R$  random bits. Then there exists a pseudorandom generator for  $\mathcal{A}$  that succeeds with high probability and runs in  $\mathcal{O}(S \log R)$  bits.*



## References

- 1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.
- 2 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 363–372, 2011.
- 3 Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 716–729, 2017.
- 4 Michael Bosse, Gabriel Agamennoni, and Igor Gilitschenski. Robust estimation and applications in robotics. *Found. Trends Robotics*, 4(4):225–269, 2016.
- 5 Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020.
- 6 Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.
- 7 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.
- 8 Kenneth L. Clarkson, Ruosong Wang, and David P. Woodruff. Dimensionality reduction for tukey regression. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 1262–1271, 2019.
- 9 Kenneth L. Clarkson and David P. Woodruff. Sketching for  $M$ -estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 921–939, 2015.
- 10 Hadi Daneshmand, Aurélien Lucchi, and Thomas Hofmann. Starting small - learning with adaptive sample sizes. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pages 1463–1471, 2016.
- 11 Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 1646–1654, 2014.
- 12 Antoine Guitton and William W Symes. Robust and stable velocity analysis using the huber function. In *SEG Technical Program Expanded Abstracts 1999*, pages 1166–1169. Society of Exploration Geophysicists, 1999.
- 13 Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.
- 14 Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 202–208, 2005.
- 15 Rajesh Jayaram and David P. Woodruff. Perfect  $l_p$  sampling in a data stream. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 544–555, 2018.
- 16 Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *PODS: International Conference on Management of Data*, pages 29–40, 2022.
- 17 Yifei Jiang, Yi Li, Yiming Sun, Jiaxin Wang, and David P. Woodruff. Single pass entrywise-transformed low rank approximation. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 4982–4991, 2021.
- 18 Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26, Proceedings.*, pages 315–323, 2013.

- 19 Tyler B. Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 7276–7286, 2018.
- 20 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 49–58, 2011.
- 21 Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 2530–2539, 2018.
- 22 Sepideh Mahabadi, Ilya P. Razenshteyn, David P. Woodruff, and Samson Zhou. Non-adaptive adaptive sampling on turnstile streams. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1251–1264, 2020.
- 23 Raghu Meka. Cs289ml: Algorithmic machine learning notes, 2017. URL: <https://raghumeika.github.io/CS289ML/gdnotes.pdf>.
- 24 Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error lp-sampling with applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1143–1160, 2010.
- 25 Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Math. Program.*, 155(1-2):549–573, 2016.
- 26 Arkadi Nemirovski, Anatoli B. Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.
- 27 Arkadi Semenovich Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization, 1983.
- 28 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
- 29 Eric Price, Sandeep Silwal, and Samson Zhou. Hardness and algorithms for robust and sparse optimization. In *International Conference on Machine Learning, ICML*, pages 17926–17944, 2022.
- 30 Xun Qian, Peter Richtárik, Robert M. Gower, Alibek Sailanbayev, Nicolas Loizou, and Egor Shulgin. SGD with arbitrary sampling: General analysis and improved rates. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, pages 5200–5209, 2019.
- 31 Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 2647–2655, 2015.
- 32 Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
- 33 Nicolas Le Roux, Mark Schmidt, and Francis R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems.*, pages 2672–2680, 2012.
- 34 Farnood Salehi, Patrick Thiran, and L. Elisa Celis. Coordinate descent with bandit sampling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 9267–9277, 2018.
- 35 Sebastian U. Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 4381–4391, 2017.
- 36 Mikkel Thorup and Yin Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 615–624. SIAM, 2004.

- 37 Murad Tukan, Xuan Wu, Samson Zhou, Vladimir Braverman, and Dan Feldman. New coresets for projective clustering and applications. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2022.
- 38 Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.
- 39 Zhengyou Zhang. Parameter estimation techniques: a tutorial with application to conic fitting. *Image Vis. Comput.*, 15(1):59–76, 1997.
- 40 Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning ICML*, pages 1–9, 2015.

## A Missing Proofs from Section 2

**Proof of Lemma 11.** Given  $\mathbf{A} \in \mathbb{R}^{n \times d}$ , let  $\mathbf{A}_i = \mathbf{a}_i^\top \mathbf{a}_i$  for all  $i \in [n]$ . For a fixed coordinate  $k \in [d]$ , we define a vector  $\mathbf{v}^{(k)} \in \mathbb{R}^n$  so that for each  $i \in [n]$ , the  $i$ -th coordinate of  $\mathbf{v}^{(k)}$  is the  $k$ -th coordinate of  $\mathbf{A}_i \mathbf{x} \in \mathbb{R}^d$ .

Suppose we run a separate COUNTSKETCH instance on  $\mathbf{v}^{(k)}$ . For a fixed index  $i \in [n]$ , let  $h(i)$  be the bucket of  $\mathcal{T}$  to which  $v_i^{(k)}$  hashes. For each  $j \in [n]$ , let  $I_j$  be the indicator variable for whether  $v_j^{(k)}$  also hashes to bucket  $h(i)$ , so that  $I_j = 1$  if  $h(i) = h(j)$  and  $I_j = 0$  if  $h(i) \neq h(j)$ . Similarly for each  $j \in [n]$ , let  $s_j$  be a random sign assigned to  $j$ , so that the estimate for  $v_i^{(k)}$  by a single row of COUNTSKETCH is

$$\sum_{j \in [n]} s_i s_j I_j v_j^{(k)} = v_i^{(k)} + \sum_{j: h(j) = h(i)} r_j v_j^{(k)},$$

where  $r_j = s_i s_j$  satisfies  $r_j = 1$  with probability  $\frac{1}{2}$  and  $r_j = -1$  with probability  $\frac{1}{2}$ . Thus if  $y_i$  is the estimate for  $v_i^{(k)}$ , then for any real number  $u$ , we have that

$$\Pr[y_i = v_i^{(k)} + u] = \Pr[y_i = v_i^{(k)} - u],$$

so that the probability mass function of  $y_i$  is symmetric about  $v_i^{(k)}$ . Thus given  $\ell$  independent instances of COUNTSKETCH with estimates  $y_i^{(k,1)}, \dots, y_i^{(k,\ell)}$  for  $v_i^{(k)}$  and any real numbers  $u^{(1)}, \dots, u^{(\ell)}$ ,

$$\Pr[y_i^{(k,1)} = v_i^{(k)} + u^{(1)}, \dots, y_i^{(k,\ell)} = v_i^{(k)} + u^{(\ell)}] = \Pr[y_i^{(k,1)} = v_i^{(k)} - u^{(1)}, \dots, y_i^{(k,\ell)} = v_i^{(k)} - u^{(\ell)}].$$

Therefore, the joint probability mass function is symmetric about  $(v_i^{(k)}, \dots, v_i^{(k)})$  and so the median across the  $\ell$  instances of COUNTSKETCH is an unbiased estimator to  $v_i^{(k)}$ . Finally, we have due to the properties of COUNTSKETCH that if each hash function  $h$  maps to a universe of size  $\mathcal{O}(\frac{1}{\varepsilon^2})$  and  $\ell = \mathcal{O}(\log(nT))$ , then with probability at least  $1 - \frac{1}{\text{poly}(T,n)}$ , the output estimate for  $v_i^{(k)}$  has additive error at most  $\varepsilon \cdot \left(\sum_{j \in \text{tail}(2/\varepsilon^2)} (v_i^{(k)})^2\right)^{1/2}$ .

Thus using each of the estimated outputs across all  $k \in [d]$ , then for a fixed  $i \in [n]$ , we can output a vector  $\mathbf{y}_i$  such that  $\mathbb{E}[\mathbf{y}_i] = \mathbf{A}_i \mathbf{x}$  and with probability at least  $1 - \frac{1}{\text{poly}(T,n)}$ ,

$$|\|\mathbf{y}_i\|_2 - \|\mathbf{A}_i \mathbf{x}\|_2| \leq \varepsilon \cdot \left(\sum_{i \in \text{tail}(2/\varepsilon^2)} \|\mathbf{A}_i \mathbf{x}\|_2^2\right)^{1/2}.$$

For a fixed  $k \in [d]$ , then our algorithm intends to hash the  $k$ -th coordinate of  $\mathbf{A}_i \mathbf{x} \in \mathbb{R}^d$ . However, since  $\mathbf{x}$  is only given after the data structure is already formed and in particular, after  $\mathbf{A}_i$  is given, then COUNTSKETCH must hash the  $k$ -th row of  $\mathbf{A}_i$  entirely, thus storing  $\mathcal{O}\left(\frac{d}{\varepsilon^2} \log^2(nT)\right)$  bits for each coordinate  $k \in [d]$ . Hence across all  $k \in [d]$ , the algorithm uses the total space  $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2(nT)\right)$ . ◀

**Proof of Lemma 12.** Since  $\tilde{S}$  includes all the rows of  $S$ , then  $F_G(\tilde{S}) \geq F_G(S)$ . Since each level  $j \in [K]$  acquires  $\mathcal{O}\left(\frac{(1+\alpha)^j \alpha^3}{\log n}\right)$  dummy rows that each contribute  $\mathcal{O}\left(\frac{\hat{M}}{(1+\alpha)^j \alpha^2}\right)$  to  $F_G$  in  $\tilde{S}$ , then each level of  $F_G(S)$  contributes at most  $\mathcal{O}\left(\frac{\hat{M} \cdot \alpha}{\log n}\right)$  more to  $F_G(\tilde{S})$ . Because  $K = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$ , then the total additional contribution by the dummy rows is at most  $\mathcal{O}\left(\hat{M}\right)$ . Since  $\hat{M} \leq 2F_G(S)$ , then it follows that for sufficiently small constant in the contribution of each dummy row, we have  $F_G(\tilde{S}) - F_G(S) \leq F_G(S)$  and thus,  $F_G(\tilde{S}) \leq 2F_G(S)$ . ◀

**Proof of Lemma 13.** Observe that the number of rows that exceed  $\frac{\hat{M}}{2^j}$  is at most  $2^{j+1}$ . Thus the expected number of rows that exceed  $\frac{\hat{M}}{2^j}$  sampled by  $S_j$  is at most  $\frac{1}{2}$ . Hence by Chernoff bounds, the probability that the number of rows that exceed  $\frac{\hat{M}}{2^j}$  sampled by  $S_j$  is more than  $t = \mathcal{O}\left(\frac{\log n}{\alpha^3}\right)$  is  $\frac{1}{\text{poly}(nT)}$ . ◀

**Proof of Lemma 14.** Suppose that for each  $j \in [K]$ , level  $j$  consists of  $N_j$  rows and note that  $N_j \geq \mathcal{O}\left(\frac{(1+\alpha)^j \alpha^3}{\log n}\right)$  elements due to the dummy rows. Each element is sampled with some probability  $p_{L_j}$ , where  $L_j = \max\left(1, \log \frac{\alpha^2(1+\alpha)^j}{\log n}\right)$  and thus  $p_{L_j}(1+\alpha)^j > 1$  since  $p_{L_j} = \frac{1}{2^{L_j}}$ . Let  $\hat{N}_j$  be the number of items sampled in  $\tilde{S}_{L_j}$ . We have  $\mathbb{E}\left[2^{L_j} \cdot \hat{N}_j\right] = N_j$  and the second moment is at most  $N_j \cdot 2^{L_j} \leq \frac{\alpha^2}{\log n} (N_j)^2$ . Thus by Chernoff bounds with  $\mathcal{O}(\log n)$ -wise independence, we have that with high probability,

$$(1 - \mathcal{O}(\alpha))N_j \leq 2^{L_j} \cdot \hat{N}_j \leq (1 + \mathcal{O}(\alpha))N_j.$$

Each estimated row norm is a  $(1 + \alpha)$ -approximation to the actual row norm due to Lemma 13. Thus by Lemma 12, we have that  $F_G(\tilde{S}_j) \leq 2F_G(S_j)$  so that each of the  $\hat{N}_j$  rows will be detected by the threshold of COUNTSKETCH with the tail guarantee, i.e., Lemma 11. Moreover, we assume that a noisy row with  $(1 + \alpha)$ -approximation to the row norm of the original vector suffices to obtain a  $(1 + \varepsilon)$ -approximation to the contribution of the row. Therefore, the result then follows in an ideal scenario where  $G(\mathbf{v}) \in \left[\frac{\hat{M}}{2^j}, \frac{2\hat{M}}{2^j}\right)$  if and only if the corresponding row  $\mathbf{a}_i$  satisfies  $G(\mathbf{a}_i) \in \left[\frac{\hat{M}}{2^j}, \frac{2\hat{M}}{2^j}\right)$ . Unfortunately, this may not be true because  $G(\mathbf{a}_i)$  may lie near the boundary of the interval  $\left[\frac{\hat{M}}{2^j}, \frac{2\hat{M}}{2^j}\right)$  while the estimate  $G(\mathbf{v})$  has a value that does not lie within the interval. In this case,  $G(\mathbf{v})$  is used toward the estimation of some other level set.

Hence, our algorithm randomizes the boundaries of the level sets  $\left[\frac{4\gamma\hat{M}}{2^j}, \frac{8\gamma\hat{M}}{2^j}\right)$  by choosing  $\gamma \in [1/2, 1)$  uniformly at random. Since the threshold of COUNTSKETCH is  $\mathcal{O}\left(\frac{\alpha^3}{\log n}\right)$  then the probability that each row  $\mathbf{a}_i$  is misclassified over the choice of  $\gamma$  is at most  $\mathcal{O}(\varepsilon)$ . Moreover, if  $\mathbf{a}_i$  is misclassified, then its contribution can only be classified into level set  $j - 1$  or  $j + 1$ , inducing an incorrect multiplicative factor of at most two. Hence, the error due to the misclassification across all rows is at most  $\mathcal{O}(\varepsilon)$  fraction of  $F_G(S_j)$  in expectation. By Markov's inequality, this error is a most  $\varepsilon$ -fraction of  $F_G(S_j)$  with probability at least  $3/4$ . Then by taking the median across  $\mathcal{O}(\log(nT))$  independent instances, we obtain high probability of success. ◀

**Proof of Lemma 15.** Conditioned on the correctness of each of the estimates  $\widetilde{F}_G(\widetilde{S}_j)$ , which occurs with high probability by Lemma 14, the probability that the algorithm selects  $j \in [K]$  is  $\frac{\widetilde{F}_G(\widetilde{S}_j)}{\sum_{j \in [K]} \widetilde{F}_G(\widetilde{S}_j)}$ . Conditioned on the algorithm selecting  $j \in [K]$ , then either the algorithm will choose a dummy row, or it will choose a row uniformly at random from the rows  $\mathbf{v} \in X_j$ , where  $X_j$  is the set of heavy-hitters reported by  $H_j$  with  $L_2$  norm in  $\left[ \frac{8\gamma\widehat{M}}{(1+\alpha)^{j+1}}, \frac{8\gamma\widehat{M}}{(1+\alpha)^j} \right)$ . The latter event occurs with probability  $\frac{\widetilde{F}_G(\widetilde{S}_j)}{\widetilde{F}_G(\widetilde{S}_j)}$ . Due to the tail guarantee of COUNTSKETCH in Lemma 11, we have that each heavy hitter  $\mathbf{v} \in X_j$  corresponds to a vector  $\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)$  such that  $\|\mathbf{v} - \mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)\|_2 \leq \varepsilon \|\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)\|_2$ . Moreover, by Lemma 11, we have that  $\mathbb{E}[\mathbf{v}] = \mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)$ . Hence the probability that vector  $\mathbf{v}$  is selected is  $\frac{(1 \pm \mathcal{O}(\alpha))G(\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i))}{\widetilde{F}_G(\widetilde{S}_j)}$ . ◀

Putting things together, we have the guarantees of Theorem 5 for our  $G$ -sampler.

► **Theorem 5.** *Given an  $(\alpha, \varepsilon)$ -smooth gradient  $G$ , there exists an algorithm SAMPLER that outputs a noisy vector  $\mathbf{v}$  such that  $\|\mathbf{v} - \mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)\|_2 \leq \alpha \|\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)\|_2$  and  $\mathbb{E}[\mathbf{v}] = \mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)$  is  $(1 \pm \mathcal{O}(\varepsilon)) \frac{\|G(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i, \mathbf{a}_i)\|_2}{\sum_{j \in [n]} \|G(\langle \mathbf{a}_j, \mathbf{x} \rangle - b_j, \mathbf{a}_j)\|_2} + \frac{1}{\text{poly}(n)}$ . The algorithm uses  $d^2 \text{poly}(\log(nT), \frac{1}{\alpha})$  update time per arriving row and  $Td^2 \text{poly}(\log(nT), \frac{1}{\alpha})$  total bits of space.*

**Proof.** We define a class  $C_k$  of rows as the subset of rows of the input matrix  $\mathbf{A}$  such that  $2^k \leq \|\mathbf{a}_i\|_2 < 2^{k+1}$ . We first use the estimator algorithm in Theorem 9 to sample a class  $k$  of rows with probability  $\frac{\sum_{\mathbf{a}_i \in C_k} G(\langle \mathbf{a}_i, x \rangle - b_i, \mathbf{a}_i)}{\sum_{j \in [n]} G(\langle \mathbf{a}_j, x \rangle - b_j, \mathbf{a}_j)}$ . Once a class  $C_k$  is selected, then outputting a row from  $C_k$  under the correct distribution follows from Lemma 15. The space complexity follows from storing a  $d \times d$  matrix in each of the  $\mathcal{O}\left(\frac{\log^2(nT)}{\alpha^3}\right)$  buckets in COUNTSKETCH for threshold  $\mathcal{O}\left(\frac{\alpha^3}{\log(nT)}\right)$  and high probability of success. ◀

## B Missing Proofs from Section 3

**Proof of Lemma 17.** Let  $C > 0$  be a sufficiently large constant. For any  $t \in [T]$  and  $i \in [n]$ ,  $G(\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)) \geq \frac{1}{CT} \sum_{j \in [n]} G(\mathbf{a}_j(\langle \mathbf{a}_j, x \rangle - b_j))$  only if there exists a row in  $\mathbf{a}_i \circ b_i$  whose sensitivity is at least  $\frac{1}{CT}$ . However, we have explicitly stored all rows  $\mathbf{a}_i \circ b_i$  with sensitivity  $\Omega\left(\frac{1}{T}\right)$  and removed them from each  $G$ -sampler.

Thus, for all  $j \in [\beta]$  so that  $h(i) \neq j$  for any index  $i \in [n]$  such that  $G(\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)) \leq \frac{1}{CT} \sum_{k \in [n]} G(\mathbf{a}_k(\langle \mathbf{a}_k, x \rangle - b_k))$ , we have

$$\sum_{i: h(i)=j} G(\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)) \leq \frac{\log(Td)}{200T} \sum_{k \in [n]} G(\mathbf{a}_k(\langle \mathbf{a}_k, x \rangle - b_k)),$$

with probability at least  $1 - \frac{1}{\text{poly}(Td)}$  by Bernstein's inequality and a union bound over  $j \in [\beta]$ , where  $\beta = \Theta(T)$  is sufficiently large. Intuitively, by excluding the hash indices containing “heavy” matrices, the remaining hash indices contain only a small fraction of the mass with high probability.

We analyze the probability that any bucket containing rows with sensitivity less than  $\mathcal{O}\left(\frac{1}{T}\right)$  are sampled more than  $\Omega(T \log(Td))$  times, since we create  $\mathcal{O}(T \log(Td))$  separate  $G$ -samplers for each of these buckets. By a coupling argument and Chernoff bounds, the probability that any  $j \in [\beta]$  with  $\sum_{i: h(i)=j} G(\mathbf{a}_i(\langle \mathbf{a}_i, x \rangle - b_i)) \leq \frac{\log(Td)}{200T} \sum_{k \in [n]} G(\mathbf{a}_k(\langle \mathbf{a}_k, x \rangle - b_k))$

$b_k$ )) is sampled more than  $200 \log(Td)$  times is at most  $\frac{1}{\text{poly}(Td)}$  for any  $t \in [T]$ , provided there is no row with  $h(i) = j$  whose sensitivity is at least  $\frac{1}{CT}$ . Thus, the probability that some bucket  $j \in [\beta]$  is sampled more than  $200 \log(Td)$  times across  $T$  steps is at most  $\frac{1}{\text{poly}(Td)}$ .

In summary, we would like to maintain  $T$  separate instances of  $G$ -samplers for the heavy matrices and  $\Theta(\log(Td))$  separate instances of  $G$ -samplers for each hash index that does not contain a heavy matrix, but this creates a  $\Omega(T^2)$  space dependency. Instead, we explicitly store the heavy rows with sensitivity  $\Omega(\frac{1}{T})$ , removing them from the heavy matrices, and manually perform the sampling, rather than rely on the  $G$ -sampler subroutine. There can be at most  $\mathcal{O}(Td \log n)$  such rows, resulting in  $\mathcal{O}(Td^2 \log n)$  overall space for storing these rows explicitly. Since the resulting matrices are light by definition, we can maintain  $\Theta(\log(Td))$  separate instances of  $G$ -samplers for each of the  $\Theta(T)$  buckets, which results in  $\tilde{\mathcal{O}}(Td^2)$  space overall. With probability at least  $\frac{98}{100}$ , any hash index not containing a heavy matrix is sampled only once, so each time  $t \in [T]$  has access to a fresh  $G$ -sampler. ◀

**Derandomization of the algorithm.** To derandomize our algorithm, we first recall the following formulation of Nisan's pseudorandom generator.

► **Theorem 19** (Nisan's Pseudorandom Generator, [28]). *Let  $\mathcal{A}$  be an algorithm that uses  $S = \Omega(\log n)$  space and  $R$  random bits. Then there exists a pseudorandom generator for  $\mathcal{A}$  that succeeds with high probability and runs in  $\mathcal{O}(S \log R)$  bits.*

The goal of Nisan's PRG is to fool a small space tester by generating a number of pseudorandom bits in a read-once tape in place of a number of truly random bits. In the row-arrival model, the updates to each row  $\mathbf{a}_i$  of  $\mathbf{A} \in \mathbb{R}^{n \times d}$  arrive sequentially, so it suffices to use a read-once input tape. Thus a tester that is only allowed to  $S$  space cannot distinguish between the output of our algorithm using true randomness and pseudorandom bits generated by Nisan's PRG. Since our algorithm uses  $S = Td^2 \text{polylog}(Tnd)$  bits of space and  $R = \text{poly}(n, T, d)$  bits of randomness, then it can be randomized by Nisan's PRG while using  $Td^2 \text{polylog}(Tnd)$  total space.





# Finding the KT Partition of a Weighted Graph in Near-Linear Time

Simon Apers ✉

CNRS and IRIF, Paris, France

Paweł Gawrychowski ✉

Institute of Computer Science, University of Wrocław, Poland

Troy Lee ✉

Centre for Quantum Software and Information, University of Technology Sydney, Australia

---

## Abstract

In a breakthrough work, Kawarabayashi and Thorup (J. ACM'19) gave a near-linear time deterministic algorithm to compute the weight of a minimum cut in a simple graph  $G = (V, E)$ . A key component of this algorithm is finding the  $(1 + \varepsilon)$ -KT partition of  $G$ , the coarsest partition  $\{P_1, \dots, P_k\}$  of  $V$  such that for every non-trivial  $(1 + \varepsilon)$ -near minimum cut with sides  $\{S, \bar{S}\}$  it holds that  $P_i$  is contained in either  $S$  or  $\bar{S}$ , for  $i = 1, \dots, k$ . In this work we give a near-linear time randomized algorithm to find the  $(1 + \varepsilon)$ -KT partition of a *weighted* graph. Our algorithm is quite different from that of Kawarabayashi and Thorup and builds on Karger's framework of tree-respecting cuts (J. ACM'00).

We describe a number of applications of the algorithm. (i) The algorithm makes progress towards a more efficient algorithm for constructing the polygon representation of the set of near-minimum cuts in a graph. This is a generalization of the cactus representation, and was initially described by Benczúr (FOCS'95). (ii) We improve the time complexity of a recent quantum algorithm for minimum cut in a simple graph in the adjacency list model from  $\tilde{O}(n^{3/2})$  to  $\tilde{O}(\sqrt{mn})$ , when the graph has  $n$  vertices and  $m$  edges. (iii) We describe a new type of randomized algorithm for minimum cut in simple graphs with complexity  $\mathcal{O}(m + n \log^6 n)$ . For graphs that are not too sparse, this matches the complexity of the current best  $\mathcal{O}(m + n \log^2 n)$  algorithm which uses a different approach based on random contractions.

The key technical contribution of our work is the following. Given a weighted graph  $G$  with  $m$  edges and a spanning tree  $T$  of  $G$ , consider the graph  $H$  whose nodes are the edges of  $T$ , and where there is an edge between two nodes of  $H$  iff the corresponding 2-respecting cut of  $T$  is a non-trivial near-minimum cut of  $G$ . We give a  $\mathcal{O}(m \log^4 n)$  time deterministic algorithm to compute a spanning forest of  $H$ .

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis

**Keywords and phrases** Graph theory

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.32

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2111.01378> [1]

**Funding** *Paweł Gawrychowski:* Partially supported by the Bekker programme of the Polish National Agency for Academic Exchange (PPN/BEK/2020/1/00444).

*Troy Lee:* Supported in part by the Australian Research Council Grant No: DP200100950.



© Simon Apers, Paweł Gawrychowski, and Troy Lee;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 32; pp. 32:1–32:14



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Given a weighted and undirected graph  $G$  with  $n$  vertices and  $m$  edges<sup>1</sup>, the minimum cut problem is to find the minimum weight  $\lambda(G)$  of a set of edges whose removal disconnects  $G$ . When  $G$  is unweighted, this is simply the minimum number of edges whose removal disconnects  $G$ , also known as the edge connectivity of  $G$ . The minimum cut problem is a fundamental problem in theoretical computer science whose study goes back to at least the 1960s when the first polynomial time algorithm computing edge connectivity was given by Gomory and Hu [12]. In the current state-of-the-art, there are near-linear time randomized algorithms for the minimum cut problem in weighted graphs [9, 15, 21] and near-linear time *deterministic* algorithms in the case of simple graphs<sup>2</sup> [14, 18]. Very recently, Li [19] has given an almost-linear time (i.e. time  $\mathcal{O}(m^{1+o(1)})$ ) deterministic algorithm for weighted graphs as well.

The best known algorithms for weighted graphs all rely on a framework developed by Karger [15] which, for an input graph  $G$ , relies on finding  $\mathcal{O}(\log n)$  spanning trees of  $G$  such that with high probability one of these spanning trees will contain at most 2 edges from a minimum cut of  $G$ . In this case the cut is said to 2-respect the tree. A key insight of Karger is that, given a spanning tree  $T$  of  $G$ , the problem of finding a 2-respecting cut of  $T$  that has minimum weight in  $G$  can be solved deterministically in near-linear time, specifically time  $\mathcal{O}(m \log^2 n)$ . After standing for 20 years, the bound for this minimum-weight 2-respecting cut problem was recently improved by Gawrychowski, Mozes, and Weimann [9], who gave a deterministic  $\mathcal{O}(m \log n)$  time algorithm, and independently by Mukhopadhyay and Nanongkai [21] who gave a randomized algorithm with complexity  $\mathcal{O}(m \log n + n \log^4 n)$ .

The best algorithms in the case of a simple graph  $G$  rely on a quite different approach, pioneered by Kawarabayashi and Thorup [18]. This approach begins by finding the minimum degree  $d$  of a vertex in  $G$ . Then the question becomes if there is a non-trivial cut, i.e. a cut where both sides of the corresponding bipartition have cardinality at least 2, whose weight is less than  $d$ . This problem is solved by finding what we call the  $(1 + \varepsilon)$ -KT partition of the graph. Let  $\mathcal{B}_\varepsilon^{nt}(G)$  be the set of all bipartitions  $\{S, \bar{S}\}$  of the vertex set corresponding to non-trivial cuts whose weight is at most  $(1 + \varepsilon)\lambda(G)$ . The  $(1 + \varepsilon)$ -KT partition of  $G$  is the coarsest partition  $\{P_1, \dots, P_k\}$  of the vertex set such that for any  $\{S, \bar{S}\} \in \mathcal{B}_\varepsilon^{nt}(G)$  it holds that  $P_i$  is contained in either  $S$  or  $\bar{S}$ , for each  $i = 1, \dots, k$ . If one considers the multigraph  $G'$  formed from  $G$  by identifying vertices in the same set  $P_i$ , then  $G'$  preserves all non-trivial  $(1 + \varepsilon)$ -near minimum cuts of  $G$ . Kawarabayashi and Thorup further show that for any  $\varepsilon < 1$  the graph  $G'$  only has  $\tilde{\mathcal{O}}(n)$  edges. This bound crucially uses that the original graph is simple. The edge connectivity of  $G$  is thus the minimum of  $d$  and the edge connectivity of  $G'$ . One can use Gabow's deterministic  $\mathcal{O}(\lambda m' \log n)$  edge connectivity algorithm [8] for a multigraph with  $m'$  edges and edge connectivity  $\lambda$  to check in time  $\tilde{\mathcal{O}}(nd \log n) = \tilde{\mathcal{O}}(m)$  if the edge connectivity of  $G'$  is less than  $d$  and, if so, compute it. In the most technical part of their work, Kawarabayashi and Thorup give a deterministic algorithm to find the  $(1 + \varepsilon)$ -KT partition of a simple graph  $G$  in time  $\tilde{\mathcal{O}}(m)$ , giving an  $\tilde{\mathcal{O}}(m)$  time deterministic algorithm overall for edge connectivity. The key tool in their algorithm is the PageRank algorithm, which they use for finding low conductance cuts in the graph.

<sup>1</sup> Throughout this paper we will use  $n$  and  $m$  to denote the number of vertices and edges of the input graph.

<sup>2</sup> A simple graph is an unweighted graph with no self loops and at most one edge between any pair of vertices.

The KT partition has proven to be a very useful concept. Rubinfeld, Schramm, and Weinberg [23] also go through the  $(1+\varepsilon)$ -KT partition to give a near-optimal  $\tilde{O}(n)$  randomized query algorithm determining the edge connectivity of a simple graph in the *cut query* model. In the cut query model one can query a subset of the vertices  $S$  and receive in return the number of edges with exactly one endpoint in  $S$ . En route to their result, [23] also improved the bound on the number of inter-component edges in the  $(1+\varepsilon)$ -KT partition of a simple graph to  $\mathcal{O}(n)$ , for any  $\varepsilon < 1$ . In the case  $\varepsilon = 0$  this was independently done by Lo, Schmidt, and Thorup [20]. The KT partition approach is also used in the current best *randomized* algorithm for edge connectivity, which runs in time  $\mathcal{O}(\min\{m + n \log^2 n, m \log n\})$  [10].<sup>3</sup>

## 1.1 Main result

In this work we give the first near-linear time randomized algorithm to find the  $(1+\varepsilon)$ -KT partition of a *weighted* graph, for  $0 \leq \varepsilon \leq 1/16$ . An interesting aspect of our algorithm is that it uses Karger's 2-respecting cut framework to find the  $(1+\varepsilon)$ -KT partition, thereby combining the aforementioned major lines of work on the minimum cut problem.

We describe the result in more detail. Let  $G = (V, E, w)$  be a weighted graph, where  $E$  is the set of edges and  $w : E \rightarrow \mathbb{R}_+$  assigns a positive weight to each edge. For a set  $S \subseteq V$  let  $\Delta_G(S)$  be the set of all edges of  $G$  with exactly one endpoint in  $S$ . A *cut* of  $G$  is a set of edges of the form  $\Delta_G(S)$  for some  $\emptyset \neq S \subsetneq V$ . We call  $S$  and  $\bar{S}$  the *shores* of the cut. Let  $w(\Delta_G(S)) = \sum_{e \in \Delta_G(S)} w(e)$ . We use  $\lambda(G) = \min_{\emptyset \neq S \subsetneq V} w(\Delta_G(S))$  for the minimum weight of a cut in  $G$ .

We will be interested in partitions of  $V$  and the partial order on partitions induced by *refinement*. For two partitions  $\mathcal{X}, \mathcal{Y}$  of  $V$  we say that  $\mathcal{X} \preceq \mathcal{Y}$  iff for every  $X \in \mathcal{X}$  there is a  $Y \in \mathcal{Y}$  with  $X \subseteq Y$ . In this case we say  $\mathcal{X}$  is a *refinement* of  $\mathcal{Y}$ . The *meet* of two partitions  $\mathcal{X}$  and  $\mathcal{Y}$ , denoted  $\mathcal{X} \wedge \mathcal{Y}$ , is the partition  $\mathcal{Z}$  such that  $\mathcal{Z} \preceq \mathcal{X}, \mathcal{Z} \preceq \mathcal{Y}$  and for any other partition  $\mathcal{W}$  satisfying these two conditions  $\mathcal{W} \preceq \mathcal{Z}$ . In other words,  $\mathcal{X} \wedge \mathcal{Y}$  is the greatest lower bound on  $\mathcal{X}$  and  $\mathcal{Y}$  under  $\preceq$ . Explicitly,  $\mathcal{X} \wedge \mathcal{Y}$  is the partition consisting of all non-empty pairwise intersections between sets from  $\mathcal{X}$  and  $\mathcal{Y}$ . For a set of partitions  $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_K\}$  we write  $\bigwedge \mathcal{D} = \mathcal{D}_1 \wedge \dots \wedge \mathcal{D}_K$ .

For our applications we need to consider not only minimum cuts, but also near-minimum cuts. For  $\varepsilon \geq 0$ , let  $\mathcal{B}_\varepsilon(G) = \{\{S, \bar{S}\} : w(\Delta_G(S)) \leq (1+\varepsilon)\lambda(G)\}$  be the set of all bipartitions of  $V$  corresponding to  $(1+\varepsilon)$ -near minimum cuts. Let  $\mathcal{B}_\varepsilon^{nt}(G) \subseteq \mathcal{B}_\varepsilon(G)$  be the set of all the non-trivial cuts in  $\mathcal{B}_\varepsilon(G)$ . The  $(1+\varepsilon)$ -KT partition of  $G$  is exactly  $\bigwedge \mathcal{B}_\varepsilon^{nt}(G)$ .

Both  $\bigwedge \mathcal{B}_\varepsilon(G)$  and  $\bigwedge \mathcal{B}_\varepsilon^{nt}(G)$  are important sets for understanding the structure of (near)-minimum cuts in a graph. Consider first  $\bigwedge \mathcal{B}_0(G)$ , the meet of the set of all bipartitions corresponding to minimum cuts. This set arises in the cactus decomposition of  $G$  [7], a compact representation of all minimum cuts of  $G$ . A cactus is a connected multigraph where every edge appears in exactly one cycle. The edge connectivity of a cactus is 2 and the minimum cuts are obtained by removing any two edges from the same cycle. A *cactus decomposition* of a graph  $G$  is a cactus  $H$  on  $\mathcal{O}(n)$  vertices and a mapping  $\phi : V(G) \rightarrow V(H)$  such that  $\Delta_G(\phi^{-1}(S))$  is a mincut of  $G$  iff  $\Delta_H(S)$  is a mincut of  $H$ . The mapping  $\phi$  does not have to be injective, so multiple vertices of  $G$  can map to the same vertex of  $H$ . In this case, however, the cactus decomposition property means that all vertices in  $\phi^{-1}(\{v\})$  must be on the same side of every minimum cut of  $G$ , for every  $v \in V(H)$ . Thus as  $v$  ranges over  $V(H)$

<sup>3</sup> The bound quoted in [10] is  $\mathcal{O}(m + n \log^3 n)$  but the improvement to Karger's algorithm by [9] reduces this to  $\mathcal{O}(m + n \log^2 n)$ .

the sets  $\phi^{-1}(\{v\})$  give the elements of  $\bigwedge \mathcal{B}_0(G)$  (note that  $\phi^{-1}(\{v\})$  can also be empty). A cactus decomposition of a weighted graph can be constructed by a randomized algorithm in near-linear time [16], thus this also gives a near-linear time randomized algorithm to compute  $\bigwedge \mathcal{B}_0(G)$ .

Lo, Schmidt, and Thorup [20] give a version of the cactus decomposition that only represents the non-trivial minimum cuts. In fact, they give a deterministic  $\mathcal{O}(n)$  time algorithm that converts a standard cactus into one representing the non-trivial minimum cuts. Combining this with the near-linear time algorithm to compute a cactus decomposition, this gives a near-linear time randomized algorithm to compute  $\bigwedge \mathcal{B}_0^{nt}(G)$  as well.

The situation changes once we go to near-minimum cuts, which can no longer be represented by a cactus, but require the deformable polygon representation from [3–5]. This construction is fairly intricate, and the best known randomized algorithm to construct a deformable polygon representation of the  $(1+\varepsilon)$ -near mincuts of a graph builds on the Karger-Stein algorithm and takes time  $\mathcal{O}(n^{2(1+\varepsilon)})$  [3, Section 6.3]. A prerequisite to constructing a deformable polygon representation is being able to compute  $\bigwedge \mathcal{B}_\varepsilon(G)$  as, analogously to the case of a cactus, these sets will be the “atoms” that label regions of the polygons.

Our main result in this work is to give a randomized algorithm to compute  $\bigwedge \mathcal{B}_\varepsilon(G)$  and  $\bigwedge \mathcal{B}_\varepsilon^{nt}(G)$  in time  $\mathcal{O}(m \log^5 n)$ .

► **Theorem 1.** *Let  $G = (V, E, w)$  be a graph with  $n$  vertices and  $m$  edges. For  $0 \leq \varepsilon \leq 1/16$  let  $\mathcal{B}_\varepsilon = \{\{S, \bar{S}\} : w(\Delta(S)) \leq (1+\varepsilon)\lambda(G)\}$  and  $\mathcal{B}_\varepsilon^{nt} \subseteq \mathcal{B}_\varepsilon$  be the subset of  $\mathcal{B}_\varepsilon$  containing only non-trivial cuts. Both  $\bigwedge \mathcal{B}_\varepsilon$  and  $\bigwedge \mathcal{B}_\varepsilon^{nt}$  can be computed with high probability by a randomized algorithm with running time  $\mathcal{O}(m \log^5 n)$ .*

In the rest of this paper, we focus on computing  $\bigwedge \mathcal{B}_\varepsilon^{nt}$ . It is easy to construct  $\bigwedge \mathcal{B}_\varepsilon$  from  $\bigwedge \mathcal{B}_\varepsilon^{nt}$  deterministically in  $\mathcal{O}(n)$  time (see full version [1]).

## 1.2 Applications

By building on our KT partition algorithm, we make progress on a number of problems.

1. The polygon representation is a compact representation of the set of near-minimum cuts of a weighted graph, originally described by Benczúr [3, 4] and Benczúr-Goemans [5]. It extends the cactus representation [7], which only works for the set of exact minimum cuts, and has played a key role in recent breakthroughs on the traveling salesperson problem [11, 17]. For a general weighted graph the polygon representation has size  $\mathcal{O}(n^2)$ , and Benczúr has given a randomized algorithm to construct a polygon representation of the  $(1+\varepsilon)$ -near mincuts of a graph in time  $\mathcal{O}(n^{2(1+\varepsilon)})$  [3, Section 6.3] by building on the Karger-Stein algorithm. It is an open question whether we can construct a polygon representation in time  $\tilde{\mathcal{O}}(n^2)$  for  $\varepsilon > 0$ . In his thesis [3, pg. 126], Benczúr says, “It already seems hard to directly identify the system of atoms within the  $\tilde{\mathcal{O}}(n^2)$  time bound,” where the system of atoms is defined analogously to the  $(1+\varepsilon)$ -KT partition but for the set of all  $(1+\varepsilon)$ -near minimum cuts, not just the non-trivial ones. One can easily construct the set of atoms from a  $(1+\varepsilon)$ -KT partition, thus our KT partition algorithm gives a  $\tilde{\mathcal{O}}(m)$  time algorithm for this task as well, making progress on this open question.
2. The  $(1+\varepsilon)$ -KT partition of a weighted graph is exactly what is needed to give an optimal *quantum* algorithm for minimum cut: Apers and Lee [2] showed that the quantum query and time complexity of minimum cut in the adjacency matrix model is  $\tilde{\Theta}(n^{3/2}\sqrt{\tau})$  for a weighted graph where the ratio of the largest to smallest edge weights is  $\tau$ , with the algorithm proceeding by finding a  $(1+\varepsilon)$ -KT partition.

In the case where the graph is instead represented as an adjacency list, they gave an algorithm with query complexity  $\tilde{O}(\sqrt{mn\tau})$  but whose running time is larger at  $\tilde{O}(\sqrt{mn\tau} + n^{3/2})$ . The bottleneck in the time complexity is the time taken to find a  $(1 + \varepsilon)$ -KT partition of a weighted graph with  $\tilde{O}(n)$  edges. Using the near-linear time randomized algorithm we give to find a  $(1 + \varepsilon)$ -KT partition here improves the time complexity of this algorithm to  $\tilde{O}(\sqrt{mn\tau})$ , matching the query complexity.

Both quantum algorithms also used the following observation [2, Lemma 2]: if in a weighted graph  $G$  the ratio of the largest edge weight to the smallest is  $\tau$ , then the total weight of inter-component edges in a  $(1 + \varepsilon)$ -KT partition of  $G$  for  $\varepsilon < 1$  is  $\mathcal{O}(\tau n)$ , which can be tight.

3. The best randomized algorithm to compute the edge connectivity of a simple graph is the 2-out contraction approach of Ghaffari, Nowicki, and Thorup [10], which has running time  $\mathcal{O}(\min\{m + n \log^2 n, m \log n\})$ . Using our algorithm to find a  $(1 + \varepsilon)$ -KT partition in a weighted graph we can follow Karger's 2-respecting tree approach to compute the edge connectivity of a simple graph in time  $\mathcal{O}(m + n \log^6 n)$ , thus also achieving the optimal bound on graphs that are not too sparse.

A detailed treatment of these applications is deferred to the full version [1]. Apart from these examples, we are hopeful that our near-optimal randomized algorithm for finding the KT partition of a weighted graph will find further applications.

## 2 KT partition algorithm

We now give a more detailed treatment of our algorithm to compute the KT partition  $\bigwedge \mathcal{B}_\varepsilon^{nt}$ . The first obstacle we face is that the number of near-minimum cuts in  $G$  can be  $\Omega(n^2)$ , so we cannot afford to consider all of them. An idea to get around this is to try the following:

1. Efficiently find a “small” subset  $\mathcal{B}' \subseteq \mathcal{B}_\varepsilon^{nt}$  such that  $\bigwedge \mathcal{B}' = \bigwedge \mathcal{B}_\varepsilon^{nt}$ . We call such a subset a *generating set*.

A greedy argument shows that such a subset  $\mathcal{B}'$  exists of size at most  $n - 1$ . We initialize  $\mathcal{B}' = \{\{S, \bar{S}\}\}$  for some element  $\{S, \bar{S}\}$  in  $\mathcal{B}_\varepsilon^{nt}$ . We then iterate through the elements  $\{T, \bar{T}\}$  of  $\mathcal{B}_\varepsilon^{nt}$  and add it to  $\mathcal{B}'$  iff  $\bigwedge \mathcal{B}' \cup \{T, \bar{T}\} \neq \bigwedge \mathcal{B}'$ . Each bipartition added to  $\mathcal{B}'$  increases the number of elements in  $\bigwedge \mathcal{B}'$  by at least 1. As this size can be at most  $n$ , and begins with size 2 the total number of sets at termination is at most  $n - 1$ . This shows that a small generating set exists, but there still remains the problem of finding such a generating set *efficiently*.

Assuming we get past the first obstacle, there remains a second obstacle. The most straightforward algorithm to compute the meet of  $k$  partitions of a set of size  $n$  takes time  $\Theta(kn \log n)$ , which is again too slow if  $k = \Theta(n)$ . Thus we will also need to

2. Exploit the structure of  $\mathcal{B}'$  to compute  $\bigwedge \mathcal{B}'$  efficiently.

Apers and Lee [2] give an approach to accomplish (1) and (2) following Karger's framework of tree respecting cuts. Karger shows that in near-linear time one can compute a set of  $K \in \mathcal{O}(\log n)$  spanning trees  $T_1, \dots, T_K$  of  $G$  such that every  $(1 + \varepsilon)$ -near minimum cut of  $G$  2-respects at least one of these trees. Let  $\mathcal{B}_i \subseteq \mathcal{B}_\varepsilon^{nt}$  be the bipartitions corresponding to non-trivial near-minimum cuts that 2-respect  $T_i$ . To compute  $\bigwedge \mathcal{B}_\varepsilon^{nt}$  it suffices to compute  $\mathcal{C}_i = \bigwedge \mathcal{B}_i$  for each  $i = 1, \dots, K$  and then compute  $\bigwedge_{i=1}^K \mathcal{C}_i$ . The latter can be done in time  $\mathcal{O}(n \log^2 n)$  by the aforementioned algorithm. This leaves the problem of computing  $\bigwedge \mathcal{B}_i$ .

A key observation from [2] gives a generating set  $\mathcal{B}'_i$  for  $\mathcal{B}_i$  of size  $\mathcal{O}(n)$ . It is constructed as follows. First we add bipartitions to  $\mathcal{B}_i$  that correspond to near-minimum cuts that *1-respect*  $T_i$ . This is a set of size  $\mathcal{O}(n)$ , and Karger has shown that all near-minimum cuts that 1-respect a tree can be found in time  $\mathcal{O}(m)$ .

Next, we focus on the cuts that strictly *2-respect*  $T_i$ . There can be  $\mathcal{O}(n^2)$  such cuts. To handle them one creates a graph  $H$  whose nodes are the *edges* of  $T_i$  and where there is an edge between nodes  $e$  and  $f$  iff the 2-respecting cut of  $T_i$  defined by  $\{e, f\}$  is a near-minimum cut in  $\mathcal{B}_i$ . Let  $F$  be a spanning forest of  $H$ . We then add to  $\mathcal{B}'_i$  the 2-respecting cuts indexed by the edges in  $E(F)$ . The resulting set  $\mathcal{B}'_i$  has size  $\mathcal{O}(n)$  and it follows from [2, Lemma 29] that the resulting  $\mathcal{B}'_i$  is a generating set for  $\mathcal{B}_i$ .

Apers and Lee give a *quantum* algorithm to find a spanning forest of  $H$  with running time  $\tilde{\mathcal{O}}(n^{3/2})$ . They then give a randomized algorithm to compute  $\bigwedge \mathcal{B}'_i$  in time  $\tilde{\mathcal{O}}(n)$ . As our main technical contribution, we give a deterministic algorithm to find a spanning forest of  $H$  in time  $\mathcal{O}(m \log^4 n)$ . We also replace the randomization used in the algorithm to compute  $\bigwedge \mathcal{B}'_i$  with an appropriate data structure to give an  $\tilde{\mathcal{O}}(n)$  deterministic algorithm to compute the meet. The details of this last procedure are deferred to the full version [1].

### 3 Spanning tree of near-minimum 2-respecting cuts in near-linear time

By the preceding discussion, we reduced the problem of constructing the KT partition to that of finding a spanning forest in a new graph  $H$ . This graph is derived from the original graph  $G$  and a given spanning forest  $T$  of  $G$ . Here we describe a deterministic near-linear time algorithm for this task, which is our main technical contribution.

Before describing the spanning forest algorithm, it is interesting to compare the problem of finding a spanning forest of  $H$  with the original problem solved by Karger of finding a minimum-weight 2-respecting cut of  $T$ . To find a spanning forest of  $H$  we potentially have to find  $\Omega(n)$  many  $(1 + \varepsilon)$ -near minimum cuts, which we accomplish with only an additional logarithmic overhead in the running time. The first insight to how this might be possible is to note that Karger's original algorithm to find the minimum weight 2-respecting cut actually does something stronger than needed. Let  $\text{cost}(e, f)$  be the weight of the 2-respecting cut of  $T$  defined by  $\{e, f\}$ . For *every* edge  $e$  of  $T$  Karger's algorithm attempts to find an  $f^* \in \arg \min_f \text{cost}(e, f)$ . It does not always succeed in this task, but if the candidate  $f'$  returned for edge  $e$  is not such a minimizer, then for  $f^* \in \arg \min_f \text{cost}(e, f)$  it must be the case that the candidate  $g$  returned for  $f^*$  satisfies  $\text{cost}(f^*, g) \leq \text{cost}(e, f^*)$ . In this way, the algorithm still succeeds to find a minimum weight 2-respecting cut in the end.

In contrast, we give an algorithm that for every edge  $e$  of  $T$  actually finds

$$f^* \in \arg \min_f \{ \text{cost}(e, f) : \{e, f\} \text{ defines a non-trivial cut} \} .$$

We then show that this suffices to implement a round of Borůvka's spanning forest algorithm [22] on  $H$  in near-linear time. Borůvka's spanning forest algorithm consists of  $\log n$  rounds and maintains the invariant of having a partition  $\{S_1, \dots, S_k\}$  of the vertex set and a spanning tree for each set  $S_i$ . The algorithm terminates when there is no outgoing edge from any set of the partition, at which point the collection of spanning trees for the sets of the partition is a spanning forest of  $H$ . The sets of the partition are initialized to be individual nodes of  $H$ .

In each round of Borůvka's algorithm the goal is to find an outgoing edge from each set  $S_i$  of the partition which is not already a connected component. Consider a node  $e$  of  $H$  with  $e \in S_i$ . We can find the best partner  $f$  for  $e$  and check if  $\{e, f\}$  indeed gives rise to a non-trivial  $(1 + \varepsilon)$ -near minimum cut and so is an edge of  $H$ . The problem is that  $f$



could also be in  $S_i$  in which case the edge  $\{e, f\}$  is not an outgoing edge of  $S_i$  as desired. To handle this, we maintain a data structure that allows us to find both the best partner  $f$  for  $e$ , but also the best partner  $f'$  for  $e$  that lies in a different set of the partition from  $f$ . We call this operation a *categorical top two* query. If there actually is an edge of  $H$  with one endpoint  $e$  and the other endpoint outside of  $S_i$  then either  $\{e, f\}$  or  $\{e, f'\}$  will be such an edge. Following the approach of [9] to the minimum-weight 2-respecting cut problem, combined with an efficient data structure for handling categorical top two queries, we are able to do this for all nodes  $e$  of  $H$  in near-linear time, which allows us to implement a round of Borůvka's algorithm in near-linear time.

### 3.1 Spanning tree of near-minimum 2-respecting cuts in near-linear time

We give a more precise description of the algorithm after settling on some notation. Let  $G = (V, E, w)$  be a weighted undirected graph. We will assume throughout that  $G$  is connected, and in particular that  $m \geq n - 1$ , as the KT partition of a disconnected graph can be easily determined from its connected components. Let  $T$  be a spanning tree of  $G$ . We will choose an  $r \in V$  with degree 1 in  $T$  to be the root of  $T$ . We view  $T$  as a directed graph with all edges directed away from  $r$ . With some abuse of notation, we will also use  $T$  to refer to this directed version. If we remove any edge  $e \in E(T)$  from  $T$  then  $T$  becomes disconnected into two components. We use  $e^\downarrow \subseteq V$  to denote the set of vertices in the component not containing the root, and  $T_e \subseteq E(T)$  to denote the set of edges in the subtree rooted at the head of  $e$ , i.e. the edges in the subgraph of  $T$  induced by  $e^\downarrow$ . We further use the shorthand  $\text{cost}(e) = w(\Delta(e^\downarrow))$  for the weight of the cut with shore  $e^\downarrow$ .

Two edges  $e, f \in E(T)$  define a unique cut in  $G$  which we denote by  $\text{cut}_T(e, f)$  (or  $\text{cut}(e, f)$  if it is clear from the context which  $T$  we are referring to). The cut depends on the relationship between  $e$  and  $f$ . If  $e \in T_f$  or  $f \in T_e$  then we say that  $e$  and  $f$  are *descendant* edges. Without loss of generality, say that  $f \in T_e$ . Then the cut defined by  $e$  and  $f$  is  $\text{cut}(e, f) = \Delta(e^\downarrow \setminus f^\downarrow)$ . If  $e$  and  $f$  are not descendant edges, then we say they are *independent*. For independent edges we see that  $\text{cut}(e, f) = \Delta(e^\downarrow \cup f^\downarrow)$ . In both cases we use  $\text{cost}(e, f)$  to denote the weight of the corresponding cut.

In a KT partition we are only interested in non-trivial cuts. We first state the following simple claim that characterizes when  $\text{cut}(e, f)$  is trivial, the proof of which is in the full version [1].

► **Proposition 2.** *Let  $G = (V, E, w)$  be a connected graph with  $n$  vertices and let  $T$  be a spanning tree of  $G$  with root  $r$ . For  $e, f \in E(T)$  if  $\text{cut}(e, f)$  is trivial then*

1. *If  $e, f$  are independent then they must be the unique edges incident to the root.*
2. *If  $e, f$  are descendant then there is a vertex  $v \in V$  such that  $e$  is the edge incoming to  $v$  and  $f$  is the unique edge outgoing from  $v$ , or vice versa.*

By choosing a root  $r$  for  $T$  that has degree 1 we avoid the case of item 1 of Proposition 2. Thus we only have to worry about trivial cuts when  $e, f$  are descendant.

With that out of the way, we now turn to our main theorem, which is the key routine in our  $(1 + \varepsilon)$ -KT partition algorithm.

► **Theorem 3.** *Let  $G = (V, E, w)$  be a connected graph with  $n$  vertices and  $m$  edges and let  $T$  be a spanning tree of  $G$ . For a given parameter  $\beta$ , define the graph  $H$ , with  $V(H) = E(T)$  and  $E(H) = \{\{e, f\} \in E(T)^{(2)} : \text{cost}(e, f) \leq \beta \text{ and } \text{cut}(e, f) \text{ non-trivial}\}$ . There is a deterministic algorithm that given adjacency list access to  $G$  and  $T$  outputs a spanning forest of  $H$  in  $\mathcal{O}(m \log^4 n)$  time.*



We prove Theorem 3 by following Borůvka's algorithm to find a spanning forest of  $H$ . Throughout the algorithm we maintain a subgraph  $F$  of  $H$  that is a forest, initialized to be the empty graph on vertex set  $V(H) = E(T)$ . At the end of the algorithm,  $F$  will be a spanning forest of  $H$ . The algorithm proceeds in rounds. In each round, for every tree in the forest, we find an edge connecting it to another tree in the forest, if such an edge exists. If  $H$  has  $k$  connected components, then in each round the number of trees in  $F$  minus  $k$  goes down by at least a factor of 2, and so the algorithm terminates in  $\mathcal{O}(\log n)$  rounds.

The main work is implementing a round of Borůvka's algorithm. We will think of the nodes of  $F$  as having colors, where nodes in the same tree of the forest have the same color, and nodes in distinct trees have distinct colors. The goal of a single round is to find, for each color  $c$ , a pair of edges  $e, f \in T$  such that  $c = \text{color}(e) \neq \text{color}(f)$  and  $\{e, f\} \in E(H)$ , or detect that there is no such pair with these properties, in which case the nodes colored  $c$  in  $F$  already form a connected component of  $H$ . As we need to refer to such pairs often we make the following definition.

► **Definition 4** (partner). *Let  $T$  and  $H$  be as in Theorem 3. Given an assignment of colors to the edges of  $T$  we say that  $f$  is a partner for  $e$  if  $\{e, f\} \in E(H)$  and  $\text{color}(e) \neq \text{color}(f)$ .*

We will actually do something stronger than what is required to implement a round of Borůvka's algorithm, which we encapsulate in the following code header.

---

■ **Algorithm 1** RoundEdges.

---

**Input:** Adjacency list access to  $G$ , a spanning tree  $T$  of  $G$ , a parameter  $\beta$ , and an assignment of colors to each  $e \in E(T)$ .

**Output:** For every  $e \in E(T)$  output a partner  $f \in E(T)$ , or report that no partner for  $e$  exists.

---

The implementation of RoundEdges is our main technical contribution. Let us first see how to use RoundEdges to find a spanning forest of  $H$ .

► **Lemma 5.** *Let  $G$ ,  $T$  and  $H$  be as in Theorem 3. There is a deterministic algorithm that makes  $\mathcal{O}(\log n)$  calls to RoundEdges and in  $\mathcal{O}(n \log n)$  additional time outputs a spanning forest of  $H$ .*

**Proof.** We construct a spanning forest of  $H$  by maintaining a collection of trees  $F$  that will be updated in rounds by Borůvka's algorithm until it becomes a spanning forest. We initialize  $F = (E(T), \emptyset)$  and give all  $e \in E(T)$  distinct colors. We maintain the invariants that  $F$  is a forest and that nodes in the same tree have the same color and those in different trees have distinct colors.

Consider a generic round where  $F$  contains  $q$  trees. We call RoundEdges with the current color assignment. For every  $e$  which has one we obtain a partner  $f$  such that  $\{e, f\} \in E(H)$  and  $\text{color}(e) \neq \text{color}(f)$ . For each color class  $c$  we select one  $e$  with  $\text{color}(e) = c$  which has a returned partner (if it exists) and let  $X$  be the set of selected edges. We then find a maximal subset of edges  $X' \subseteq X$  that do not create a cycle among the color classes by computing a spanning forest of the graph whose supervertices are given by the color classes and edges given by  $X$ . We add the edges in  $X'$  to  $E(F)$ . Finally we merge the color classes of the connected components in  $F$  by appropriately updating the color assignments, and we pass the updated forest and color assignments to the next round of the algorithm. Each of the steps in a single round can be executed in  $\mathcal{O}(n)$  time.

We have that  $|X'| \geq (q - \text{cc}(H))/2$  where  $\text{cc}(H)$  is the number of connected components of  $H$ . Each edge from  $X'$  added to  $F$  decreases the number of trees in  $F$  by one. Thus  $q - \text{cc}(H)$  decreases by at least a factor of 2 in each round and the algorithm terminates after  $\mathcal{O}(\log n)$  rounds. The time spent outside of the calls to **RoundEdges** is  $\mathcal{O}(n)$  for each of the  $\mathcal{O}(\log n)$  rounds. This is  $\mathcal{O}(n \log n)$  overall.  $\blacktriangleleft$

If a node  $e$  has a partner  $f$ , then  $\{e, f\}$  can either be a pair of descendant or independent edges. To implement **RoundEdges** we will separately handle these cases, as described in the next two subsections.

### 3.2 Descendant edges

We follow the approach from [9] originally designed to find a single pair  $\{e, f\}$  of descendant edges that minimizes  $\text{cost}(e, f)$  over all  $e, f \in E(T)$  in  $\mathcal{O}(m \log n)$  time. Their approach actually does something stronger (as does Karger's original algorithm): for every  $e \in E(T)$  it finds the best match in the subtree  $T_e$ , i.e., it returns an edge  $f^* \in \arg \min\{\text{cost}(e, f) \mid f \in T_e\}$ . In order to implement the descendant edge part of **RoundEdges** we have three additional complications to handle:

1. The edge  $f^*$  might have the same color as  $e$ .
2. The resulting cut  $(e, f^*)$  might be a trivial cut.
3. Edge  $e$  may have no partner in  $T_e$  but still have a partner  $f$  such that  $e \in T_f$ . This partnership may not be discovered when we are looking for partners of  $f$  if there is another  $g \in T_f$  with  $\text{cost}(f, g) \leq \text{cost}(e, f)$ .

Item 1 can be easily solved by, in addition to finding  $f^*$ , also finding  $g^* \in \arg \min\{\text{cost}(e, f) \mid f \in E(T_e), \text{color}(f) \neq \text{color}(f^*)\}$ . Phrasing things in this way, rather than simply looking for the edge  $h$  with color different from  $e$  which minimizes  $\text{cost}(e, h)$ , helps to limit the dependence of the query on  $e$  and thus reduce the query time. If there is an  $f \in T_e$  with  $\text{color}(f) \neq \text{color}(e)$  and  $\text{cost}(e, f) \leq \beta$  then at least one of  $f^*, g^*$  will satisfy this too.

For item 2, we use the result of Proposition 2 that descendant edges that give rise to trivial cuts have a very constrained structure. This allows us to avoid trivial cuts when looking for a partner of  $e$ .

Item 3 is relatively subtle and does not arise in the minimum weight 2-respecting cut problem. To explain the issue we have to first say something about the high level structure of our implementation of **RoundEdges**. We will perform an Euler tour of  $T$  and, when the tour visits edge  $e$  for the first time, we will look for a partner  $f$  for  $e$  in  $T_e$ . The issue is the following, which we explain in the context of the very first round of Borůvka's algorithm so we do not have to worry about nodes having different colors. Suppose that in the graph  $H$  the only edge incident to node  $e$  is a node  $f$  with  $e \in T_f$ . Thus in the execution of **RoundEdges** we want to find  $f$  as a partner of  $e$ . When the Euler tour is at  $e$  we will not find any suitable partner for  $e$ , as there is none in  $T_e$ . We would like to identify  $f$  as a partner for  $e$  when the Euler tour visits  $f$  for the first time. However, if there is a  $g \in T_f$  with  $\text{cost}(f, g) < \text{cost}(f, e)$  then the algorithm will return  $g$  as a partner of  $f$  rather than  $e$ . To handle this we will actually make two passes over  $T$ . In the first pass, when we visit edge  $e$  for the first time we look for a partner  $f$  in  $T_e$ . In the second pass, we handle the case where the partner of  $e$  might be an ancestor of  $e$ . To do this we need to de-activate nodes. When the Euler tour visits  $f$  for the first time, we first find the lowest cost partner for  $f$  in  $T_f$ . We then de-activate this node, and again find the best active partner for  $f$  in  $T_f$ . Repeating this process, we will eventually find  $e$  if  $\{e, f\}$  is indeed an edge of  $H$  and  $e, f$  have different colors.

## 32:10 Finding the KT Partition of a Weighted Graph in Near-Linear Time

Now we turn to more specific implementation details. A key idea in [9] is that we can do an Euler tour of  $T$  while maintaining a data structure such that when we first visit an edge  $e$  we can easily look up  $\text{cost}(e, f)$  for any  $f \in T_e$ . The way this is maintained can be best understood by noting that for  $f \in T_e$ :

$$\begin{aligned} \text{cost}(e, f) &= w(\Delta(e^\perp \setminus f^\perp)) \\ &= w(e^\perp \setminus f^\perp, (e^\perp)^c) + w(e^\perp \setminus f^\perp, f^\perp) \\ &= \text{cost}(e) + \underbrace{\text{cost}(f) - 2w(f^\perp, (e^\perp)^c)}_{\text{score}_e(f)}, \end{aligned} \quad (1)$$

where for convenience we defined  $\text{score}_e(f) = \text{cost}(f) - 2w(f^\perp, (e^\perp)^c)$ , where the superscript  $c$  denotes taking the complement.

We begin the algorithm by computing  $\text{cost}(e)$  for every  $e \in E(T)$ , which can be done in  $\mathcal{O}(m)$  time [15]. We then do an Euler of  $T$  while maintaining a data structure such that, when we are considering  $e \in E(T)$ , for every  $f \in T_e$  the value of the data structure at location  $f$  is  $\text{cost}(e, f)$ . For  $f \notin T_e$  this will not in general be the case.

As can be seen from Equation (1), the key to maintaining this data structure is how to update the values  $w(f^\perp, (e^\perp)^c)$  when we descend edge  $e$ . Consider the case where we are currently at edge  $e' = (z, x)$  and move to a descending edge  $e = (x, y)$ . For two vertices  $u, v$  let  $p(u, v)$  be the set of edges on the path from  $u$  to  $v$  in  $T$ , and let  $\text{lca}(u, v)$  be their lowest common ancestor in  $T$ . For  $f \in T_e$  we see that

$$w(f^\perp, (e^\perp)^c) = w(f^\perp, (e')^c) + \sum_{\substack{\{u, v\} \in E \\ f \in p(u, v), \text{lca}(u, v) = x}} w(\{u, v\}) . \quad (2)$$

By its definition in (1) we can compute  $\text{score}_e(f)$  from  $\text{score}_{e'}(f)$  by *subtracting*  $2w(\{u, v\})$  from for every  $\{u, v\} \in E$  such that  $f \in p(u, v)$  and  $\text{lca}(u, v) = x$ . We implement this step for all  $f$  by looping over all  $\{u, v\} \in E$  with  $\text{lca}(u, v) = x$ . After this update we have that  $\text{cost}(e, f) = \text{cost}(e) + \text{score}(f)$  for every  $f \in T_e$ . This shows how to descend down  $T$  while keeping the invariant. The full tree is then explored by taking an Euler tour through  $T$ , and whenever we go back up in the tree we revert the score updates. This allows us to find candidate  $f \in T_e$  for every  $e \in E(T)$ . To bound the number of updates, note that each of the  $m$  edges has a unique lca, and we only do an update corresponding to an edge when the lca is visited by the Euler tour. Since the Euler tour visits every vertex at most twice, the number of updates is at most  $2m$ . In addition, the number of categorical top two queries is  $n - 1$ .

The resulting algorithm is formalized in the full version [1], and it leads to the following theorem.

► **Theorem 6.** *Given an assignment  $e.\text{color}$  for each  $e \in E(T)$ , there is a deterministic algorithm that runs in time  $\mathcal{O}(m \log^2 n)$  and for each  $e$  finds an  $f$  such that*

1.  $\{e, f\} \in H$
  2.  $e \in T_f$  or  $f \in T_e$
  3.  $e.\text{color} \neq f.\text{color}$
- if such an  $f$  exists.*

### 3.3 Independent edges

The goal now is to find, for every edge  $e \in E(T)$ , a partner  $f \in E(T)$  such that  $e, f$  are independent, or decide that there is no such  $f$ . As we chose the root of  $T$  to have degree 1, by Proposition 2 we do not have to worry about trivial cuts in the independent edge case.

Instead of considering all edges  $e \in E(T)$  one-by-one, we first find a so-called *heavy path decomposition* of  $T$  [13, 24], which is a partition of the edges of  $T$  into *heavy paths*. We define this partition recursively: first, find the heavy path starting at the root by repeatedly descending to the child of the current node with the largest subtree. This creates the topmost heavy path starting at the root (called its head) and terminating at a leaf (called its tail). Second, remove the topmost heavy path from  $T$  and repeat the reasoning on each of the obtained smaller trees. The crucial property is that, for any node  $u$ , the path from  $u$  to the root in  $T$  intersects at most  $\log n$  heavy paths.

We can now iterate over all pairs of heavy paths  $h, h'$  to look for a partner  $f \in h'$  for every  $e \in h$ . We cannot literally carry out this plan as the number of pairs of heavy paths can be  $\Omega(n^2)$  and so we cannot explicitly consider every pair. We show next that many pairs  $h, h'$  result in a trivial case and that all these trivial pairs can be solved together in one batch. We then bound the number of non-trivial pairs and show that in near-linear time we can explicitly process all of them. The idea of processing pairs of heavy paths, and explicitly considering only the non-trivial ones, was introduced in the context of 2-respecting cuts by Mukhopadhyay and Nanongkai [21] (see also [9]).

Consider two distinct heavy paths  $h, h'$ , where  $h$  is the path  $u_1 - u_2 - \dots - u_q$  and  $h'$  is the path  $v_1 - v_2 - \dots - v_{q'}$ . We let  $e_i = (u_i, u_{i+1})$  for  $i = 1, \dots, q-1$  and  $f_i = (v_i, v_{i+1})$  for  $i = 1, \dots, q'-1$ . It can be that not all pairs  $e_i, f_j$  are independent, see Figure 1. However, we can easily identify the subpaths of  $h, h'$  containing pairwise independent edges in constant time by computing the lowest common ancestor  $v$  of the tails of  $h, h'$ . If  $v = v_{p'}$  lies on  $h'$  then  $e_i, f_j$  will be independent for  $1 \leq i < q$  and  $p' \leq j < q'$ , and similarly if  $v$  lies on  $h$ . In general we assume that  $p, p'$  have been determined so that  $e_i, f_j$  are independent for all  $p' \leq i < q$  and  $p' \leq j < q'$ , and that these pairs comprise all of the independent pairs on  $h, h'$ . We can associate to  $h, h'$  a  $(q-1)$ -by- $(q'-1)$  matrix  $M^{(h, h')}$  where for  $p' \leq i < q$  and  $p' \leq j < q'$

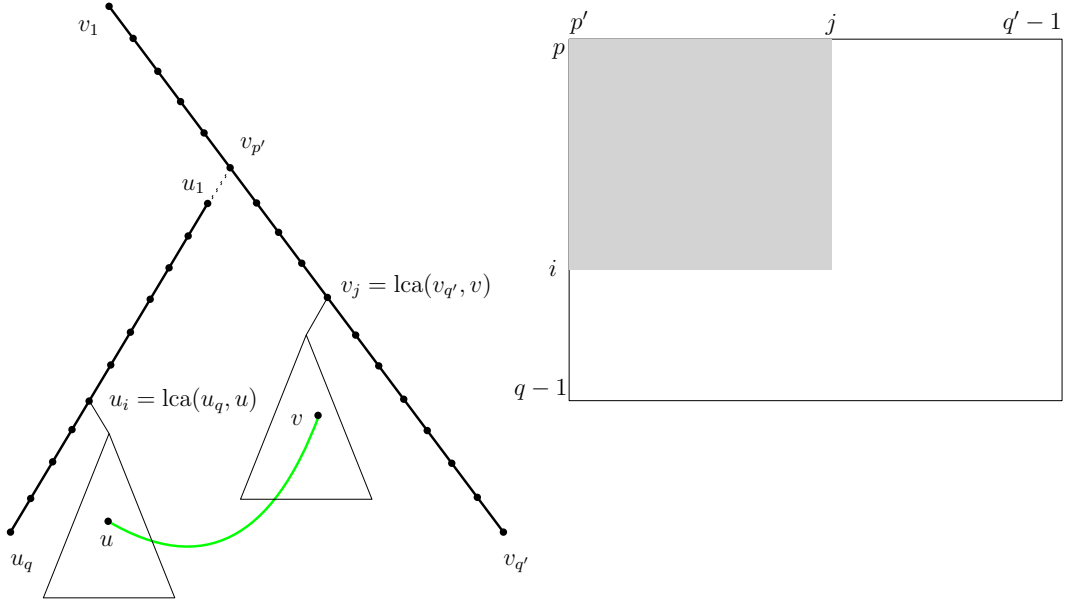
$$\begin{aligned} M^{(h, h')}[i, j] &= \text{cost}(e_i, f_j) \\ &= \text{cost}(e_i) + \text{cost}(f_j) - 2w(e_i^\downarrow, f_j^\downarrow), \end{aligned} \quad (3)$$

and  $M^{(h, h')}$  is undefined otherwise.<sup>4</sup> All values of  $\text{cost}(e)$  can be computed in  $\mathcal{O}(m)$  total time [15]. To efficiently evaluate  $M^{(h, h')}$ , we will prepare a list  $L(h, h')$  of all edges that contribute to  $w(e^\downarrow, f^\downarrow)$  for independent  $e, f$  with  $e \in h, f \in h'$ . For many  $h, h'$  the list  $L(h, h')$  will be empty, leading to the trivial case mentioned above. The following lemma bounds the size of all the non-empty lists and shows they can be constructed efficiently (proof in full version [1]).

► **Lemma 7.** *The total length of all lists  $L(h, h')$  is  $\mathcal{O}(m \log^2 n)$  and all non-empty lists  $L(h, h')$  can be constructed deterministically in time  $\mathcal{O}(m \log^2 n)$ .*

We can now describe how to find a partner  $f$  for every  $e$  such that  $e, f$  are independent. The algorithm first solves together in one batch the case where the partner of  $e \in h$  is in a heavy path  $h'$  where  $L(h, h')$  is empty. After that we explicitly consider all  $h, h'$  with  $L(h, h')$  non-empty. We consider these two cases in the next two subsections.

<sup>4</sup> We could restrict  $M^{(h, h')}$  to the submatrix on which it is defined, but find it notationally easier for the  $i, j$  indices in  $M^{(h, h')}$  to match the edge labels.



■ **Figure 1** Contribution of an edge  $\{u, v\} \in L(h, h')$  (denoted in green on the left) to  $M^{(h, h')}[., .]$  (denoted in grey on the right).

### 3.3.1 Empty lists

► **Lemma 8.** *There is a deterministic algorithm that in time  $\mathcal{O}(m + n)$  finds a partner for every edge  $e \in E(T)$  that has a partner  $f$  such that  $e, f$  are independent and  $e \in h, f \in h'$  with  $L(h, h')$  empty.*

**Proof.** The key observation is that if  $L(h, h')$  is empty then  $\text{cost}(e, f) = \text{cost}(e) + \text{cost}(f)$  by Equation (3). As can be seen from Equation (1) and Equation (3), for any edge  $f'$  it always holds that  $\text{cost}(e, f') \leq \text{cost}(e) + \text{cost}(f')$ , whether  $e, f'$  are descendant or independent. Thus in this case it suffices for us to find *any*  $f'$  of color different from  $e$  such that  $\text{cost}(e) + \text{cost}(f') \leq \beta$ , and  $\text{cut}(e, f')$  is non-trivial as this ensures  $\text{cost}(e, f') \leq \beta$ . We are guaranteed such an  $f'$  exists as  $f$  satisfies this.

We can compute  $\text{cost}(f')$  for every  $f' \in E(T)$  in time  $\mathcal{O}(m)$  [15]. Then in time  $\mathcal{O}(n)$  with one pass over  $E(T)$  we compute the edge  $f_1$  of lowest cost and the edge  $f_2$  of lowest cost that is of color different to  $f_1$ . We then repeat this categorical top two query twice more, each time excluding all previously found edges. At the end we obtain edges  $f_1, \dots, f_6$ . We claim that for every  $e$ , at least one of these must be a valid partner.

Consider any particular  $e$ . The first categorical top two query can only fail to find a valid partner for  $e$  if one of  $f_1, f_2$  creates a trivial cut with  $e$ . In this case, the second categorical top two query can only fail if one of  $f_3, f_4$  creates a trivial cut with  $e$  as well. By Proposition 2, however, there are at most two possible edges that can create a trivial cut with  $e$ , thus in this case the third categorical top two query must succeed and we find a valid partner for  $e$ . ◀

### 3.3.2 Non-empty lists

The more difficult case is to find partners among pairs  $h, h'$  with  $L(h, h')$  non-empty. While we defer the proof details to the full paper [1], we describe the key insight of the algorithm, which relies on the special structure of  $M^{(h, h')}$ . As above, say that  $h$  is the path  $u_1 - u_2 - \dots - u_q$

and  $h'$  is the path  $v_1 - v_2 - \dots - v_{q'}$ , and let  $e_i = (u_i, u_{i+1})$  for  $i = 1, \dots, q-1$  and  $f_i = (v_i, v_{i+1})$  for  $i = 1, \dots, q'-1$ . Further suppose  $e_i, f_j$  are independent for all  $p \leq i < q, p' \leq j < q'$ . We have that  $M^{(h,h')}[i, j] = \text{cost}(e_i) + \text{cost}(f_j) - 2w(e_i^\downarrow, f_j^\downarrow)$  for  $p \leq i < q, p' \leq j < q'$ . Recall that  $L(h, h')$  is defined precisely as the list of edges that contribute to  $w(e^\downarrow, f'^\downarrow)$  for independent  $e \in h, f' \in h'$ . The contribution of a specific edge  $\{u, v\} \in L(h, h')$  can be understood as follows: let  $u_i$  be the lowest common ancestor of  $u$  and  $u_q$ , and  $v_j$  be the lowest common ancestor of  $v$  and  $v_{q'}$ . Then the weight of  $\{u, v\}$  contributes to  $M[a, b]$  for every  $p \leq a \leq i, p' \leq b \leq j$ . This is depicted in Figure 1. We can compute these indices  $i$  and  $j$  for every  $\{u, v\} \in L(h, h')$ . This takes constant time per edge using an appropriate LCA structure [6], and so total time  $\mathcal{O}(|L(h, h')|)$ .

Our algorithm crucially builds on this “sparse” representation of  $M^{(h,h')}$  to efficiently transfer non-empty lists. The following lemma is proven in the full version [1].

► **Lemma 9.** *Let  $\mathcal{F} = \{e \mid \exists h, h', f : e \in h, f \in h', e, f \text{ are partners and } L(h, h') \text{ non-empty}\}$ . There is a deterministic algorithm to find a partner for every  $e \in \mathcal{F}$  in time  $\mathcal{O}(m \log^3 n)$ .*

## References

- 1 Simon Apers, Paweł Gawrychowski, and Troy Lee. Finding the KT partition of a weighted graph in near-linear time. *CoRR*, abs/2111.01378, 2021. [arXiv:2111.01378](#).
- 2 Simon Apers and Troy Lee. Quantum complexity of minimum cut. In *Proceedings of the 36th Computational Complexity Conference (CCC '21)*, pages 28:1–28:3. LIPIcs, 2021.
- 3 András Benczúr. *Cut structures and randomized algorithms in edge-connectivity problems*. PhD thesis, MIT, 1997.
- 4 András A. Benczúr. A representation of cuts within  $6/5$  times the edge connectivity with applications. In *Proceedings of 36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, pages 92–102. IEEE Computer Society, 1995. [doi:10.1109/SFCS.1995.492466](#).
- 5 András A. Benczúr and Michel X. Goemans. Deformable polygon representation and near-mincuts. In Martin Grötschel, Gyula O. H. Katona, and Gábor Sági, editors, *Building Bridges: Between Mathematics and Computer Science*, pages 103–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. [doi:10.1007/978-3-540-85221-6\\_3](#).
- 6 Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In *Proceedings of 4th Latin American Symposium on Theoretical Informatics (LATIN '00)*, pages 88–94. Springer, 2000.
- 7 Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii (Studies in Discrete Optimization)*, pages 290–306, 1976. Appeared in Russian.
- 8 Harold N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, 1995. [doi:10.1006/jcss.1995.1022](#).
- 9 Paweł Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in  $\mathcal{O}(m \log^2 n)$  time. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP '20)*, volume 168 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 10 Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '20)*, pages 1260–1279. SIAM, 2020. [doi:10.1137/1.9781611975994.77](#).
- 11 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '11)*, pages 550–559. IEEE, 2011.

- 12 Ralph E. Gomory and Te C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. URL: <http://www.jstor.org/stable/2098881>.
- 13 Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.
- 14 Monika Henzinger, Satish Rao, and Di Wang. Local flow partitioning for faster edge connectivity. *SIAM Journal on Computing*, 49(1):1–36, 2020. doi:10.1137/18M1180335.
- 15 David R. Karger. Minimum cuts in near-linear time. *Journal of the ACM*, 47(1):46–76, 2000. Announced at STOC 1996.
- 16 David R. Karger and Debmalya Panigrahi. A near-linear time algorithm for constructing a cactus representation of minimum cuts. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*, pages 246–255. SIAM, 2009. URL: <http://dl.acm.org/citation.cfm?id=1496770.1496798>.
- 17 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *53rd Annual ACM-SIGACT Symposium on Theory of Computing (STOC '21)*, pages 32–45, 2021.
- 18 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *Journal of the ACM*, 66(1):4:1–4:50, 2019. Announced at STOC 2015. doi:10.1145/3274663.
- 19 Jason Li. Deterministic mincut in almost-linear time. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC '21)*, pages 384–395. ACM, 2021. doi:10.1145/3406325.3451114.
- 20 On-Hei S. Lo, Jens M. Schmidt, and Mikkel Thorup. Compact cactus representations of all non-trivial min-cuts. *Discrete Applied Mathematics*, 303:296–304, 2020. doi:10.1016/j.dam.2020.03.046.
- 21 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20)*, pages 496–509. ACM, 2020.
- 22 Jaroslav Nesetril, Eva Milková, and Helena Nesetrilová. Otakar Borůvka on the minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1-3):3–36, 2001. doi:10.1016/S0012-365X(00)00224-7.
- 23 Aviad Rubinfeld, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS '18)*, pages 39:1–39:16. LIPIcs, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.
- 24 Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.



# Maximum Matching Sans Maximal Matching: A New Approach for Finding Maximum Matchings in the Data Stream Model

Moran Feldman  

Department of Computer Science, University of Haifa, Israel

Ariel Szarf 

Department of Mathematics and Computer Science, Open University of Israel, Ra'anana, Israel

---

## Abstract

The problem of finding a maximum size matching in a graph (known as the *maximum matching* problem) is one of the most classical problems in computer science. Despite a significant body of work dedicated to the study of this problem in the data stream model, the state-of-the-art single-pass semi-streaming algorithm for it is still a simple greedy algorithm that computes a maximal matching, and this way obtains  $1/2$ -approximation. Some previous works described two/three-pass algorithms that improve over this approximation ratio by using their second and third passes to improve the above mentioned maximal matching. One contribution of this paper continues this line of work by presenting new three-pass semi-streaming algorithms that work along these lines and obtain improved approximation ratios of 0.6111 and 0.5694 for triangle-free and general graphs, respectively.

Unfortunately, a recent work [30] shows that the strategy of constructing a maximal matching in the first pass and then improving it in further passes has limitations. Additionally, this technique is unlikely to get us closer to single-pass semi-streaming algorithms obtaining a better than  $1/2$ -approximation. Therefore, it is interesting to come up with algorithms that do something else with their first pass (we term such algorithms non-maximal-matching-first algorithms). No such algorithms are currently known (to the best of our knowledge), and the main contribution of this paper is describing such algorithms that obtain approximation ratios of 0.5384 and 0.5555 in two and three passes, respectively, for general graphs (the result for three passes improves over the previous state-of-the-art, but is worse than the result of this paper mentioned in the previous paragraph for general graphs). The improvements obtained by these results are, unfortunately, numerically not very impressive, but the main importance (in our opinion) of these results is in demonstrating the potential of non-maximal-matching-first algorithms.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Theory of computation  $\rightarrow$  Graph algorithms analysis; Theory of computation  $\rightarrow$  Approximation algorithms analysis

**Keywords and phrases** Maximum matching, semi-streaming algorithms, multi-pass algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.33

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2109.05946> [20]

**Funding** *Moran Feldman:* Research supported in part by Israel Science Foundation (ISF) grant number 459/20.

## 1 Introduction

The problem of finding a maximum size matching in a graph (known as the *maximum matching* problem) is one of the most classical problems in computer science, and many polynomial time algorithms have been designed for it over the years (see, e.g., [9, 15, 23]). Due to its central role, the maximum matching problem is often one of the first problems



© Moran Feldman and Ariel Szarf;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 33; pp. 33:1–33:24



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

considered when new computational models are suggested. One such model is the data stream model, which is motivated by Big-Data applications, and has been the subject of an enormous amount of research over the last couple of decades.

In the data stream model, the algorithm receives the input in the form of a stream which it can read sequentially, but due to memory restrictions, the algorithm can store only a small part of this stream. This means that the algorithm has to process (in some sense) the input stream while reading it, and never gets an opportunity to see all the parts of the input at the same time. Traditional algorithms for this model, known as *streaming algorithms*, are allowed only memory that is poly-logarithmic in the natural parameters of the problem. Obtaining a streaming algorithm for a problem is very desirable, but is often not possible. In particular, many graph problems provably do not admit streaming algorithms, and the maximum matching problem is among these problems if one would like an algorithm for the problem to output an (approximately) maximum matching because such a matching might be of linear size in the number of vertices. Nevertheless, non-trivial streaming algorithms have been designed for the maximum matching problem when only the (approximate) size of a maximum matching is desired (see Section 1.1 for details).

The resistance of many graph problems to streaming algorithms has motivated Feigenbaum et al. [19] to suggest semi-streaming algorithms, which are algorithms for the data stream model that are allowed a space complexity of  $O(n \log^c n)$  for some constant  $c \geq 0$ , where  $n$  is the number of vertices in the graph. Such algorithms turn out to be a sweet-spot that on the one hand allows many results of interest, and on the other hand, does not lead to triviality because  $O(n \log^c n)$  is less than the space necessary for storing the input graph (unless this graph is very sparse). In particular, Feigenbaum et al. [19] observed that one can obtain  $1/2$ -approximation for the maximum matching problem using a simple semi-streaming algorithm that greedily constructs a maximal matching.<sup>1</sup>

The above  $1/2$ -approximation semi-streaming algorithm for the maximum matching problem also has the desirable property that it reads the input stream only once (i.e., it makes a single pass over it). Surprisingly, no single-pass semi-streaming algorithm improving over the approximation ratio of this simple algorithm was suggested in the decade and a half that has already passed since the work of [19] (in contrast, Kapralov [26] showed that no such algorithm can have an approximation ratio better than  $1/(1 + \ln 2) \approx 0.59$ , improving over previous inapproximability results due to [22, 25]). Given this lack of progress, interest arose in obtaining improved approximation ratios for relaxed versions of the above problem. Perhaps, one of the simplest such relaxations is to allow the algorithm to make a few (usually two or three) sequential passes over the input stream.

The last line of work was introduced by Konrad et al. [29], and was later studied by [18]. The state-of-the-art results for it are summarized in Table 1. We note that beside the state-of-the-art results for general input graphs, Table 1 also gives improved results for bipartite and triangle-free graphs. All the known results in this line of work (to the best of our knowledge) start by greedily constructing a maximal matching during the first pass over the input stream, and then augmenting this matching in the subsequent passes. Recently, Konrad and Naidu [30] showed that this technique has limitations (specifically, even for bipartite graphs, a two-pass semi-streaming algorithm based on this technique cannot obtain a better than  $2/3$ -approximation, which is much more strict than the inapproximability known for general two-pass semi-streaming algorithms [2]). Additionally, and arguably

---

<sup>1</sup> A maximal matching is a matching that is inclusion-wise maximal, and it is well-known that the size of any maximal matching is a  $1/2$ -approximation for the size of a maximum matching.

■ **Table 1** The state-of-the-art approximation ratios for semi-streaming algorithms using two or three passes, and our improvements over these ratios (the number to the right of each improvement is the number of the theorem formally stating it).

| Number of Passes | Type of Graphs | State-of-the-Art                                                                                | This Paper                                                                                           | Approach       |
|------------------|----------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|----------------|
| Two-Pass         | Bipartite      | $2 - \sqrt{2} \approx \frac{1}{2} + \frac{1}{11.66}$ [28]<br>$\approx 0.5857$                   | –                                                                                                    | –              |
|                  | Triangle-Free  | $\frac{1}{2} + \frac{1}{16} = 0.5625$ [24]                                                      | –                                                                                                    | –              |
|                  | General        | $\frac{1}{2} + \frac{1}{32} = 0.53125$ [24]                                                     | $\frac{1}{2} + \frac{1}{26} \approx 0.5385$ (1)                                                      | non-MMF        |
| Three-Pass       | Bipartite      | $0.6067 \approx \frac{1}{2} + \frac{1}{9.37}$ [28]                                              | $\frac{1}{2} + \frac{1}{9} \approx 0.6111$ (3)                                                       | MMF            |
|                  | Triangle-Free  | $\frac{1}{2} + \frac{1}{10} = 0.6$ [24]                                                         | $\frac{1}{2} + \frac{1}{9} \approx 0.6111$ (3)                                                       | MMF            |
|                  | General        | $\frac{1}{2} + \frac{81}{1600} \approx \frac{1}{2} + \frac{1}{19.753}$ [24]<br>$\approx 0.5506$ | $\frac{1}{2} + \frac{1}{14.4} \approx 0.5694$ (4)<br>$\frac{1}{2} + \frac{1}{18} \approx 0.5555$ (2) | MMF<br>non-MMF |

more importantly, multi-pass algorithms that use their first pass for constructing a maximal matching are unlikely to be a step towards a single-pass semi-streaming algorithm with a better than  $1/2$ -approximation guarantee.

Given the above observations, it is natural to believe that the future of the study of semi-streaming algorithms for the maximum matching problem lies in algorithms that use their first pass in a more sophisticated way than simply constructing the traditional maximal matching. We term such algorithms *non-maximal-matching-first algorithms* (or non-MMF algorithms for short). In this paper, we present the first non-MMF algorithms, which leads to improvements over the state-of-the-art both for two and three passes. Admittedly, the improvements we obtain are numerically not very impressive, but their main importance (in our opinion) is in demonstrating the potential of non-MMF algorithms.

To intuitively understand our non-MMF algorithms, one should note that greedily constructing a maximal matching is equivalent to greedily constructing a graph whose connected components are of size at most 2 (where the size of a connected component is defined as the number of vertices in it). Therefore, a natural generalization is to greedily construct in the first pass a graph whose connected components are of size at most 3. There are two intuitive advantages for doing that compared to constructing a maximal matching.

- A connected component of size 3 can contribute two edges to the output matching if it is “augmented” during the next passes with a single additional edge. In contrast, doing the same with a connected component of size 2 requires “augmenting” it with two additional edges. It is important to note that there is a significant conceptual difference between an augmentation of a connected component with one or two edges. Augmenting a connected component with two edges requires finding pairs of edges that augment the *same* connected component, while augmenting with a single edge does not require such a synchronization.
- If we are not able to enjoy the above advantage because many connected components end up being of size 2 rather than 3. Then, the fact that this has happened despite us only restricting the components to be of size at most 3 implies that few edges of a maximum matching intersect only a single connected component of the constructed graph; and therefore, the constructed graph must have many connected components compared to the size of a maximum matching.

Using the above ideas, we prove the following two theorems. The proof of Theorem 1 appears in Section 3, while the adaptation of this proof leading to Theorem 2 is deferred to the full version of this paper [20].

► **Theorem 1.** *There exists a non-MMF 2-pass  $(7/13 = 1/2 + 1/26)$ -approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

► **Theorem 2.** *There exists a non-MMF 3-pass  $(5/9 = 1/2 + 1/18)$ -approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

As mentioned above, both Theorems 1 and 2 represent an improvement over the state-of-the-art. However, it turns out that we can further improve over Theorem 2 using new MMF algorithms (i.e., algorithms that construct a maximal matching in their first pass). This leads to the following theorems whose proofs appear in Section 4 and Appendix A, respectively.

► **Theorem 3.** *There exists a 3-pass  $(11/18 = 1/2 + 1/9)$ -approximation semi-streaming algorithm for finding a maximum size matching in a triangle-free graph.<sup>2</sup>*

► **Theorem 4.** *There exists a 3-pass  $(1/2 + 1/14.4)$ -approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithms used to prove Theorems 3 and 4 are strongly based on the algorithms suggested by Kale and Tirodkar [24]. For example, the first two passes of the algorithm suggested by Theorem 3 are identical to a two-pass algorithm presented by [24], and the third pass of this algorithm is very similar to the third pass of the three-pass algorithm of [24]. Our novelty, however, is in our ability to analyze the algorithm obtained by putting these two components together.

## 1.1 Related Work

As mentioned in Section 1, streaming algorithms are not appropriate for the maximum matching problem when the algorithm is required to output an (approximately) maximum matching. However, some non-trivial streaming algorithms are known for this problem when the algorithm is only required to estimate the size of the maximum matching. Kapralov et al. [27] designed a poly-log approximation streaming algorithm for this problem under the assumption that the edges in the input stream are ordered in a uniformly random order. A different line of work [13, 17, 32] considered graphs of bounded arboricity  $\alpha$ , cumulating with the work of McGregor and Vorotnikova [33], who designed  $(\alpha + 2)(1 + \varepsilon)$ -approximation streaming algorithm for this problem requiring only  $O(\varepsilon^{-2} \log n)$  space. In contrast, Assadi et al. [5] showed that  $(1 - \varepsilon)$ -approximation of the size of the maximum matching cannot be obtained by a single pass algorithm, even if this algorithm is allowed a semi-streaming space complexity, and [6, 8] lower bounded the number of passes required to obtain such a good approximation using a sub-polynomial space complexity.

Recall that, to date, the best single-pass semi-streaming algorithm for the maximum matching problem is still the natural greedy algorithm, which guarantees  $1/2$ -approximation. Chitnis et al. [12] presented an exact single-pass algorithm for this problem. However, this algorithm requires  $\tilde{O}(k^2)$  memory, where  $k$  is an upper bound on the size of the maximum matching (which the algorithm needs to know upfront), and thus, this algorithm is a semi-streaming algorithm only when  $k = \tilde{O}(\sqrt{n})$ . Given the difficulty to improve over the guarantee of the greedy algorithm using single-pass semi-streaming algorithms, people

---

<sup>2</sup> We recall that every bipartite graph is triangle-free, and therefore, the same result is obtained also for bipartite graphs.

consider also relaxed versions of the maximum matching problem. One standard relaxation is to allow the algorithm to make multiple passes over the input stream. Section 1 surveys algorithms of this kind that use two or three passes. Another line of work considers algorithms that assume a constant (but possibly large) number of passes. The first result of this kind was presented by Feigenbaum et al. [19] (in the same paper that also introduced the notion of semi-streaming algorithms), and guaranteed  $(2/3 - \varepsilon)$ -approximation using  $O(\varepsilon^{-1} \log \varepsilon^{-1})$  passes for bipartite graphs. Later [31] showed how to obtain  $(1 - \varepsilon)$ -approximation for general graphs using  $(\varepsilon^{-1})^{O(\varepsilon^{-1})}$  passes, and the number of passes necessary to obtain this guarantee was improved by many further works (see, e.g., [1, 4, 7, 21]). Another standard relaxation for the maximum matching problem is to assume that the edges of the input stream appear in a uniformly random order. The state-of-the-art for this relaxation is a  $(2/3 + \varepsilon_0)$ -approximation single-pass semi-streaming algorithm, where  $\varepsilon_0 > 0$  is some absolute constant [3, 10] (see also the references therein for previous works on this relaxation).

The related maximum weight matching problem was also studied heavily in the context of the data stream model. Here, it is not immediately clear that one can obtain a constant approximation ratio using a single-pass semi-streaming algorithm. However, Feigenbaum et al. [19] presented the first such algorithm guaranteeing  $1/6$ -approximation, and this ratio was improved in series of works [14, 16, 31, 35]. The current state-of-the-art for the problem is  $(1/2 - \varepsilon)$ -approximation due to Paz and Schwartzman [34]. Since this approximation ratio is essentially identical to the state-of-the-art for the (unweighted) maximum matching problem, any further progress on the maximum weight matching problem (beyond removing the  $\varepsilon$ ) will imply an improvement over the guarantee of the greedy algorithm for the (unweighted) maximum matching problem. It is also worth mentioning that a recent reduction due to Bernstein et al. [11] shows that the reverse is also true in a sense. More specifically, any semi-streaming algorithm for bipartite unweighted graphs can be translated into such an algorithm for weighted graphs with the same number of passes and a loss of only  $1 - \varepsilon$  in the approximation guarantee. Naturally, this reduction automatically extends some of our results to the weighted case.

## 2 Preliminaries

In this section we present the problem that we study more formally, and also introduce the notation used throughout the rest of the paper. We are interested in semi-streaming algorithms for the problem of finding a maximum size matching in a graph  $G = (V, E)$  of  $n$  vertices. A semi-streaming algorithm for this problem is an algorithm with a space complexity of  $O(n \log^c n)$  (for some constant  $c \geq 0$ ) that initially has no knowledge about the edges of  $E$ . Instead, the edges of  $E$  appear sequentially in an “input stream”, and the algorithm may make one or more passes over this input stream. In each pass the algorithm sees the edges one by one, and may do arbitrary calculations after viewing each edge. It is important to note that the space complexity allowed for the algorithm does not suffice for storing all the edges of the graph (unless the graph is very sparse), and this is the reason that the algorithm might benefit from doing multiple passes over the input stream. It is standard to assume that the vertices of  $V$  are known upfront, and that each vertex of  $V$  can be stored using  $O(\log n)$  bits (which implies that every edge of  $E$  can also be stored using this asymptotic number of bits).

Throughout the paper, we consider only unweighted graphs and matchings. We also denote by  $M^*$  an arbitrary maximum matching of  $G$  (i.e., an arbitrary optimal solution for our problem). Notation-wise, we treat  $M^*$  (and any other matching considered in the paper)

as a set of the edges included in it. Similarly, when considering a connected component  $C$  of a graph, we treat it as a set of the vertices in it, which in particular, implies that  $|C|$  is the number of such vertices.

Given a set of edges  $S$  or a path  $P$  in a graph, we denote by  $V(S)$  and  $V(P)$  the set of vertices intersecting any edge of  $S$  or  $P$ , respectively. Similarly, the set of edges included in the path  $P$  is denoted by  $E(P)$ . Often we need to consider collections of paths (or triangles) in a given graph. For clarity, such collections are always denoted using calligraphic letters, and we extend the above notation to such collections. In other words, if  $\mathcal{P}$  is a collection of paths, then  $V(\mathcal{P})$  and  $E(\mathcal{P})$  is the set of vertices and edges, respectively, that are included in these paths. Finally, given a set  $S$  of edges and a vertex  $v$ , we use  $\deg_S(v)$  to denote the degree of the vertex  $v$  in the subgraph  $(V, S)$ .

### 3 Two-Pass Non-MMF Algorithm

In this section we prove Theorem 1, which we repeat below for convenience.

► **Theorem 1.** *There exists a non-MMF 2-pass  $(7/13 = 1/2 + 1/26)$ -approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithm whose existence is guaranteed by Theorem 1 appears as Algorithm 1. In its first pass, this algorithm greedily grows a set  $P$  of edges that form either triangles or partial triangles (i.e., isolated edges or paths of length 2). For simplicity, we refer below to the connected components of  $(V, P)$  that are not isolated vertices as partial triangles although, technically, they can also be full triangles. In the second pass of Algorithm 1, the algorithm tries to convert the partial triangles of  $P$  into more involved structures in one of two ways. To understand these ways, we need to define some terms. First, we designate some of the vertices of every partial triangle as “connection vertices”. Specifically, all the vertices of a triangle are considered connection vertices; in a path of length 2 only the two end points are considered to be connection vertices; and finally, in an isolated edge there are no connection vertices. We refer to a partial triangle that was not converted yet into a more involved structure as a “naïve” partial triangle. The first way in which Algorithm 1 tries to convert the partial triangles of  $P$  into more involved structures is by greedily adding edges that connect a connection vertex of a naïve partial triangle with an isolated vertex. The set  $A_1$  in the algorithm includes the edges that were added in this way. In parallel, the algorithm also tries a second way to convert the partial triangles of  $P$  into more involved structures, which is to greedily add edges that connect a connection vertex of a naïve partial triangle either to a connection vertex of another naïve partial triangle or to an isolated vertex. The set  $A_2$  in the algorithm includes the edges that were added in this way. Upon termination, Algorithm 1 outputs a maximum matching in the set of all the edges that it kept. We recall that given a connected component  $C$  of a graph, the notation  $|C|$  represents the number of vertices in  $C$ .

We begin the analysis of Algorithm 1 by noting that it is indeed a semi-streaming algorithm. The proof of the next observation can be found in the full version of this paper [20].

► **Observation 5.** *Algorithm 1 is a semi-streaming algorithm.*

In the rest of this section we analyze the approximation ratio of Algorithm 1. Recall that we use  $M^*$  to denote some maximum matching of  $G$ . Our first objective in the analysis of the approximation ratio of Algorithm 1 is to lower bound the number of edges of  $M^*$  that can potentially be added either to  $A_1$  or to  $A_2$ . Towards this goal, we define a charging scheme



---

**Algorithm 1** MAXIMUM MATCHING VIA GREEDY TRIANGLES – TWO PASSES.
 

---

```

// First Pass
1 Let $P \leftarrow \emptyset$.
2 for every edge e that arrives do
3 if every connected component of the graph $(V, P \cup \{e\})$ is either an isolated vertex,
 a path of length at most 2 or a triangle (cycle of size 3) then Add e to P .

// Second Pass
4 Let $A_1 \leftarrow \emptyset$ and $A_2 \leftarrow \emptyset$.
5 for every edge $(u, v) \notin P$ that arrives do
6 Let C_u and C_v be the connected components of u and v , respectively, in (V, P) .
 We assume without loss of generality that $|C_u| > 1$, otherwise we swap the
 roles of u and v . // Note that we cannot have $|C_u| = |C_v| = 1$ because
 the edge (u, v) was not added to P in the first pass.
7 if no edge of A_1 intersects C_u and C_v , $|C_v| = 1$ and u is a connection vertex of
 C_u then Add the edge (u, v) to A_1 .
8 if no edge of A_2 intersects C_u and C_v , $|C_v| = 1$ and u is a connection vertex of
 C_u then Add the edge (u, v) to A_2 .
9 else if no edge of A_2 intersects C_u and C_v , and u and v are connection vertices
 of C_u and C_v , respectively then Add the edge (u, v) to A_2 .

10 return a maximum matching in the graph $(V, P \cup A_1 \cup A_2)$.

```

---

$\pi$ . Under the charging scheme  $\pi$ , every edge  $(u, v) \in M^*$  charges the connected components of  $u$  and  $v$  in  $(V, P)$ . Each one of these connected components is charged one unit by  $(u, v)$ , unless it is an isolated edge or an isolated vertex, in which case it is charged only half a unit or nothing by  $(u, v)$ , respectively. We note that when  $u$  and  $v$  belong to the same connected component of  $(V, P)$ , then this connected component is charged twice by  $(u, v)$ .<sup>3</sup>

The following observation provides an upper bound on the total charged by all the edges of  $M^*$  together. Let  $(\#single)$  be the number of isolated edges in  $P$ ,  $(\#double)$  be the number of connected components in  $(V, P)$  that are paths of length 2 and  $(\#triangle)$  be the number of triangles in  $P$ .

► **Observation 6.** *The total charge according to  $\pi$  is at most  $(\#single) + 3(\#double) + 3(\#triangle)$ .*

**Proof.** Every positive amount charged by  $\pi$  is charged to some connected component of  $(V, P)$  which is not an isolated vertex. Therefore, to prove the observation we only need to show that every isolated edge of  $(V, P)$  is charged at most one unit, and every connected component of  $(V, P)$  that is either a path of length 2 or a triangle is charged at most 3 units. Below we argue that this is indeed the case.

Each connected component  $C$  of  $(V, P)$  can be charged at most once for every one of its vertices since the fact that  $M^*$  is a matching implies that every vertex of  $C$  can appear in at most a single edge of  $M^*$ . For isolated edges of  $(V, P)$ , this implies that they can be

---

<sup>3</sup> Intuitively, the charge assigned to the connected components of  $u$  and  $v$  is proportional to the “blame” that can be assigned to them if  $(u, v)$  ends up to be outside  $P$ . For example, an isolated edge could not alone prevent  $(u, v)$  from being added to  $P$ , but two such edges (one intersecting  $u$  and the other intersecting  $v$ ) could, together, prevent  $(u, v)$  from being added to  $P$ . Therefore, we assign a charge of  $1/2$  to isolated edges. Observation 7 is based on this intuition.



charged at most twice, and therefore, they are charged at most one unit because they are charged half a unit in each charge. Similarly, connected components of  $(V, P)$  that are either paths of length 2 or triangles contain 3 vertices, and therefore, can be charged at most three times. Since every one of these charges is of a single unit, the total charge to each connected component of these kinds is at most 3. ◀

To complement the last observation, let us now describe a simple lower bound on the total charging done by all the edges of  $M^*$  according to  $\pi$ . Let  $(\# \text{component-free})$  be the number of edges of  $M^*$  that connect a connection vertex of a connected component of  $(V, P)$  to an isolated vertex of  $(V, P)$ ,  $(\# \text{component-component})$  be the number of edges of  $M^*$  that connect connection vertices of two different connected components of  $(V, P)$ ,  $(\# \text{single-single})$  be the number of edges of  $M^*$  whose two end points belong to (not necessarily distinct) isolated edges of  $(V, P)$ ,  $(\# \text{single-component})$  be the number of edges of  $M^*$  that connect a vertex of an isolated edge of  $(V, P)$  with a connection vertex of some (other) connected component of  $(V, P)$  and  $(\# \text{middle})$  be the number of edges that either intersect the middle vertex of a length 2 path connected component of  $(V, P)$  or are included within a triangle connected component of  $(V, P)$ . For convenience, the definitions of the notation we use are summarized in Appendix B.

► **Observation 7.** *The total charge of all the edges of  $M^*$  according to the charging scheme  $\pi$  is at least  $(\# \text{component-free}) + 2(\# \text{component-component}) + (\# \text{single-single}) + 1.5(\# \text{single-component}) + (\# \text{middle})$ .*

**Proof.** Since the edges of  $M^*$  counted by  $(\# \text{component-free})$  intersect a connection vertex, they must intersect a connected component of  $(V, P)$  which is not an isolated vertex or an isolated edge, and therefore, they charge this connected component one unit. Hence, the total charge by all the edges counted by  $(\# \text{component-free})$  is at least  $(\# \text{component-free})$ . Similar logic shows that the total charge by all the edges counted by  $(\# \text{component-component})$ ,  $(\# \text{single-single})$ ,  $(\# \text{single-component})$  and  $(\# \text{middle})$  is at least  $2(\# \text{component-component})$ ,  $(\# \text{single-single})$ ,  $1.5(\# \text{single-component})$  and  $(\# \text{middle})$ , respectively. The observation now follows since the edges of  $M^*$  counted by  $(\# \text{component-free})$ ,  $(\# \text{component-component})$ ,  $(\# \text{single-single})$ ,  $(\# \text{middle})$  and  $(\# \text{single-component})$  are distinct. ◀

Combining Observations 6 and 7, we get the following inequality.

$$\begin{aligned} & (\# \text{component-free}) + 2(\# \text{component-component}) + (\# \text{single-single}) \\ & + 1.5(\# \text{single-component}) + (\# \text{middle}) \leq (\# \text{single}) + 3(\# \text{double}) + 3(\# \text{triangle}) . \end{aligned} \quad (1)$$

In its current form, Inequality (1) is not very useful. We later derive from it a more convenient inequality, but before doing this we need to prove a few other inequalities. Let  $(\# \text{non-}M^* \text{-triangles})$  denote the number of triangle connected components of  $(V, P)$  that do not include any edge of  $M^*$  within them.

► **Lemma 8.** *The following inequalities hold*

$$\begin{aligned} & (\# \text{component-free}) + (\# \text{component-component}) + (\# \text{single-single}) \\ & + (\# \text{middle}) + (\# \text{single-component}) \geq |M^*| , \end{aligned} \quad (2)$$

$$(\# \text{double}) + (\# \text{triangle}) - (\# \text{non-}M^* \text{-triangles}) \geq (\# \text{middle}) , \quad (3)$$

$$2(\# \text{single-single}) + (\# \text{single-component}) \leq 2(\# \text{single}) , \quad (4)$$

and they imply together

$$(\# \text{component-free}) + (\# \text{component-component}) + 2(\# \text{single}) \\ + (\# \text{double}) + (\# \text{triangle}) - (\# \text{non-}M^* \text{-triangles}) \geq |M^*| .$$

**Proof.** Since every edge that is included in a connected component of  $(V, P)$  which is a path of length 2 must include the middle vertex of this path, every edge  $e \in M^*$  that is not counted by either  $(\# \text{component-free})$ ,  $(\# \text{component-component})$ ,  $(\# \text{single-single})$ ,  $(\# \text{single-component})$  or  $(\# \text{middle})$  must either connect a vertex of an isolated edge of  $(V, P)$  to an isolated vertex or connect two isolated vertices of  $(V, P)$ . However, such edges cannot exist. Specifically, assume towards a contradiction that  $(u, v)$  is an edge of  $M^*$  such that  $u$  is an isolated vertex of  $(V, P)$  and  $v$  is either another isolated vertex of  $(V, P)$  or belongs to an isolated edge of this graph. Then, the edge  $(u, v)$  should have been added by Algorithm 1 to  $P$  upon arrival, which contradicts the fact that its end point  $u$  ended up as an isolated vertex of  $(V, P)$ . Hence, every edge  $e \in M^*$  is counted by either  $(\# \text{component-free})$ ,  $(\# \text{component-component})$ ,  $(\# \text{single-single})$ ,  $(\# \text{single-component})$  or  $(\# \text{middle})$ , which implies Inequality (2).

Recall that every edge counted by  $(\# \text{middle})$  must either be included in a triangle connected component of  $(V, P)$  or intersect the middle vertex of a path of length 2 connected component of  $(V, P)$ . Since  $M^*$  is a matching, only one edge of  $M^*$  can intersect the middle vertex of a given length 2 path or be included in a given triangle, and therefore, every edge counted by  $(\# \text{middle})$  can be associated with a distinct path of length 2 or triangle component of  $(V, P)$  that is not counted by  $(\# \text{non-}M^* \text{-triangles})$ , which implies Inequality (3).

Every edge counted by  $(\# \text{single-single})$  touches two end-points of isolated edges of  $(V, P)$ . Similarly, every edge counted by  $(\# \text{single-component})$  intersects an end-point of an isolated edge of  $(V, P)$ . Since every end-point of an isolated edge of  $(V, P)$  can be touched by at most a single edge of  $M^*$  because  $M^*$  is a matching, this implies that the number of end points of the isolated edges of  $(V, P)$  is at least  $2(\# \text{single-single}) + (\# \text{single-component})$ . However, this number is also equal to  $2(\# \text{single})$ , which implies Inequality (4). ◀

The last inequality in the previous lemma provides a lower bound on  $(\# \text{component-free}) + (\# \text{component-component})$ , and one can view  $(\# \text{component-free}) + (\# \text{component-component})$  as a count of edges of  $M^*$  that have potential to be added to  $A_2$  in Algorithm 1. The next lemma is the promised derivative of Inequality (1), and it provides a lower bound on  $(\# \text{component-free})$ . Observe that  $(\# \text{component-free})$  is a count of edges of  $M^*$  that have the potential to be added to  $A_1$ .

► **Lemma 9.**  $2|M^*| \leq (\# \text{component-free}) - (\# \text{non-}M^* \text{-triangles}) + 2(\# \text{single}) + 4(\# \text{double}) + 4(\# \text{triangle})$ .

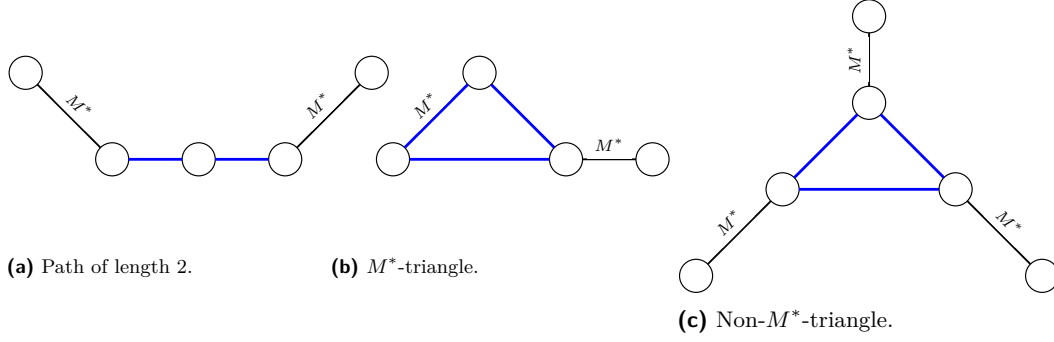
**Proof.** Adding twice Inequality (2) to Inequality (1), we get

$$2|M^*| - (\# \text{component-free}) - (\# \text{single-single}) - 0.5(\# \text{single-component}) - (\# \text{middle}) \\ \leq (\# \text{single}) + 3(\# \text{double}) + 3(\# \text{triangle}) .$$

The lemma now follows by adding Inequality (3) and half of Inequality (4) to the last inequality. ◀

So far we have shown lower bounds on the size of the sets of edges that have a potential to be added to  $A_1$  or  $A_2$  by Algorithm 1. Our next step is to lower bound the size of the sets  $A_1$  and  $A_2$  that Algorithm 1 ends up constructing using this potential.

### 33:10 Maximum Matching Sans Maximal Matching



■ **Figure 1** A graphical study of the maximum number of  $M^*$  edges that can intersect connection vertices of various types of partial triangles. Sub-figures (a) and (b) show that at most two such edges can intersect the connection vertices of a path of length 2 and an  $M^*$ -triangle (i.e., a triangle that includes an edge of  $M^*$ ). Sub-figure (c) shows that the connection vertices of a non- $M^*$ -triangle can intersect up to 3 edges of  $M^*$ .

► **Lemma 10.**  $3|A_1| \geq (\# \text{component-free}) - (\# \text{non-}M^* \text{-triangles})$ .

**Proof.** We say that an edge  $e$  of  $M^*$  counted by  $(\# \text{component-free})$  is excluded by an edge  $f \in A_1$  if  $e$  and  $f$  intersect the same connected component of  $(V, P)$ . One can observe that every edge  $e$  counted by  $(\# \text{component-free})$  is excluded by some edge of  $A_1$  (possibly itself) when Algorithm 1 terminates because otherwise Algorithm 1 would have added  $e$  to  $A_1$ , which would have resulted in  $e$  excluding itself. Therefore, we can upper bound  $(\# \text{component-free})$  by counting the number of edges excluded by the edges of  $A_1$ .

Let  $(u, v)$  be an edge of  $A_1$ , and assume without loss of generality that  $v$  is the end point of this edge which is an isolated vertex of  $(V, P)$ . This implies that  $u$  is a connection vertex of a connected component  $C_u$  of  $(V, P)$  which is either a path of length 2 or a triangle. If  $C_u$  is a path of length 2, then the edge  $(u, v)$  can exclude only edges counted by  $(\# \text{component-free})$  that intersect either  $v$  or a connection vertex of  $C_u$ , and there can be only 3 such edges because  $M^*$  is a matching (see Figure 1a). Next, consider the case in which  $C_u$  is a triangle which is not counted by  $(\# \text{non-}M^* \text{-triangles})$ . In this case there can be at most 2 edges of  $M^*$  intersecting  $C_u$  (see Figure 1b), and since  $(u, v)$  can exclude only edges that intersect either  $C_u$  or  $v$ , we get that it can exclude at most 3 edges.<sup>4</sup> It remains to consider the case in which  $C_u$  is a triangle counted by  $(\# \text{non-}M^* \text{-triangles})$ . In this case,  $(u, v)$  can again exclude every edge of  $M^*$  that intersects  $C_u$  or  $v$ , and this time there can be at most 4 such edges (see Figure 1c). Combining all the above, we get that the number of edges excluded by all the edges of  $A_1$  is at most

$$3|A_1| + |\{e \in A_1 \mid e \text{ intersects a triangle counted by } (\# \text{non-}M^* \text{-triangles})\}|.$$

As explained above, this expression is an upper bound on  $(\# \text{component-free})$ . Furthermore, since  $A_1$  includes at most a single edge intersecting every connected component of  $(V, P)$ , the second term in this expression is upper bounded by  $(\# \text{non-}M^* \text{-triangles})$ . Therefore, we get

$$(\# \text{component-free}) \leq 3|A_1| + (\# \text{non-}M^* \text{-triangles}).$$

The lemma now follows by rearranging this inequality. ◀

<sup>4</sup> One of the two  $M^*$  edges intersecting  $C_u$  is guaranteed to be an edge of the triangle itself since the triangle is not counted by  $(\# \text{non-}M^* \text{-triangles})$ . Since such edges cannot be counted by  $(\# \text{component-free})$ , we get that the edge  $(u, v)$  can exclude at most 2 edges of  $(\# \text{component-free})$  rather than 3. However, we ignore this observation as it does not lead to a better approximation guarantee.

The next corollary now follows by combining Lemmata 9 and 10.

► **Corollary 11.**  $2|M^*| \leq 3|A_1| + 2(\#single) + 4(\#double) + 4(\#triangle)$ .

► **Lemma 12.** *It holds that  $4|A_2| \geq (\#component-component) + (\#component-free) - (\#non-M^*-triangles)$ .*

The proof of Lemma 12 is quite similar to the proof of Lemma 10. Therefore, and due to space constrained, we defer it to Appendix C. The next corollary now follows by combining Lemma 12 and the final inequality in Lemma 8.

► **Corollary 13.**  $|M^*| \leq 4|A_2| + 2(\#single) + (\#double) + (\#triangle)$ .

Let us now denote  $L = (\#single) + (\#double) + (\#triangle) + \max\{|A_1|, |A_2|\}$ . We argue below that  $L$  is a lower bound on the size of the solution produced by Algorithm 1. However, before proving this, let us show first that  $L$  is large.

► **Lemma 14.**  $L \geq 7/13|M^*|$ .

**Proof.** Plugging the definition of  $L$  into Corollaries 11 and 13 yields the inequalities  $2|M^*| \leq 3L - (\#single) + (\#double) + (\#triangle)$  and  $|M^*| \leq 4L - 2(\#single) - 3(\#double) - 3(\#triangle)$ . Adding the first of these inequalities three times to the second one gives  $7|M^*| \leq 13L - 5(\#single) \leq 13L$ , where the second inequality holds since  $(\#single)$  is non-negative by definition. The lemma now follows by rearranging the above inequality. ◀

As promised, we now argue that Algorithm 1 produces a matching of size at least  $L$ .

► **Lemma 15.** *Algorithm 1 outputs a matching of size at least  $L$ .*

**Proof.** Since Algorithm 1 outputs a maximum matching in  $(V, P \cup A_1 \cup A_2)$ , to prove the lemma it suffices to show that the graph  $(V, P \cup A_1)$  includes a matching of size  $(\#single) + (\#double) + (\#triangle) + |A_1|$  and the graph  $(V, P \cup A_2)$  includes a matching of size  $(\#single) + (\#double) + (\#triangle) + |A_2|$ . We prove below only the claim regarding  $(V, P \cup A_2)$ . The claim regarding  $(V, P \cup A_1)$  can be proved analogously.

Let  $H$  be the number of edges in  $A_2$  that connect two non-isolated vertices of  $(V, P)$ . Then, we classify the connected components of  $(V, P \cup A_2)$  as follows, and show how to build a large matching  $M$  based on this classification.

- $(V, P \cup A_2)$  includes  $(\#single) + (\#double) + (\#triangle) - |A_2| - H$  connected components that are (i) not an isolated node, and (ii) appear also in  $(V, P)$ . Each one of these connected components contains at least one edge, and therefore, can contribute some edge to  $M$ .
- $(V, P \cup A_2)$  includes  $|A_2| - H$  connected components that consist of a connected component  $C$  of  $(V, P)$  that has connection vertices and an edge  $e$  connecting a connection vertex of  $C$  to an isolated vertex of  $(V, P)$ . One can observe that the combination of  $C$  and  $e$  must be either a path of length 3 or a triangle and an edge attached to one of its vertices, and in both cases this combined connected component contains two vertex disjoint edges which it can contribute to the matching  $M$ .
- $(V, P \cup A_2)$  includes  $H$  connected components that consist of two connected components  $C_1, C_2$  of  $(V, P)$  that have connection vertices and an edge  $e$  connecting a connection vertex of  $C_1$  with a connection vertex of  $C_2$ . There are three shapes that the connected component obtained in this way can take: a path of length 5, a triangle with a path of length 3 attached to one of its vertices or two triangles and an edge connecting them. However, one can observe that all these shapes include three vertex disjoint edges that can be contributed to the matching  $M$ .

### 33:12 Maximum Matching Sans Maximal Matching

By collecting from every connected component of  $(V, P \cup A_2)$  the edges that it can contribute to  $M$  according to the above analysis, we get a matching in  $(V, P \cup A_2)$  of size at least

$$\begin{aligned} & [(\# \text{single}) + (\# \text{double}) + (\# \text{triangle}) - |A_2| - H] + 2[|A_2| - H] + 3H \\ & = (\# \text{single}) + (\# \text{double}) + (\# \text{triangle}) + |A_2|. \end{aligned} \quad \blacktriangleleft$$

Lemmata 14 and 15 imply together the following corollary. Together with Observation 5, this corollary implies Theorem 1.

► **Corollary 16.** *Algorithm 1 is a  $7/13$ -approximation algorithm.*

Before concluding this section, we note that Theorem 2 is proved in the full version of this paper [20] by splitting the second pass of Algorithm 1 into two passes. One pass that constructs  $A_1$ , and a second pass that constructs  $A_2$ , while making sure not to use again connected components of  $(V, P)$  already used by  $A_1$ .

## 4 Three-Pass Algorithm for Triangle-Free Graphs

In this section we prove Theorem 3, which we repeat here for convenience.

► **Theorem 3.** *There exists a 3-pass  $(11/18 = 1/2 + 1/9)$ -approximation semi-streaming algorithm for finding a maximum size matching in a triangle-free graph.*

We refer to the algorithm whose existence is guaranteed by Theorem 3 as TRIANGLEFREEALG. In its first pass, TRIANGLEFREEALG constructs a maximal matching  $M_0$  of  $G$ . Formally, the pseudocode for this pass appears as Algorithm 2.

### Algorithm 2 TRIANGLEFREEALG – FIRST PASS.

- 
- 1 Let  $M_0 \leftarrow \emptyset$ .
  - 2 **for** every edge  $e$  that arrives **do**
  - 3     Add  $e$  to  $M_0$  if it does not intersect any edge that already belongs to  $M_0$ .
- 

We say that an edge  $e \in E$  is a *wing* if  $e$  includes exactly one vertex of  $V(M_0)$ . Intuitively, the reason we are interested in wings is that one can obtain an augmenting path<sup>5</sup> for  $M_0$  by combining an edge  $(u, v) \in M_0$  with two wings: one wing that intersects  $u$  and one wing that intersects  $v$ . The second pass of TRIANGLEFREEALG grows a set  $W$  of wings. Since we hope to construct multiple augmenting paths using these wings, the algorithm makes sure to limit the number of wings in  $W$  that intersect any given vertex  $u$  (specifically, the algorithm allows only a single wing in  $W$  to intersect  $u$  if  $u \in V(M_0)$ , and otherwise it allows up to two wings of  $W$  to intersect  $u$ ). The pseudocode of this second pass appears as Algorithm 3.

Algorithm 3 also includes a post-processing step in which a set  $\mathcal{P}_1$  of augmenting paths (with respect to  $M_0$ ) is constructed using  $W$ . This is done by constructing an auxiliary multi-graph  $G_A$  over the vertices of  $V \setminus V(M_0)$  in which there is an edge between two nodes  $u, v \in V \setminus V(M_0)$  for every path  $P_{u,v}$  of length 3 in  $W \cup M_0$  between them. One can note that every such path  $P_{u,v}$  must be an augmenting path consisting of an edge  $e \in M_0$  and two wings from  $W$ : one intersecting  $u$  and an end-point of  $e$ , and the other intersecting  $v$  and the other end-point of  $e$ . Algorithm 3 finds a maximum size matching  $M_A$  in  $G_A$ , and then sets  $\mathcal{P}_1$  to be the collection of (augmenting) paths corresponding to the edges of  $M_A$ .

---

<sup>5</sup> A path  $P$  is an augmenting path for a matching  $M$  if  $M \oplus E(P)$  is a valid matching of size  $|M| + 1$ .

---

**Algorithm 3** TRIANGLEFREEALG – SECOND PASS.
 

---

```

1 Let $W \leftarrow \emptyset$.
2 for every edge e that arrives do
3 if e intersects exactly one vertex $u \in V(M_0)$ then
4 Let v denote the other end-point of e (i.e., the end-point that is not u).
5 if $\deg_W(u) < 1$ and $\deg_W(v) < 2$ then Add e to W .

// Post-processing
6 Let G_A be a multi-graph over the vertices $V \setminus V(M_0)$. For every path $P_{u,v}$ of length
 3 in $W \cup M_0$ between two vertices $u, v \in V \setminus V(M_0)$, we add an edge (u, v) to the
 graph G_A . // This is a multi-graph because there might be multiple
 such paths between a pair of vertices of $V \setminus V(M_0)$.
7 Find a maximum size matching M_A in G_A .
8 Let $\mathcal{P}_1 \leftarrow \{P_{u,v} \mid (u, v) \in M_A\}$.
```

---

Consider now an edge  $e \in M_0$  that does not appear in any path of  $\mathcal{P}_1$  and is connected by some wing  $w \in W$  to some vertex  $u \notin V(M_0) \cup V(\mathcal{P}_1)$ . The pair  $e, w$  can be extended into an augmenting path if one can find another wing  $w'$  connecting the other end of  $e$  (the end that does not intersect  $w$ ) to a vertex  $v \notin V(M_0) \cup V(\mathcal{P}_1)$  that is not  $u$ . The third pass of TRIANGLEFREEALG greedily constructs a collection  $\mathcal{P}_2$  of augmenting paths in this way. A pseudocode of this pass appears as Algorithm 4. After completing the pass, Algorithm 4 returns the matching obtained by augmenting  $M_0$  with the augmenting paths of  $\mathcal{P}_1$  and  $\mathcal{P}_2$ .

---

**Algorithm 4** TRIANGLEFREEALG – THIRD PASS.
 

---

```

1 Let $\mathcal{P}_2 \leftarrow \emptyset$.
2 for every edge w' that arrives do
3 if there exist 4 vertices $u, a, b, v \in V \setminus (V(\mathcal{P}_1) \cup V(\mathcal{P}_2))$ such that: (i) $u \notin V(M_0)$,
 (ii) $w' = (u, a)$, (iii) $(a, b) \in M_0$ and (iv) $(b, v) \in W$ then
4 Add the path u, a, b, v to \mathcal{P}_2 . // Note that $u \neq v$ because otherwise
 u, a, b, v would have been a triangle.

5 return $M_0 \oplus (\bigcup_{P \in \mathcal{P}_1 \cup \mathcal{P}_2} E(P))$.
```

---

We begin the analysis of TRIANGLEFREEALG with the following lemma, which shows that this algorithm returns a matching, and also gives a basic lower bound on the size of this matching. Due to space constraints, the proof of this lemma is deferred to Appendix C.

► **Lemma 17.** *The paths in  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are vertex disjoint, and therefore, the output of TRIANGLEFREEALG is a matching of size  $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$ .*

Using the last lemma we can also bound the space complexity of Algorithm 4. The technical proof of the next corollary can be found in the full version of this paper [20].

► **Corollary 18.** *TRIANGLEFREEALG is a semi-streaming algorithm.*

It remains to analyze the approximation ratio of TRIANGLEFREEALG. Our analysis roughly follows the flow of the algorithm, and thus, we begin by observing that the matching  $M_0$  constructed in the first pass of this algorithm is of size at least  $|M^*|/2$  (recall that  $M^*$  is a maximum size matching of  $G$ ) because  $M_0$  is a maximal matching of  $G$  by construction.

### 33:14 Maximum Matching Sans Maximal Matching

In its second pass, TRIANGLEFREEALG constructs the set  $W$  of wings. Our next objective is to lower bound the size of  $W$ . Towards this goal, we need to define  $W_M$  to be the set of all edges of  $M^*$  that are wings (we recall that an edge  $e$  is a wing if exactly one of its end points appear in  $V(M_0)$ ).

► **Observation 19.**  $|W_M| \geq 2(|M^*| - |M_0|)$ .

**Proof.** Since  $M_0$  is a maximal matching, every edge of  $M^*$  intersects at least one edge of  $M_0$ . Hence, every edge of  $W_M$  includes a single end-point of an edge of  $M_0$ , and every edge of  $M^* \setminus W_M$  includes two end-points of edges of  $M_0$  (the two end-points might belong to different edges or to the same edge), which implies  $|M_0| \geq (|W_M| + 2|M^* \setminus W_M|)/2 = |M^*| - |W_M|/2$ . Rearranging this inequality completes the proof of the observation. ◀

► **Lemma 20.**  $|W| \geq \frac{2}{3}|W_M| \geq \frac{4}{3}(|M^*| - |M_0|)$ .

**Proof.** Let  $I = V(M_0) \cap V(W_M)$ , and let  $I_F$  be the set of vertices in  $I$  that do not appear in any edge of  $W$ . Every vertex  $a \in I_F \subseteq I$  must belong to some wing  $w(a) \in W_M$  by the definition  $I$ . However, this wing was not added to  $W$  (because  $a \in I_F$ ), which implies that the condition in Line 5 of Algorithm 3 evaluated to FALSE when  $w(a)$  arrived. Since  $a$  is not covered by any edge of  $W$  (i.e.,  $\deg_W(v) = 0$ ), the fact that this condition evaluated to FALSE implies that the end point of  $w(a)$  that does not belong to  $V(M_0)$  must have a degree of 2 under  $W$ . Formally, if we denote by  $u(a)$  the end point of  $w(a)$  that does not belong to  $V(M_0)$ , then we must have  $\deg_W(u(a)) = 2$ .

We now observe that (i) every wing in  $W_M$  contains a disjoint vertex of  $V \setminus V(M_0)$  because  $W_M$  is a subset of the optimal matching  $M^*$ , and (ii) every wing in  $W$  contains only one vertex of  $V \setminus V(M)$  because it is a wing. These two observations imply together

$$|W| \geq \sum_{a \in I_F} \deg_W(u(a)) = 2|I_F| . \quad (5)$$

In contrast, since (i) every wing in  $W$  contains a single vertex of  $V(M_0)$ , and (ii) all the vertices of  $I \setminus I_F \subseteq V(M_0)$  appear in some wing of  $W$ ,

$$|W| \geq |I| - |I_F| = |W_M| - |I_F| , \quad (6)$$

where the equality holds since every edge of  $W_M$  is a wing, and therefore, intersects a single vertex of  $V(M_0)$ . The lemma now follows by adding two copies of Inequality (6) to Inequality (5). ◀

We now get to the analysis of the third pass of TRIANGLEFREEALG, and our first goal in this analysis is to identify a set of paths that have a potential (in some sense) to end up in  $\mathcal{P}_2$ . Let  $\mathcal{P}'$  be the set of paths of length 3 in  $G$  that consist of a wing of  $W_M$  followed by an edge of  $M_0$  and then a wing of  $W$ . We think of the paths in  $\mathcal{P}'$  as directed from their  $W_M$  to their  $W$  edge, and consider two paths that differ only in their direction to be different paths. This is important because if there is an edge  $e \in M_0$  incident to two edges  $w_1, w_2 \in W \cap W_M$ , then the path  $w_1, e, w_2$  fulfills the requirements to belong to  $\mathcal{P}'$  both when  $w_1$  is considered the first edge in it and when  $w_2$  is considered the first edge of the path. Thus, the fact that we treat the direction of the path as part of the path's definition allows both the paths  $w_1, e, w_2$  and  $w_2, e, w_1$  to appear in  $\mathcal{P}'$ .

► **Observation 21.**  $|\mathcal{P}'| \geq \frac{10}{3}|M^*| - \frac{16}{3}|M_0|$ .



**Proof.** Since  $\deg_W(a) \leq 1$  for every vertex  $a \in V(M_0)$ , there are  $|W|$  end-points of  $M_0$  that intersect an edge of  $W$ . Let us denote these end-points by  $V_W$ , and for every end-point  $a \in V_W$ , we denote by  $b(a)$  the other end-point of the same edge of  $M_0$ . Formally,  $V_W = V(M_0) \cap V(W)$ , and  $b(a)$  is the single element of the set  $\{b \mid (a, b) \in M_0\}$ . One can now observe that  $\mathcal{P}'$  includes a (distinct) path for every wing of  $W_M$  that intersect  $b(a)$  for some vertex  $a \in V_W$ . Therefore,

$$\begin{aligned} |\mathcal{P}'| &= |\{b(a) \mid a \in V_W\} \cap V(W_M)| \\ &\geq |\{b(a) \mid a \in V_W\}| + |V(W_M) \cap V(M_0)| - |V(M_0)| = |W| + |W_M| - |V(M_0)| \\ &\geq \frac{4}{3}(|M^*| - |M_0|) + 2(|M^*| - |M_0|) - |V(M_0)| = \frac{10}{3}|M^*| - \frac{16}{3}|M_0|, \end{aligned}$$

where the first equality holds since  $\{b(a) \mid a \in V_W\}$  is a subset of  $V(M_0)$ , and the last inequality follows from Observation 19 and Lemma 20.  $\blacktriangleleft$

A path in  $\mathcal{P}'$  has a potential to be added to  $\mathcal{P}_2$  only if none of its vertices appears in  $\mathcal{P}_1$ . Let  $\mathcal{P}''$  be the set of such paths (formally,  $\mathcal{P}'' = \{P \in \mathcal{P}' \mid V(P) \cap V(\mathcal{P}_1) = \emptyset\}$ ). The following lemma lower bounds the size of  $\mathcal{P}''$ .

► **Lemma 22.**  $|\mathcal{P}''| \geq |\mathcal{P}'| - 6|\mathcal{P}_1| \geq \frac{10}{3}|M^*| - \frac{16}{3}|M_0| - 6|\mathcal{P}_1|$ .

**Proof.** The second inequality of the lemma follows from Observation 21, and therefore, we concentrate on proving the first inequality. Towards this goal, assume that  $P' \in \mathcal{P}'$  is a path that intersects with a path  $P_1 \in \mathcal{P}_1$  on an internal vertex. Since the middle edge of both paths is an edge of  $M_0$ , this implies that the two paths intersect on both their internal vertices. Furthermore, since both end-edges of  $P_1$  and one end-edge of  $P'$  belong to  $W$ , there must be an internal vertex  $a \in V(M_0)$  of both paths that intersects an edge of  $W$  in both paths. However, since  $\deg_W(a) \leq 1$ , the edges of  $W$  intersecting  $a$  in both paths must be identical, which implies that the paths  $P'$  and  $P_1$  intersect also on some end-point. Since  $P'$  and  $P_1$  where chosen as general paths of  $\mathcal{P}'$  and  $\mathcal{P}_1$ , respectively, that intersect on an internal node, this implies that the difference  $|\mathcal{P}'| - |\mathcal{P}''|$  is equal to the number of paths in  $\mathcal{P}'$  that intersect a path of  $\mathcal{P}_1$  in an *end-point*. The rest of the proof is devoted to proving that the last number is at most  $6|\mathcal{P}_1|$ .

Since each path of  $\mathcal{P}_1$  has only two end points, to prove that the paths of  $\mathcal{P}_1$  intersect at most  $6|\mathcal{P}_1|$  paths of  $\mathcal{P}'$  at an end-point, it suffices to show that every vertex of  $V \setminus V(M_0)$  can appear in at most 3 paths of  $\mathcal{P}'$ . To see why that is the case, consider an arbitrary vertex  $u \in V \setminus V(M_0)$ . If  $u$  belongs to some path  $P' \in \mathcal{P}'$ , then it must be in one of two roles as follows.

- If  $u$  is the last vertex of the path, then the last edge of the path is an edge  $e \in W$  that includes  $u$ , and the other edges of the path  $P'$  are the single edge of  $M_0$  intersecting  $e$  and the single edge of  $W_M$  intersecting  $e$ . Note that this means that the identity of the entire path is determined by the edge  $e$ , and therefore, the number of paths of  $\mathcal{P}'$  in which  $u$  is the last vertex can be upper bounded by  $\deg_W(u) \leq 2$ .
- If  $u$  is the first vertex of the path, then the first edge of the path is the single edge  $e \in W_M$  that includes  $u$ , and the other edges of the path are the single edge  $e' \in M_0$  that intersect  $e$  and the single edge  $e'' \in W$  that intersects  $e'$ . Hence, the entire path is determined by the fact that  $u$  is its first vertex, and therefore, there can be only a single path in  $\mathcal{P}'$  in which  $u$  is the first vertex.  $\blacktriangleleft$

Originally, all the paths of  $\mathcal{P}''$  can be picked in the third pass of TRIANGLEFREEALG (Algorithm 4) since they are vertex disjoint from the paths of  $\mathcal{P}_1$ . However, as Algorithm 4 starts to add paths to  $\mathcal{P}_2$ , it stops being possible to add some paths of  $\mathcal{P}''$  to  $\mathcal{P}_2$ . Still, we

can lower bound the size of  $\mathcal{P}_2$  in terms of the size of  $\mathcal{P}''$ . The proof of the next lemma is based on a logic similar to the one used in the previous proof. Thus, we defer this proof to Appendix C.

► **Lemma 23.**  $|\mathcal{P}_2| \geq \frac{1}{6}|\mathcal{P}''| \geq \frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|$ .

► **Corollary 24.** *The size of the output of TRIANGLEFREEALG is  $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| \geq \frac{11}{18}|M^*| = (\frac{1}{2} + \frac{1}{9})|M^*|$ .*

**Proof.** The size of the output of TRIANGLEFREEALG is  $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$  by Lemma 17, thus, we only need to lower bound this sum. To do this, note that

$$\begin{aligned} |M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| &\geq |M_0| + |\mathcal{P}_1| + \{\frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|\} \\ &= \frac{5}{9}|M^*| + \frac{1}{9}|M_0| \geq \frac{5}{9}|M^*| + \frac{1}{18}|M^*| = \frac{11}{18}|M^*|, \end{aligned}$$

where the first inequality follows from Lemma 23, and the second inequality follows from the observation made at the beginning of this section (namely, that  $|M_0|$  is a  $1/2$ -approximation for  $|M^*|$  because  $M_0$  is a maximal matching). ◀

Theorem 3 now follows from Corollaries 18 and 24.

## 5 Conclusion and Future Work

We have presented in this paper a new approach for semi-streaming algorithms for the maximum matching problem, and showed that this approach can be used to improve the state-of-the-art in two and three passes. Our approach calls for a more sophisticated logic in the first pass rather than simply building a maximal matching in a greedy fashion, as is done by previous algorithms. In our implementation of this approach, we greedily built in this pass connected components of size 3 (recall that greedily building a maximal matching is equivalent to greedily building connected components of size 2). Similarly, one can try to greedily construct in the first pass larger connected component, which we believe is likely to yield even better approximation guarantees. However, the analysis of algorithms based on such a first pass is likely to be inelegant, and to require a lot of case analysis since larger components allow many more configurations compared to smaller components. It might also be interesting to try to come up with an interesting algorithm that uses a completely different kind of logic in its first pass.

In addition to the above, we have used in this paper the traditional technique to improve over the state-of-the-art for three passes. Further improving the approximation ratio of two-pass and three-pass algorithms (or proving that this is not possible), is a nice question that is still open. We conclude by recalling that the most basic open question in this field of research is still breaking the (almost trivial)  $1/2$ -approximation for single-pass algorithms. We hope that our new approach will lead to progress on both the above open questions.

---

## References

- 1 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.*, 4(4):17:1–17:40, 2018. doi:10.1145/3154855.
- 2 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.

- 3 Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 19:1–19:13, 2021. doi:10.4230/LIPIcs.ICALP.2021.19.
- 4 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. *arXiv e-prints*, pages arXiv–2011, 2020.
- 5 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742. SIAM, 2017. doi:10.1137/1.9781611974782.113.
- 6 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 354–364. IEEE, 2020. doi:10.1109/FOCS46700.2020.00041.
- 7 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *4th Symposium on Simplicity in Algorithms (SOSA)*, pages 165–171, 2021. doi:10.1137/1.9781611976496.18.
- 8 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021. doi:10.1145/3406325.3451110.
- 9 Michel L. Balinski and Jaime Gonzalez. Maximum matchings in bipartite graphs via strong spanning trees. *Networks*, 21(2):165–179, 1991. doi:10.1002/net.3230210203.
- 10 Aaron Bernstein. Improved bounds for matching in random-order streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.12.
- 11 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 668–681. ACM, 2021. doi:10.1145/3406325.3451113.
- 12 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016. doi:10.1137/1.9781611974331.ch92.
- 13 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15, 2017. doi:10.4230/LIPIcs.ESA.2017.29.
- 14 Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 28 of *LIPIcs*, pages 96–104, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.
- 15 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.
- 16 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discret. Math.*, 25(3):1251–1265, 2011. doi:10.1137/100801901.

- 17 Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Trans. Algorithms*, 14(4):48:1–48:23, 2018. doi:10.1145/3230819.
- 18 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. doi:10.1109/ICDMW.2016.0092.
- 19 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.
- 20 Moran Feldman and Ariel Szarf. Maximum matching sans maximal matching: A new approach for finding maximum matchings in the data stream model. *CoRR*, abs/2109.05946, 2021. arXiv:2109.05946.
- 21 Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic  $(1+\epsilon)$ -approximate maximum matching with  $\text{poly}(1/\epsilon)$  passes in the semi-streaming model and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 248–260. ACM, 2022. doi:10.1145/3519935.3520039.
- 22 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012. doi:10.1137/1.9781611973099.41.
- 23 John E. Hopcroft and Richard M. Karp. An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.
- 24 Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.15.
- 25 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013. doi:10.1137/1.9781611973105.121.
- 26 Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893, 2021. doi:10.1137/1.9781611976465.112.
- 27 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014. doi:10.1137/1.9781611973402.55.
- 28 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117 of *LIPIcs*, pages 74:1–74:16, 2018. doi:10.4230/LIPIcs.MFCS.2018.74.
- 29 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242, 2012. doi:10.1007/978-3-642-32512-0\_20.

- 30 Christian Konrad and Kheeran K. Naidu. On two-pass streaming algorithms for maximum bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 19:1–19:18, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.19.
- 31 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and 9th International Workshop on Randomization and Computation (RANDOM)*, pages 170–181, 2005. doi:10.1007/11538462\_15.
- 32 Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 17:1–17:12, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.17.
- 33 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *1st Symposium on Simplicity in Algorithms (SOSA)*, volume 61 of *OASICS*, pages 14:1–14:4, 2018. doi:10.4230/OASICS.SOSA.2018.14.
- 34 Ami Paz and Gregory Schwartzman. A  $(2 + \epsilon)$ -approximation for maximum weight matching in the semi-streaming model. *ACM Trans. Algorithms*, 15(2):18:1–18:15, 2019. doi:10.1145/3274668.
- 35 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012. doi:10.1007/s00453-010-9438-5.

## A Three-Pass Algorithm for General Graphs

In this section we prove Theorem 4, which we repeat here for convenience.

► **Theorem 4.** *There exists a 3-pass  $(1/2 + 1/14.4)$ -approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithm that we use to prove Theorem 4 is given as Algorithm 5. Since this algorithm is very similar to the algorithm TRIANGLEFREEALG presented in Section 4, we use below the terminology and notation defined in the last section.

Intuitively, the reason why TRIANGLEFREEALG does not apply to general graphs is that given an edge  $(a, b) \in M_0$ , a wing  $(u, a) \in W_M$  and a wing  $(b, v) \in W$ , we are not guaranteed that these three edges form an augmenting path for the matching  $M_0$  because they might represent a triangle. To overcome this hurdle, Algorithm 5 constructs two sets of edges in its second pass: a set  $W_1$  constructed exactly like the set  $W$  in TRIANGLEFREEALG, and a set  $W_2$  constructed in the same way, but while excluding the edges of  $W_1$ . Since  $W_1$  and  $W_2$  are disjoint, given an edge  $(a, b) \in M_0$  and a wing  $(u, a) \in W_M$ , at most one of the sets  $W_1$  or  $W_2$  can contain a wing that forms a triangle together with these two edges, which intuitively allows us to bound the deterioration in the approximation guarantee resulting from the existence of such triangles.

We note that the analysis of TRIANGLEFREEALG up to Lemma 20 applies to Algorithm 5 with a single modification. Namely, Lemma 20 provides a lower bound on the size of the set  $W$ , which translates into an identical lower bound on the size of the corresponding set  $W_1$  in Algorithm 5.

In the rest of this section, it will be convenient to work with the set  $W'_2$  constructed by Algorithm 6 (note that Algorithm 6 is used for analysis purposes only). Intuitively,  $W'_2$  is constructed in the same general way in which  $W_1$  and  $W_2$  are constructed; however, while all the edges of the input stream are considered in the construction of  $W_1$ , and only the edges of  $E \setminus W_1$  are considered in the construction of  $W_2$ , the construction of  $W'_2$  takes into account the edges of  $(E \setminus W_1) \cup W_M$ .

---

**Algorithm 5** MAXIMUM MATCHING VIA AUGMENTING PATHS – GENERAL GRAPHS.

---

```

// First Pass
1 Let $M_0 \leftarrow \emptyset$.
2 for every edge e that arrives do
3 | Add e to M_0 if it does not intersect any edge that already belongs to M_0 .

// Second Pass
4 Let $W_1 \leftarrow \emptyset$, $W_2 \leftarrow \emptyset$.
5 for every edge e that arrives do
6 | if e intersects exactly one vertex $u \in V(M_0)$ then
7 | Let v denote the other end-point of e (i.e., the end-point that is not u).
8 | if $\deg_{W_1}(u) < 1$ and $\deg_{W_1}(v) < 2$ then Add e to W_1 .
9 | else if $\deg_{W_2}(u) < 1$ and $\deg_{W_2}(v) < 2$ then Add e to W_2 .

// Post-processing
10 Let G_A be a multi-graph over the vertices $V \setminus V(M_0)$. For every path $P_{u,v}$ of length 3
 in $W_1 \cup W_2 \cup M_0$ between two distinct vertices $u, v \in V \setminus V(M_0)$, we add an edge
 (u, v) to the graph G_A . // This is a multi-graph because there might be
 multiple such paths between a pair of vertices of $V \setminus V(M_0)$.
11 Find a maximum size matching M_A in G_A .
12 Let $\mathcal{P}_1 \leftarrow \{P_{u,v} \mid (u, v) \in M_A\}$.

// Third Pass
13 Let $\mathcal{P}_2 \leftarrow \emptyset$.
14 for every edge w' that arrives do
15 | if there exist 4 vertices $u, a, b, v \in V \setminus (V(\mathcal{P}_1) \cup V(\mathcal{P}_2))$ such that: (i) $u \notin V(M_0)$,
 (ii) $w' = (u, a)$, (iii) $(a, b) \in M_0$, (iv) $(b, v) \in W_1 \cup W_2$ and (v) $u \neq v$ then
16 | | Add the path u, a, b, v to \mathcal{P}_2 .

17 return $M_0 \oplus (\bigcup_{P \in \mathcal{P}_1 \cup \mathcal{P}_2} E(P))$.

```

---

Since  $W'_2$  is a subset of  $W_1 \cup W_2$  by construction, the set  $W_1 \cup W_2$  that is often referred to by Algorithm 5 is identical to the set  $W_1 \cup W'_2$ . Furthermore, one can observe that the lower bound proved by Lemma 20 for  $W_1$  applies also to  $W'_2$  because all the edges of  $W_M$  are considered for addition to  $W'_2$  at some point (either during the construction of  $W_2$  or in Algorithm 6). This implies the following observation.

► **Observation 25.**  $|W_1| + |W'_2| \geq \frac{4}{3}|W_M|$ .

We now define a multi-set  $\mathcal{P}'$  similar to the set of the same name used in the analysis of TRIANGLEFREEALG. Specifically,  $\mathcal{P}'$  includes every triangle or path obtained by combining an edge  $(u, a) \in W_M$ , an edge  $(a, b) \in M_0$  and an edge  $(b, v)$  of either  $W_1$  or  $W'_2$ . Moreover, if there are multiple options to obtain a path or triangle in this way, then the multiplicity of the path or triangle in  $\mathcal{P}'$  will be equal to the number of these options. To make this point clearer, we provide a pseudocode for constructing  $\mathcal{P}'$  as Algorithm 7 (again, Algorithm 7 is used for analysis purposes only).

► **Observation 26.**  $|\mathcal{P}'| \geq \frac{20}{3}|M^*| - \frac{32}{3}|M_0|$ .



---

**Algorithm 6** Construction of  $W'_2$ .

---

```

1 Let $W'_2 \leftarrow W_2$.
2 for every edge $(u, v) \in W_1 \cap W_M$ do
3 Assume without loss of generality that u is the end point of (u, v) that belongs to
 $V(M_0)$.
4 if $\deg_{W'_2}(u) < 1$ and $\deg_{W'_2}(v) < 2$ then Add (u, v) to W'_2 .
```

---



---

**Algorithm 7** Construction of  $\mathcal{P}'$ .

---

```

1 Let $\mathcal{P}' \leftarrow \emptyset$.
2 for every edge $(u, a) \in W_M$ do
3 for every edge $(a, b) \in M_0$ do
4 for every edge $(b, v) \in W_1$ do Add the path/triangle $(u, a), (a, b), (b, v)$ to \mathcal{P}' .
5 for every edge $(b, v) \in W'_2$ do Add the path/triangle $(u, a), (a, b), (b, v)$ to \mathcal{P}' .
```

---

**Proof.** Repeating the proof of Observation 21, we get that at least  $|W_1| + |W_M| - |V(M_0)|$  paths are added to  $\mathcal{P}'$  in Line 4 of Algorithm 7, and at least  $|W'_2| + |W_M| - |V(M_0)|$  paths are added to  $\mathcal{P}'$  in Line 5 of Algorithm 7. Therefore,

$$\begin{aligned}
|\mathcal{P}'| &\geq |W_1| + |W'_2| + 2|W_M| - 2|V(M_0)| \geq \left(\frac{4}{3} + 2\right)|W_M| - 2|V(M_0)| \\
&\geq 2\left(\frac{4}{3} + 2\right)(|M^*| - |M_0|) - 2|V(M_0)| = \frac{20}{3}|M^*| - \frac{32}{3}|M_0|,
\end{aligned}$$

where the second inequality follows from Observation 25, and the last inequality follows from Observation 19.  $\blacktriangleleft$

An element (path or triangle) of  $\mathcal{P}'$  has a potential to be added to  $\mathcal{P}_2$  by Algorithm 5 only if it is a path (i.e., not a triangle) and none of its vertices appears in  $\mathcal{P}_1$ . Let  $\mathcal{P}''$  be the multi-set of such paths. The following lemma lower bounds the size of  $\mathcal{P}''$ .

► **Lemma 27.**  $|\mathcal{P}''| \geq |\mathcal{P}'| - 12|\mathcal{P}_1| - |M_0| \geq \frac{20}{3}|M^*| - \frac{35}{3}|M_0| - 12|\mathcal{P}_1|$ .

**Proof.** The second inequality of the lemma follows from Observation 26, and therefore, we concentrate on proving the first inequality. Let  $\tilde{\mathcal{P}}'$  be the multi-set of paths/triangles from  $\mathcal{P}'$  that do not intersect any vertex of  $\mathcal{P}_1$ . Repeating the proof of Lemma 22, we get that  $\tilde{\mathcal{P}}'$  contains all the paths/triangles added to  $\mathcal{P}'$  by Line 4 of Algorithm 7 except for up to  $6|\mathcal{P}_1|$  paths/triangles, and the same is true for the paths/triangles added to  $\mathcal{P}'$  by Line 5 of Algorithm 7. Since every path/triangle in  $\mathcal{P}'$  was added to this multi-set by either Line 4 or Line 5 of Algorithm 7, we get  $|\tilde{\mathcal{P}}'| \geq |\mathcal{P}'| - 12|\mathcal{P}_1|$ .

Since  $\mathcal{P}''$  includes every path of  $\tilde{\mathcal{P}}'$ , to complete the proof of the lemma it remains to show that  $\tilde{\mathcal{P}}'$  contains at most  $|M_0|$  triangles. To see this, we recall that every triangle (or path) in  $\tilde{\mathcal{P}}'$  must include a single edge of  $M_0$ , and we claim that no two triangles in  $\tilde{\mathcal{P}}'$  can share this edge (and therefore, the number of triangles is upper bounded by the number of edges in  $M_0$ ). Assume towards a contradiction that this claim does not hold, i.e., that there exist two triangles  $T_1, T_2 \in \tilde{\mathcal{P}}'$  sharing an edge  $e \in M_0$ . Each one of these triangles must include one edge of  $W_M$ . Let  $e_1$  and  $e_2$  denote the edges of  $W_M$  in  $T_1$  and  $T_2$ , respectively, and let  $e'_1$  the single edge of  $T_1$  which is not  $e$  or  $e_1$  and  $e'_2$  be the single edge of  $T_2$  which is not either  $e$  or  $e_2$ . We now need to consider two cases. The first case is when  $e_1 = e_2$ . In this case  $e'_1$  and  $e'_2$  must be also identical, and cannot belong to  $W_M$  because  $e_1 = e_2$  belongs to



### 33:22 Maximum Matching Sans Maximal Matching

$W_M$  and  $W_M$  is a subset of the matching  $M^*$ . However, this leads to a contradiction because one of the edges  $e'_1$  or  $e'_2$  must belong to  $W_1$ , and the other of these edges must belong to  $W'_2$ , and the sets  $W_1$  and  $W'_2$  can intersect only on edges of  $W_M$ .

It remains to consider the case in which  $e_1 \neq e_2$ . Let  $u_1, u_2$  be the end-points of these edges, respectively, that do not belong to the edge  $e$  of  $M_0$ . Since  $e_1 \neq e_2$  are edges of the  $W_M$ , which is a subset of the matching  $M^*$ ,  $u_1$  and  $u_2$  must be distinct. Consider now the path  $e'_1, e, e'_2$ . One can observe that this is indeed a path because (i)  $u_1 \neq u_2$  and (ii) the fact that  $e_1$  and  $e_2$  are vertex disjoint implies that  $e'_1$  and  $e'_2$  intersect different end-points of  $e$ . Furthermore, since  $T_1, T_2 \in \tilde{\mathcal{P}}'$ , this path does not intersect any vertex of  $\mathcal{P}_1$ , and thus, its existence contradicts the maximality of the matching  $M_A$  constructed by Algorithm 5 because both  $e'_1$  and  $e'_2$  belong to  $W_1 \cup W'_2 = W_1 \cup W_2$ . ◀

We are now ready to lower bound the number of augmenting paths found by Algorithm 5 during its third pass.

► **Lemma 28.**  $|P_2| \geq |\mathcal{P}''|/12 \geq \frac{5}{9}|M^*| - \frac{35}{36}|M_0| - |\mathcal{P}_1|$ .

**Proof.** The proof of the lemma is very similar to the proof of Lemma 23, except that now every path of  $\mathcal{P}_2$  might get a charge of up to 12 because the paths of  $\mathcal{P}''$  originally added to  $\mathcal{P}'$  by Line 4 of Algorithm 7 can contribute up to 6 to this charge, and the same goes for the paths of  $\mathcal{P}''$  originally added to  $\mathcal{P}'$  by Line 5 of this algorithm. ◀

Theorem 4 now follows from Corollary 18 and the next corollary.

► **Corollary 29.** *The size of the matching produced by Algorithm 5 is at least  $(\frac{1}{2} + \frac{1}{14.4})|M^*|$ .*

**Proof.** By Lemma 17, the size of the matching produced by Algorithm 5 is at least

$$|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| \geq \frac{5}{9}|M^*| + \frac{1}{36}|M_0| \geq \frac{5}{9}|M^*| + \frac{1}{72}|M^*| = (\frac{1}{2} + \frac{1}{14.4})|M^*| ,$$

where the first inequality holds by Lemma 28, and the second inequality holds since  $M_0$  (as a maximal matching) is of size at least  $\frac{1}{2}|M^*|$ . ◀

## B Notation Summary

The next table summarizes the notation used in the analyses of our non-MMF algorithms.

| Notation               | Explanation                                                                                                                                                                                   |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (#single)              | The number of isolated edges in $(V, P)$ .                                                                                                                                                    |
| (#double)              | The number of connected components in $(V, P)$ that are paths of length 2.                                                                                                                    |
| (#triangle)            | The number of triangles in $(V, P)$ .                                                                                                                                                         |
| (#component-free)      | The number of edges of $M^*$ that connect a connection vertex of a connected component of $(V, P)$ to an isolated vertex of $(V, P)$ .                                                        |
| (#component-component) | The number of edges of $M^*$ that connect connection vertices of two different connected components of $(V, P)$ .                                                                             |
| (#single-single)       | The number of edges of $M^*$ whose two end points belong to (not necessarily distinct) isolated edges of $(V, P)$ .                                                                           |
| (#single-component)    | The number of edges of $M^*$ that either connect a vertex of an isolated edge of $(V, P)$ with a connection vertex of some (other) connected component of $(V, P)$ .                          |
| (#middle)              | The number of edges that either (i) intersect the middle vertex of a length 2 path connected component of $(V, P)$ , or (ii) are included within a triangle connected component of $(V, P)$ . |

## C

 Omitted Proofs

► **Lemma 12.** *It holds that  $4|A_2| \geq (\# \text{component-component}) + (\# \text{component-free}) - (\# \text{non-}M^* \text{-triangles})$ .*

**Proof.** The proof of Lemma 12 is very similar to the proof of Lemma 10, and therefore, we only sketch it. We first define that an edge  $e \in A_2$  excludes an edge  $f$  of  $M^*$  counted by either  $(\# \text{component-component})$  or  $(\# \text{component-free})$  if they both intersect the same connected component of  $(V, P)$ . Like in the proof of Lemma 10, it can be argued that  $(\# \text{component-component}) + (\# \text{component-free})$  is upper bounded by the total number of edges of  $M^*$  excluded by the edges of  $A_2$ , and on the other hand, every edge  $e$  of  $A_2$  excludes up to  $4 + T(e)$  edges, where  $T(e)$  is the number of triangles counted by  $(\# \text{non-}M^* \text{-triangles})$  that intersect  $e$ . Therefore,

$$\begin{aligned} (\# \text{component-component}) + (\# \text{component-free}) &\leq \sum_{e \in A_2} [4 + T(e)] \\ &\leq 4|A_2| + (\# \text{non-}M^* \text{-triangles}) , \end{aligned}$$

where the second inequality holds since every connected component of  $(V, P)$  intersects only a single edge of  $A_2$ . The lemma now follows by rearranging the last inequality. ◀

► **Lemma 17.** *The paths in  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are vertex disjoint, and therefore, the output of TRIANGLEFREEALG is a matching of size  $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$ .*

**Proof.** Given the above discussion, it is clear that all the paths in  $\mathcal{P}_1 \cup \mathcal{P}_2$  are augmentation paths with respect to  $M_0$ , which implies that the first part of the lemma indeed implies the second part. Furthermore, one can observe that the condition in Line 3 of Algorithm 4 guarantees that the paths in  $\mathcal{P}_2$  are vertex disjoint from each other and from the paths of  $\mathcal{P}_1$ . Thus, to complete the proof of the lemma, it remains to argue that the paths in  $\mathcal{P}_1$  are also vertex disjoint.

Recall that the end-points of every path in  $\mathcal{P}_1$  belong to  $V \setminus V(M_0)$  and the internal points of these paths belong to  $V(M_0)$ . Hence, to show that the paths in  $\mathcal{P}_1$  are vertex disjoint, it suffices to argue this separately for their end-points and their internal nodes. Every path  $P_{u,v} \in \mathcal{P}_1$  corresponds to an edge  $(u, v)$  in the matching  $M_A$ . Since the end-points of the path  $P_{u,v}$  are also the end-points of this edge, we get that the paths in  $\mathcal{P}_1$  must have disjoint end-points because  $M_A$  is a matching. Consider now some path  $P_{u,v} \in \mathcal{P}_1$ , and let us denote the internal nodes of this path by  $a$  and  $b$ . Since  $a$  and  $b$  appear only in the edge  $(a, b)$  of  $M_0$  (because  $M_0$  is a matching), we get that if one of them belongs to a path of  $\mathcal{P}_1$ , then the other belongs to this path as well. Furthermore, by Line 5 of Algorithm 3,  $\deg_W(a) = \deg_W(b) = 1$ , which implies that any path of  $\mathcal{P}_1$  that includes the nodes  $a$  and  $b$  as internal nodes must in fact be identical to  $P_{u,v}$  itself. Hence, no two paths in  $\mathcal{P}_1$  share internal nodes. ◀

► **Lemma 23.**  $|\mathcal{P}_2| \geq \frac{1}{6}|\mathcal{P}''| \geq \frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|$ .

**Proof.** We begin the proof by observing that no edge  $e \in M_0$  is connect by two distinct wings  $w_1, w_2 \in W$  to vertices of  $V \setminus (V(M_0) \cup V(\mathcal{P}_1))$ . Assume towards a contradiction that this is not true, then there is an edge  $e$  in  $G_A$  corresponds to the path  $P$  defined as  $w_1, e, w_2$ . Since  $M_A$  is a maximum matching in  $G_A$ , it must include at least one edge that contains some end-point of  $P$  (otherwise, the edge corresponding to  $P$  could be added to  $M_A$ , which violates its maximality); which contradicts the definition of either  $w_1$  or  $w_2$ .

For every path  $P'' \in \mathcal{P}''$ , let us charge a cost of 1 to some path of  $\mathcal{P}_2$  that intersects it. To see why such a path must exist, let us denote by  $e_M$  the edge of  $P''$  that belongs to  $W_M$  (the first edge of  $P''$ ). When  $e_M$  arrives, the path  $P''$  was one candidate to be added

to  $\mathcal{P}_2$  by Algorithm 4. If this candidate was still feasible at this time (in the sense that it was vertex disjoint from  $\mathcal{P}_2$ ), then Algorithm 4 must have added either  $P''$  to  $\mathcal{P}_2$  or another path that includes  $e_M$ . In either case, following the arrival of  $e_M$ , some path intersecting  $P''$  (which is possibly  $P''$  itself) appears in  $\mathcal{P}_2$  – and can be charged.

Our next goal is to show that the total cost charged to any single path of  $\mathcal{P}_2$  is at most 6, which implies the lemma because the total cost charged to all the paths of  $\mathcal{P}_2$  is exactly  $|\mathcal{P}''|$ . We do that by making two observations.

- Since  $\mathcal{P}'' \subseteq \mathcal{P}'$ , we get by the proof of Lemma 22 that at most 3 paths of  $\mathcal{P}''$  can include any given vertex  $u \in V \setminus V(M_0)$ .
- Our second observation is that, if a path  $P'' \in \mathcal{P}''$  intersects a path  $P_2 \in \mathcal{P}_2$ , then they must intersect on an end-point of  $P_2$ . Assume towards a contradiction that they only intersect on an internal node  $a$ . Since the middle edges of both paths are edges of  $M_0$  that include  $a$ , both internal edges must be the same. Let us denote this internal edge by  $e$ . Furthermore, as explained above, there can be only a single edge  $w \in W$  that intersects  $e$  and does not include a vertex of  $V(\mathcal{P}_1)$ . This edge must belong also to both paths, and therefore, the end-point of  $w$  that does not belong to  $V(M_0)$  is an end-point of both  $P''$  and  $P_2$ .

Combining the above two observations, we get that, for every path  $P_2 \in \mathcal{P}_2$ , only paths of  $\mathcal{P}''$  intersecting an end-point of  $P_2$  can charge a cost to  $P_2$ , and there can be at most 3 paths of  $\mathcal{P}''$  intersecting each such end-point. Since  $P_2$  has only two end-points, this implies that at most 6 paths of  $\mathcal{P}''$  can charge  $P_2$ . ◀

# Ordered $k$ -Median with Outliers

Shichuan Deng ✉

IIIS, Tsinghua University, Beijing, China

Qianfan Zhang ✉

IIIS, Tsinghua University, Beijing, China

---

## Abstract

We study a natural generalization of the celebrated ordered  $k$ -median problem, named *robust ordered  $k$ -median*, also known as ordered  $k$ -median with outliers. We are given facilities  $\mathcal{F}$  and clients  $\mathcal{C}$  in a metric space  $(\mathcal{F} \cup \mathcal{C}, d)$ , parameters  $k, m \in \mathbb{Z}_+$  and a non-increasing non-negative vector  $\mathbf{w} \in \mathbb{R}_+^m$ . We seek to open  $k$  facilities  $F \subseteq \mathcal{F}$  and serve  $m$  clients  $C \subseteq \mathcal{C}$ , inducing a *service cost vector*  $\mathbf{c} = \{d(j, F) : j \in C\}$ ; the goal is to minimize the *ordered objective*  $\mathbf{w}^\top \mathbf{c}^\downarrow$ , where  $d(j, F) = \min_{i \in F} d(j, i)$  is the minimum distance between client  $j$  and facilities in  $F$ , and  $\mathbf{c}^\downarrow \in \mathbb{R}_+^m$  is the non-increasingly sorted version of  $\mathbf{c}$ . Robust ordered  $k$ -median captures many interesting clustering problems recently studied in the literature, e.g., robust  $k$ -median, ordered  $k$ -median, etc.

We obtain the *first* polynomial-time constant-factor approximation algorithm for robust ordered  $k$ -median, achieving an approximation guarantee of 127. The main difficulty comes from the presence of outliers, which already causes an unbounded integrality gap in the natural LP relaxation for robust  $k$ -median. This appears to invalidate previous methods in approximating the highly non-linear ordered objective. To overcome this issue, we introduce a novel yet very simple reduction framework that enables linear analysis of the non-linear objective. We also devise the first constant-factor approximations for *ordered matroid median* and *ordered knapsack median* using the same framework, and the approximation factors are 19.8 and 41.6, respectively.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering

**Keywords and phrases** clustering, approximation algorithm, design and analysis of algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.34

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/pdf/2011.04289.pdf>

## 1 Introduction

$k$ -supplier and  $k$ -median are two of the most fundamental clustering problems. In both problems, we are given facilities  $\mathcal{F}$  and clients  $\mathcal{C}$  in a metric space  $(\mathcal{F} \cup \mathcal{C}, d)$  and a parameter  $k \in \mathbb{Z}_+$ ; we need to select  $k$  facilities  $F \subseteq \mathcal{F}$ , and only the objective functions are different. In  $k$ -supplier, the goal is to minimize the maximum distance from each client to its nearest facility in  $F$ , i.e.,  $\max_{j \in \mathcal{C}} d(j, F)$ ;  $k$ -supplier is NP-hard to approximate to a factor better than 3 [18], and a tight 3-approximation is given in [18]. In  $k$ -median, the objective is the sum of distances from each client to its nearest facility, i.e.,  $\sum_{j \in \mathcal{C}} d(j, F)$ ;  $k$ -median is NP-hard to approximate to a factor of  $(1 + 2/e - \epsilon)$  for every  $\epsilon > 0$  [20], and several constant-factor approximations are developed [2, 9, 11, 14, 21, 26]; currently the best approximation guarantee is  $(2.675 + \epsilon)$  due to Byrka et al. [4].

Under the basic input  $(\mathcal{F}, \mathcal{C}, d, k)$ , let  $\mathbf{c}_0 = \{d(j, F) : j \in \mathcal{C}\}$  be the service cost vector induced by solution  $F$ . The theoretical computer science community has lately shown increasingly more interests in clustering problems with more nuanced objective functions than  $k$ -supplier and  $k$ -median. For example, the ordered  $k$ -median problem (OkMED) naturally unifies these two problems via the ordered objective  $\mathbf{w}_0^\top \mathbf{c}_0^\downarrow$ , where  $\mathbf{c}_0^\downarrow$  is the non-increasingly sorted version of  $\mathbf{c}_0$  and  $\mathbf{w}_0$  is a given non-increasing non-negative vector; it is easy to



© Shichuan Deng and Qianfan Zhang;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 34; pp. 34:1–34:22



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

see that  $OkMED$  recovers  $k$ -supplier and  $k$ -median using only 0-1 vectors for  $\mathbf{w}_0$ . Several constant-factor approximations have been developed for  $OkMED$  [5, 7], and currently the best ratio is  $(5 + \epsilon)$  due to Chakrabarty and Swamy [8].

Meanwhile, a parallel line of research called robust clustering (also known as clustering with outliers) also attracts a lot of attention. These problems allow us to discard a certain number of clients and define the clustering objective on the remaining clients. In *robust  $k$ -center* ( $RkCEN$ ), we are given an additional integer parameter  $m \leq |\mathcal{C}|$  besides the basic input  $(\mathcal{F}, \mathcal{C}, d, k)$  where  $\mathcal{F} = \mathcal{C}$ ; we need to open  $k$  facilities  $F \subseteq \mathcal{F}$  and choose  $m$  clients  $C \subseteq \mathcal{C}$ , and the objective is the maximum service cost in  $C$ , i.e.,  $\max_{j \in C} d(j, F)$ . Charikar et al. [10] give a 3-approximation algorithm for  $RkCEN$ . Chakrabarty et al. [6] improve the result to a best-possible 2-approximation (also see Harris et al. [17]). It is easy to see that the objective of  $RkCEN$  is equivalent to  $\mathbf{e}_{|\mathcal{C}|-m+1}^\top \mathbf{c}_0^\downarrow$ , where  $\mathbf{e}_t = \{0, \dots, 0, 1, 0, \dots, 0\}$  is the all-zero vector except for its  $t$ -th coordinate, which is 1. In *robust  $k$ -median* ( $RkMED$ ), the input is the same as  $RkCEN$  except that  $\mathcal{C}$  and  $\mathcal{F}$  are distinct, and the objective is the sum of service costs in  $C$ , i.e.,  $\sum_{j \in C} d(j, F)$ . Chen [13] gives the first constant-factor approximation for  $RkMED$ . Krishnaswamy et al. [23] employ an iterative rounding method and obtain an approximation ratio of  $(7.081 + \epsilon)$  for  $RkMED$ , which is later improved to  $(6.994 + \epsilon)$  by Gupta et al. [16]. The objective of  $RkMED$  is equivalent to  $\boldsymbol{\sigma}_{|\mathcal{C}|-m+1}^\top \mathbf{c}_0^\downarrow$ , where  $\boldsymbol{\sigma}_t = \{0, \dots, 0, 1, \dots, 1\}$  is the all-one vector except for its first  $(t - 1)$  coordinates, which are 0's.

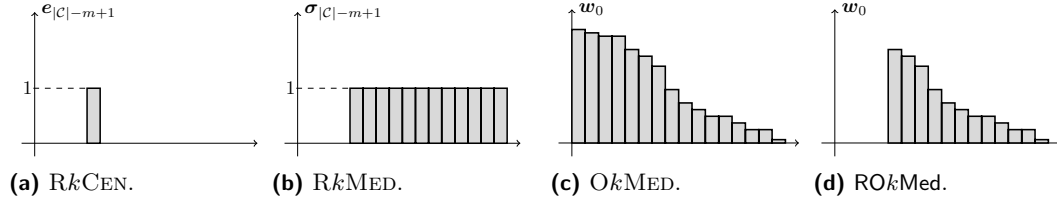
In this paper, we study a new problem called robust ordered  $k$ -median ( $ROkMed$ ). Formally, given the basic input  $(\mathcal{F}, \mathcal{C}, d, k)$ , a parameter  $m \leq |\mathcal{C}|$  and a non-increasing non-negative vector  $\mathbf{w} \in \mathbb{R}_+^m$ , we are asked to open  $k$  facilities  $F \subseteq \mathcal{F}$  and serve  $m$  clients  $C \subseteq \mathcal{C}$ , inducing a service cost vector  $\mathbf{c} = \{d(j, F) : j \in C\} \in \mathbb{R}_+^{\mathcal{C}}$  (notice that  $\mathbf{c}$  is different from  $\mathbf{c}_0$ , since  $\mathbf{c}_0$  is indexed by  $\mathcal{C}$ ); the goal is to minimize  $\mathbf{w}^\top \mathbf{c}^\downarrow$ . Clearly,  $ROkMed$  unifies the aforementioned problems of  $OkMED$ ,  $RkCEN$  and  $RkMED$  by choosing  $m$  and  $\mathbf{w}$  suitably.

We can also define the objective of  $ROkMed$  using  $\mathbf{c}_0 = \{d(j, F) : j \in \mathcal{C}\}$  as follows. Let  $\mathbf{w}_0 \in \mathbb{R}_+^{|\mathcal{C}|}$  be a non-negative vector, such that its first  $(|\mathcal{C}| - m)$  coordinates are 0's, and the remaining coordinates are non-increasing; the objective of  $ROkMed$  is  $\mathbf{w}_0^\top \mathbf{c}_0^\downarrow$ . We notice that this weight vector  $\mathbf{w}_0$  exhibits a distinctly *unimodal* shape; that is, there exists an index  $t$  (which is  $(|\mathcal{C}| - m + 1)$  here) such that  $\mathbf{w}_0$  is non-decreasing on indexes  $\{1, \dots, t\}$  and non-increasing on indexes  $\{t, \dots, |\mathcal{C}|\}$  (see Figure 1 for an example). Therefore, this objective function places a heavier emphasis on clients that are close to the “mode” of  $\mathbf{w}_0$ . This can also be motivated by the following real-world scenario. Suppose the underlying metric  $d$  models the latencies of an online streaming service in different regions (i.e., clients), where the facilities represent potential data center locations. From a business point of view, one could strategically disregard clients that have very poor latencies (they might stop using the service anyway) and clients that have very good latencies (they typically have relatively few issues or complaints); the majority of maintenance and servicing costs will then come from clients with medium latencies. By choosing  $\mathbf{w}_0$  properly, the objective of  $ROkMed$  can be the sum of *sorted* service costs, say, between the 35th percentile and the 65th percentile, thus acting as a good optimization objective for this scenario. We believe this motivating example for  $ROkMed$  offers a practical clustering criterion, and our results will stimulate more studies towards clustering objectives with *arbitrary* unimodal weight vectors.

## 1.1 Our Contributions

We first study robust ordered  $k$ -median and obtain the following main result of this paper.

► **Theorem 1.** *There exists a polynomial-time 127-approximation algorithm for  $ROkMed$ .*



**Figure 1** An illustration of different weight vectors used in the objectives of RkCEN, RkMED, OkMED and ROkMed. The coordinates are represented using unit-width rectangles.

At a high level, we build a simple reduction framework that reduces each ROkMed instance  $\mathcal{J}$  to an instance  $\mathcal{J}'$  of a new problem; the objective of  $\mathcal{J}'$  is still non-linear, but is formulated as a simple sum and easier to approximate. Moreover, by (approximately) solving the new problem, we show that the approximation guarantee of the solution in  $\mathcal{J}$  is preserved up to a constant factor in  $\mathcal{J}'$  (see Theorem 3 for the formal statement). Thus, it suffices to obtain a constant-factor approximate solution for each new instance  $\mathcal{J}'$ . To this end, we adapt the iterative rounding algorithm by Krishnaswamy et al. [23]. We note that this rounding algorithm is only applicable to the non-linear objective of  $\mathcal{J}$  thanks to our *parameterized* reduction framework. Though Gupta et al. [16] give a slightly improved iterative rounding algorithm, we do not adapt their algorithm here. We choose the original algorithm in [23] for its simplicity of presentation. The improvement based on [16] is likely to be small due to our different metric discretization method.

We extend our results to ordered matroid median (OMatMed) and ordered knapsack median (OKnapMed), which are natural generalizations of OkMED by replacing the cardinality constraint  $|F| \leq k$  with a matroid constraint and a knapsack constraint, respectively (see Section 3.2 for the formal definitions). To the best of our knowledge, no approximation algorithms are known for OMatMed and OKnapMed prior to our study.

► **Theorem 2.** *There exist a polynomial-time 19.8-approximation algorithm for OMatMed and a polynomial-time 41.6-approximation algorithm for OKnapMed.*

## 1.2 Overview of Techniques

Above all, we need to have apt approximate forms of the ordered objective and write a suitable LP relaxation for ROkMed. To start with, let us first review the sparsification method proposed by Aouad and Segev [1] and Byrka et al. [5] for OkMED. In the pre-processing phase, one first guesses disjoint intervals  $I_0, I_1, \dots$  with each  $I_t \subseteq \mathbb{R}_+$  having the form  $(x, (1 + \epsilon)x]$  for some small  $\epsilon > 0$ , so that the service costs falling into the same interval differ by only a multiplicative factor of  $(1 + \epsilon)$ . Let  $w_I^{\text{avg}}$  be the average weight multiplied with service costs in the interval  $I$  in a fixed optimal solution. If we apply the same weight  $w_I^{\text{avg}}$  to all service costs in  $I$ , we can show that the optimal solution exhibits a similar objective by only losing a  $(1 + O(\epsilon))$  factor. The pre-processing phase proceeds to build the premise that the guessed intervals  $\{I_0, I_1, \dots\}$  and the guessed average weights  $\{w_{I_0}^{\text{avg}}, w_{I_1}^{\text{avg}}, \dots\}$  roughly agree with the unknown optimal solution; this is done by showing the number of necessary guesses is bounded by a polynomial, thus we can use exhaustive search. To “pre-apply” the average weights in an LP relaxation, we define a function  $f$  as  $f(d(i, j)) = w_I^{\text{avg}} \cdot d(i, j)$  for  $d(i, j) \in I$ , and put  $f(d(i, j))$  instead of  $d(i, j)$  in the LP objective. Byrka et al. [5] implicitly use such a function and give a  $(38 + \epsilon)$ -approximation for OkMED.

Unfortunately for ROkMed, this objective function seems to suffer from the inherent unbounded integrality gap in the natural relaxation for basic RkMED (see, e.g., [23]; also recall that RkMED is a special case of ROkMed). Note that this is not an issue for OkMED



since the integrality gap in the natural relaxation for  $k$ -median is a constant [11, 20]. Roughly speaking, to overcome this gap in the robust case and obtain constant-factor approximation guarantees, one usually strengthens the relaxation by adding more constraints, obtains an almost-integral solution via an auxiliary LP, and rounds the last few fractionally-open facilities to integral ones. During the last step, extra facility-client connections will incur extra costs; to the best of our efforts, the non-linearity of the ordered objective prevents us from obtaining a constant-factor approximate solution, even if we use the aforementioned new LP objective defined via  $f$ .

We overcome this technical barrier by considering another simple but effective objective function. We replace  $f(d(i, j))$  with  $f(\lambda d(i, j))$ , where  $\lambda \in (0, 1]$  is a small constant parameter and  $f$  is defined similarly as above. We note that the same function has been used in [1] to provide a *logarithmic* approximation guarantee for  $OkMED$ ; we give a much tighter analysis here and achieve a *constant* guarantee. Intuitively speaking, by scaling the underlying metric and still comparing the solutions with the original optimum, we can bound the extra costs incurred in the robust case. We point out that for the optimal service cost vector  $\mathbf{o}$ , the gap between each  $f(\mathbf{o}_j)$  and  $f(\lambda \mathbf{o}_j)$  may be  $\omega(1/\lambda)$ , since  $f$  is in fact non-decreasing and superlinear. Nevertheless, we overcome this new gap by obtaining a linear upper bound for any integral solution to the new relaxation. More specifically, let  $\text{opt} \geq 0$  be the optimum of the original instance; we show that any integral solution with an objective of  $V$  in the  $\lambda$ -scaled relaxation induces a solution to the original problem with an objective of at most  $\lambda^{-1}(V + O(1)\text{opt})$ . Furthermore, we show that there exists an algorithm which outputs an integral solution with an objective of  $V = O(\lambda)\text{opt}$ . Combining these two results, we obtain an approximate solution for  $ROkMed$  with an objective that is  $O(1/\lambda)$  times the optimum.

### 1.3 Other Related Work

Clustering problems with more general combinatorial constraints have been extensively studied in recent years. Chen et al. [12] give a 3-approximation for matroid center. Krishnaswamy et al. [22] give the first constant-factor approximation for matroid median, and thereafter the ratio is improved in [11, 27]; currently the best ratio is 7.081 due to Krishnaswamy et al. [23]. Hochbaum and Shmoys [18] study knapsack center and give a 3-approximation. As for knapsack median, Kumar [24] gives the first constant-factor approximation algorithm; the ratio is later improved in [3, 11, 23, 27], and the best ratio so far is  $(6.387 + \epsilon)$  due to Gupta et al. [16].

## 2 The Reduction Framework

In this section, we maintain a generic problem called  $\text{ORDCLST}$ , i.e., ordered clustering, which can be later instantiated as different concrete problems such as  $ROkMed$ . An instance  $\mathcal{I}$  of  $\text{ORDCLST}$  consists of a facility set  $\mathcal{F}$ , a client set  $\mathcal{C}$ , a finite metric  $d$  on  $\mathcal{F} \cup \mathcal{C}$ , feasible facility sets  $\mathcal{F} \subseteq 2^{\mathcal{F}}$ , feasible client sets  $\mathcal{C} \subseteq 2^{\mathcal{C}}$ , and a non-increasing non-negative vector  $\mathbf{w} \in \mathbb{R}_+^m$ ; each  $C \in \mathcal{C}$  satisfies  $|C| = m$ , and  $d(u, v) \geq 1$  for  $u, v \in \mathcal{F} \cup \mathcal{C}$  that are *not* co-located. The goal is to choose  $F \in \mathcal{F}$  and  $C \in \mathcal{C}$  that induce a service cost vector  $\mathbf{c} = \{d(j, F) : j \in C\}$  such that the ordered objective  $\text{cost}(\mathbf{w}; \mathbf{c}) = \mathbf{w}^\top \mathbf{c}^\downarrow$  is minimized.

We devise a general framework that reduces  $\text{ORDCLST}$  instances to other clustering problems with simpler objective functions. Given an instance  $\mathcal{I} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, \mathbf{w})$  of  $\text{ORDCLST}$  and a *non-decreasing* function  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ , we say  $\mathcal{J} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, f)$  is a *reduced instance* of  $\mathcal{I}$ , whose goal of optimization is to choose  $F \in \mathcal{F}$  and  $C \in \mathcal{C}$  such that the new objective  $\sum_{j \in C} f(d(j, F))$  is minimized. Using this reduction, we will show that



when  $f$  satisfies some certain nice properties, we only need to study the reduced instance  $\mathcal{J}$ , whose objective might be more tangible and easier to deal with. Moreover, we will show that an approximate solution to the original instance  $\mathcal{J}$  can be directly recovered from an approximate solution to  $\mathcal{J}$  by only losing a constant factor in the approximation guarantee.

The framework adapts previous sparsification methods [1, 5] for OkMED, and generalizes the helper functions therein to overcome the technical difficulties that may be present in ORDClST (see Section 1.2 for the discussion). For convenience, for each function  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  and  $\lambda > 0$ , we define  $f_\lambda(x) := f(\lambda x)$ ,  $\forall x \geq 0$ . We let  $n_0 = |\mathcal{F} \cup \mathcal{C}|$  and present the following core theorem of the reduction framework.

► **Theorem 3.** *Let  $\mathcal{J} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, \mathbf{w})$  be an instance of ORDClST with optimum  $\text{opt} \geq 0$ . For each  $\epsilon \in (0, 1)$ , there exists an algorithm that outputs  $(n_0/\epsilon)^{O(1/\epsilon)}$  non-decreasing functions  $\mathbb{R}_+ \rightarrow \mathbb{R}_+$ , such that there is an output  $f$  satisfying the following for each  $\lambda \in (0, 1]$ . We say such  $f$  is faithful.*

- *The reduced instance  $\mathcal{J}_{f_\lambda} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, f_\lambda)$  has an optimum of at most  $\lambda(1 + 9\epsilon)\text{opt}$ .*
- *If an algorithm produces a solution with objective  $V$  for  $\mathcal{J}_{f_\lambda}$ , the same solution attains an objective of at most  $\lambda^{-1}(V + (1 + 4\epsilon)\text{opt})$  for  $\mathcal{J}$ .*

As a direct consequence, if we obtain a solution to any “faithfully” reduced instance  $\mathcal{J}_{f_\lambda}$  with an objective of  $V \leq \gamma\text{opt}$ , it is a  $\lambda^{-1}(1 + \gamma + 4\epsilon)$ -approximate solution to  $\mathcal{J}$ . Before we proceed with the proof, we discuss the sparsification method used for constructing such functions. We shall only consider these functions in the remainder of this section. We note that the same functions are also used in a much more straightforward fashion in [5].

Let  $(F^*, C^*)$  be a fixed (unknown) optimal solution to the original problem,  $\mathbf{o} \in \mathbb{R}_+^{C^*}$  be the corresponding service cost vector, and  $\text{opt} = \text{cost}(\mathbf{w}; \mathbf{o})$  be the optimal objective thereof. We first guess the exact value of  $\mathbf{o}_1^\downarrow$ , i.e., the largest service cost, which only has a polynomial number of possible values. We use exhaustive search and assume  $\mathbf{o}_1^\downarrow$  is correctly guessed in the sequel; we also assume  $\mathbf{o}_1^\downarrow > 0$ , otherwise the solution is trivial.

Let  $T$  be the smallest integer s.t.  $\epsilon(1 + \epsilon)^T > m$  and define intervals  $I_{T+1}, I_T, \dots, I_0$  where

$$I_{T+1} = \left[0, \frac{\epsilon \mathbf{o}_1^\downarrow}{m}\right]; I_t = \left(\frac{\epsilon \mathbf{o}_1^\downarrow}{m}(1 + \epsilon)^{T-t}, \frac{\epsilon \mathbf{o}_1^\downarrow}{m}(1 + \epsilon)^{T-t+1}\right], \forall t \in [T]; I_0 = \left(\frac{\epsilon \mathbf{o}_1^\downarrow}{m}(1 + \epsilon)^T, +\infty\right).$$

Since  $\bigcup_{t=0}^{T+1} I_t = \mathbb{R}_+$  and they are mutually disjoint, each  $d(i, j)$  falls into exactly one interval.

Next, to avoid technical difficulties caused by weights that are too small, we define a new vector  $\tilde{\mathbf{w}}$  where  $\tilde{w}_i = \max\{w_i, \frac{\epsilon w_1}{m}\}$ ,  $i \in [m]$ . We obtain the following simple fact, by observing  $\tilde{\mathbf{w}} \geq \mathbf{w}$  and  $\text{cost}(\tilde{\mathbf{w}}; \mathbf{v}) - \text{cost}(\mathbf{w}; \mathbf{v}) \leq m \cdot \frac{\epsilon w_1}{m} \cdot \mathbf{v}_1^\downarrow \leq \epsilon \cdot \text{cost}(\mathbf{w}; \mathbf{v})$ .

► **Fact 4.** *For each  $\mathbf{v} \subseteq \mathbb{R}_+^m$ , one has  $\text{cost}(\mathbf{w}; \mathbf{v}) \leq \text{cost}(\tilde{\mathbf{w}}; \mathbf{v}) \leq (1 + \epsilon)\text{cost}(\mathbf{w}; \mathbf{v})$ .*

Now, let us consider the optimum  $\text{opt} = \mathbf{w}^\top \mathbf{o}^\downarrow$ . In particular, we consider the entries of  $\mathbf{o}^\downarrow$  that fall into different intervals  $I_{T+1}, I_T, \dots, I_0$ , and (iteratively) define the average weight  $\mathbf{w}_t^{\text{avg}}$  w.r.t.  $\mathbf{o}^\downarrow$ ,  $\tilde{\mathbf{w}}$  and interval  $I_t$ , such that  $\mathbf{w}_0^{\text{avg}} = \tilde{\mathbf{w}}_1$  and

$$\mathbf{w}_t^{\text{avg}} = \begin{cases} \left(\sum_{j: \mathbf{o}_j^\downarrow \in I_t} \tilde{w}_j\right) / |\mathbf{o}^\downarrow \cap I_t| & \mathbf{o}^\downarrow \cap I_t \neq \emptyset, t \geq 1, \\ \mathbf{w}_{t-1}^{\text{avg}} & \mathbf{o}^\downarrow \cap I_t = \emptyset, t \geq 1. \end{cases}$$

Since  $\tilde{\mathbf{w}}$  is non-increasing, it follows that  $\mathbf{w}^{\text{avg}}$  is also non-increasing. Though the actual sequence  $\mathbf{w}^{\text{avg}}$  is unknown, we can estimate it using another non-increasing sequence  $\mathbf{w}^{\text{gss}}$  such that for each  $0 \leq t \leq T + 1$ ,  $\mathbf{w}_t^{\text{gss}}$  is an integer power of  $(1 + \epsilon)$  and satisfies  $\min_s \mathbf{w}_s^{\text{avg}} \leq \mathbf{w}_t^{\text{gss}} \leq (1 + \epsilon) \max_s \mathbf{w}_s^{\text{avg}}$ . Since the entries of  $\mathbf{w}^{\text{avg}}$  are at least  $\min_j \tilde{w}_j \geq \epsilon w_1/m$  and

at most  $w_1$ , the number of possible values is  $O(\log_{1+\epsilon}(m/\epsilon))$ . By the definition of  $T$ , we have  $T = O(\log_{1+\epsilon}(m/\epsilon))$ . Thus, using routine calculation, the number of all possible non-increasing sequences for  $w^{\text{gss}}$  is at most  $(m/\epsilon)^{O(1/\epsilon)}$ . Up to now, we have only guessed  $\mathbf{o}_1^\downarrow$  and  $w^{\text{gss}}$ , hence the total number of possible guesses is at most  $(n_0/\epsilon)^{O(1/\epsilon)}$  since  $m \leq |\mathcal{C}| \leq n_0$ .

**Proof of Theorem 3.** For each guess  $(\mathbf{o}_1^\downarrow, \{w_t^{\text{gss}}\}_{t=0}^{T+1})$ , we define a piece-wise linear function

$$f(x) = w_t^{\text{gss}} x, \quad x \in I_t, \quad 0 \leq t \leq T+1.$$

Because  $w^{\text{gss}}$  is non-increasing,  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is non-decreasing and *superlinear* (i.e.,  $f(\alpha x) \geq \alpha f(x)$  for each  $\alpha \geq 1$  and  $x \geq 0$ ). According to the previous analysis, we consider at most  $(n_0/\epsilon)^{O(1/\epsilon)}$  such functions.

To prove the theorem, it suffices to show the existence of a faithful function. In the sequel, we assume that the guessed values are as desired; that is,  $\mathbf{o}_1^\downarrow$  is precisely the largest service cost in the optimal solution and for each  $0 \leq t \leq T+1$ , one has  $w_t^{\text{gss}} \in [w_t^{\text{avg}}, (1+\epsilon)w_t^{\text{avg}}]$ . We show that the corresponding function  $f$  is faithful. We need the following two lemmas.

► **Lemma 5.** Let  $\mathbf{c} = \{d(j, F) : j \in C\} \in \mathbb{R}_+^C$  where  $F \in \mathcal{F}$  and  $C \in \mathcal{C}$ . For each  $\lambda \in (0, 1]$ , one has  $\lambda \cdot \tilde{w}^\top \mathbf{c}^\downarrow \leq \sum_{j \in C} f(\lambda \mathbf{c}_j^\downarrow) + (1 + 3\epsilon + \epsilon^2)\text{opt}$ .

**Proof.** Recall that  $|C| = m$  for each feasible  $C \in \mathcal{C}$ . We suppose  $C = [m]$  for convenience. Consider each  $j \in [m]$  s.t.  $\lambda \tilde{w}_j \mathbf{c}_j^\downarrow > f(\lambda \mathbf{c}_j^\downarrow)$ . Notice that  $\lambda \mathbf{c}_j^\downarrow \notin I_0$ , otherwise one has  $w_0^{\text{gss}} \geq w_0^{\text{avg}} = \tilde{w}_1 \geq \tilde{w}_j$  and  $\lambda \tilde{w}_j \mathbf{c}_j^\downarrow > f(\lambda \mathbf{c}_j^\downarrow) = w_0^{\text{gss}}(\lambda \mathbf{c}_j^\downarrow) \geq \lambda \tilde{w}_j \mathbf{c}_j^\downarrow$ , which is a contradiction. If  $\lambda \mathbf{c}_j^\downarrow \in I_{T+1} = [0, \epsilon \mathbf{o}_1^\downarrow/m]$ , one has  $\lambda \tilde{w}_j \mathbf{c}_j^\downarrow \leq \tilde{w}_j(\epsilon \mathbf{o}_1^\downarrow/m) \leq \epsilon \cdot w^\top \mathbf{o}^\downarrow/m$  since  $\tilde{w}_j \leq \tilde{w}_1 = w_1$ .

Then, suppose  $\lambda \mathbf{c}_j^\downarrow \in I_t$ ,  $t \in [T]$ . We claim  $\lambda \mathbf{c}_j^\downarrow \leq (1+\epsilon)\mathbf{o}_j^\downarrow$ . For the sake of contradiction, assume otherwise, i.e.,  $\lambda \mathbf{c}_j^\downarrow > (1+\epsilon)\mathbf{o}_j^\downarrow$ , thus  $\lambda \mathbf{c}_j^\downarrow$  and  $\mathbf{o}_j^\downarrow$  must be in different intervals by the definition of  $I_t$ . Suppose  $\mathbf{o}_j^\downarrow \in I_{t'}$  for some  $t' > t$ , which implies  $w_t^{\text{avg}} \geq \tilde{w}_j$ , because  $w_t^{\text{avg}}$  is the average weight on  $I_t$  w.r.t.  $\mathbf{o}^\downarrow$ , and  $\tilde{w}_j$  is the weight for  $\mathbf{o}_j^\downarrow \in I_{t'}$ . Therefore, because  $w_t^{\text{gss}} \geq w_t^{\text{avg}}$  using our initial conditions, we have  $f(\lambda \mathbf{c}_j^\downarrow) = w_t^{\text{gss}}(\lambda \mathbf{c}_j^\downarrow) \geq \lambda w_t^{\text{avg}} \mathbf{c}_j^\downarrow \geq \lambda \tilde{w}_j \mathbf{c}_j^\downarrow$ , contradicting our initial assumption. Thus the claim is true.

The above analysis shows that  $\lambda \tilde{w}_j \mathbf{c}_j^\downarrow \leq f(\lambda \mathbf{c}_j^\downarrow) + (1+\epsilon)\tilde{w}_j \mathbf{o}_j^\downarrow + \epsilon w^\top \mathbf{o}^\downarrow/m$  for each  $j \in [m]$ . We take the sum over  $j \in [m]$  and obtain

$$\lambda \cdot \tilde{w}^\top \mathbf{c}^\downarrow = \lambda \sum_{j \in [m]} \tilde{w}_j \mathbf{c}_j^\downarrow \leq \sum_{j \in [m]} f(\lambda \mathbf{c}_j^\downarrow) + (1+\epsilon)\text{cost}(\tilde{w}; \mathbf{o}) + \epsilon \cdot \text{cost}(\mathbf{w}; \mathbf{o}).$$

Combining with Fact 4 and  $\text{opt} = \text{cost}(\mathbf{w}; \mathbf{o})$ , the lemma follows. ◀

► **Lemma 6.** For each  $\lambda \in (0, 1]$ , one has  $\sum_{j \in C^*} f(\lambda \mathbf{o}_j) \leq \lambda((1+\epsilon)^3 + \epsilon + \epsilon^2)\text{opt}$ .

**Proof.** Recall that  $(F^*, C^*)$  is optimal for  $\mathcal{J}$ . Consider any non-empty  $\mathbf{o} \cap I_t$ , and it is easy to verify that  $t > 0$ . Since  $\lambda \leq 1$ , some entries in  $\lambda(\mathbf{o} \cap I_t)$  may be “shifted” to  $I_{t'}$  with  $t' > t$ . If  $t \leq T$ , the contribution of  $\lambda(\mathbf{o} \cap I_t)$  on the LHS is at most

$$\sum_{\substack{j: \mathbf{o}_j^\downarrow \in I_t, \\ \lambda \mathbf{o}_j^\downarrow \notin I_t}} \lambda w_{t+1}^{\text{gss}} \mathbf{o}_j^\downarrow + \sum_{\substack{j: \mathbf{o}_j^\downarrow \in I_t, \\ \lambda \mathbf{o}_j^\downarrow \in I_t}} \lambda w_t^{\text{gss}} \mathbf{o}_j^\downarrow \leq \lambda \sum_{j: \mathbf{o}_j^\downarrow \in I_t} (1+\epsilon) w_t^{\text{avg}} \mathbf{o}_j^\downarrow \leq \lambda(1+\epsilon)^2 \sum_{j: \mathbf{o}_j^\downarrow \in I_t} \tilde{w}_j \mathbf{o}_j^\downarrow, \quad (1)$$

where the first inequality is due to non-increasing  $w^{\text{gss}}$  and  $w_t^{\text{gss}} \in [w_t^{\text{avg}}, (1+\epsilon)w_t^{\text{avg}}]$ ; the second inequality is because within the same interval  $I_t$  where  $t \leq T$ , the values of  $\mathbf{o}_j^\downarrow$  differ by a factor no more than  $(1+\epsilon)$ . More formally, we have

$$\begin{aligned}
\sum_{j: \mathbf{o}_j^\perp \in I_t} \mathbf{w}_t^{\text{avg}} \mathbf{o}_j^\perp &= \left( \frac{1}{|\mathbf{o} \cap I_t|} \sum_{j: \mathbf{o}_j^\perp \in I_t} \tilde{\mathbf{w}}_j \right) \sum_{j': \mathbf{o}_{j'}^\perp \in I_t} \mathbf{o}_{j'}^\perp \\
&= \sum_{j: \mathbf{o}_j^\perp \in I_t} \left( \frac{1}{|\mathbf{o} \cap I_t|} \sum_{j': \mathbf{o}_{j'}^\perp \in I_t} \mathbf{o}_{j'}^\perp \right) \tilde{\mathbf{w}}_j \leq \sum_{j: \mathbf{o}_j^\perp \in I_t} (1 + \epsilon) \mathbf{o}_j^\perp \tilde{\mathbf{w}}_j,
\end{aligned}$$

hence the inequality above follows.

If  $t = T + 1$ , each such  $\mathbf{o}_j^\perp \leq \epsilon \mathbf{o}_1^\perp / m$ , thus the contribution of  $\lambda(\mathbf{o} \cap I_{T+1})$  is at most  $\epsilon \lambda \mathbf{w}_1^{\text{gss}} \mathbf{o}_1^\perp \leq \epsilon(1 + \epsilon) \lambda \tilde{\mathbf{w}}_1 \mathbf{o}_1^\perp \leq \lambda(\epsilon + \epsilon^2) \text{opt}$ , since  $\mathbf{w}_1^{\text{gss}} \leq (1 + \epsilon) \mathbf{w}_1^{\text{avg}} \leq (1 + \epsilon) \tilde{\mathbf{w}}_1$ . The lemma follows by taking the sum of (1) over each  $\mathbf{o} \cap I_t$  plus  $\lambda(\epsilon + \epsilon^2) \text{opt}$  for  $\mathbf{o} \cap I_{T+1}$ , which is

$$\begin{aligned}
\sum_{j \in C^*} f(\lambda \mathbf{o}_j) &\leq \lambda(1 + \epsilon)^2 \sum_{t=1}^T \sum_{j: \mathbf{o}_j^\perp \in I_t} \tilde{\mathbf{w}}_j \mathbf{o}_j^\perp + \lambda(\epsilon + \epsilon^2) \text{opt} \\
&\stackrel{(\text{Fact 4})}{\leq} \lambda(1 + \epsilon)^3 \text{cost}(\mathbf{w}; \mathbf{o}) + \lambda(\epsilon + \epsilon^2) \text{opt}. \quad \blacktriangleleft
\end{aligned}$$

We return to the original theorem and fix  $\lambda \in (0, 1]$ . Since  $(F^*, C^*)$  is a feasible solution to both  $\mathcal{J}$  and  $\mathcal{J}_{f_\lambda}$ , the first assertion follows using  $\epsilon < 1$  and Lemma 6. For the second assertion, let  $(F, C)$  be the solution returned by the algorithm, thus  $V = \sum_{j \in C} f(\lambda d(j, F))$ . Therefore, using Fact 4 and Lemma 5, the objective of  $(F, C)$  in the ORDCLST instance  $\mathcal{J}$  is at most  $\text{cost}(\mathbf{w}; \mathbf{c}) \leq \text{cost}(\tilde{\mathbf{w}}; \mathbf{c}) \leq \lambda^{-1}(V + (1 + 4\epsilon) \text{opt})$ , where  $\mathbf{c} = \{d(j, F) : j \in C\}$ .  $\blacktriangleleft$

### 3 Applications

In this section, we provide applications of our reduction framework in Theorem 3. Due to the space limitations, we defer some proofs and details of the algorithms to the appendix.

#### 3.1 Robust Ordered $k$ -Median

In RO $k$ Med, ORDCLST is instantiated such that  $\mathcal{F}$  consists of all subsets of  $\mathcal{F}$  with cardinality at most  $k$ , i.e.,  $\mathcal{F} = \{F \subseteq \mathcal{F} : |F| \leq k\}$ ;  $\mathcal{C}$  consists of all subsets of  $\mathcal{C}$  with cardinality exactly  $m$ , i.e.,  $\mathcal{C} = \{C \subseteq \mathcal{C} : |C| = m\}$ . Via enumerating all possible functions in Theorem 3, suppose that we have a faithful function  $f$  in what follows.

As noted before, using Theorem 3, we want to obtain a constant-factor approximate solution to  $\mathcal{J}_{f_\lambda}$  for some small constant  $\lambda \in (0, 1)$ . We adapt the iterative rounding algorithm [23]. Let  $x_{ij} \in [0, 1]$  denote the extent of assigning client  $j$  to facility  $i$ , and  $y_i \in [0, 1]$  denote the extent of opening facility  $i$ . The natural relaxation for  $\mathcal{J}_{f_\lambda}$  is given as follows.

$$\begin{aligned}
\min \quad & \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}} x_{ij} f_\lambda(d(i, j)) & (\text{LP}(f_\lambda)) \\
\text{s.t.} \quad & \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}} x_{ij} \geq m \\
& \sum_{i \in \mathcal{F}} x_{ij} \leq 1 & \forall j \in \mathcal{C} \\
& \sum_{i \in \mathcal{F}} y_i \leq k \\
& 0 \leq x_{ij} \leq y_i \leq 1 & \forall i \in \mathcal{F}, j \in \mathcal{C}.
\end{aligned}$$

Before we look at the full algorithm, let us begin with a brief overview. The algorithm consists of the following two phases, namely, pre-processing and iterative rounding.

**Pre-processing.** As is discussed in the introduction, the integrality gap in  $\text{LP}(f_\lambda)$  is unbounded since  $\text{RO}k\text{Med}$  recovers  $\text{R}k\text{MED}$ . Thus, instead of directly solving  $\text{LP}(f_\lambda)$ , we employ some pre-processing techniques and simplify the instance. In what follows, let  $\lambda_1 \in (\lambda, 1]$  be another constant. The values of  $\lambda$  and  $\lambda_1$  will be determined in the full algorithm.

First, we guess a constant number of facilities  $S_0$  as must-have choices and remove some clients in advance. Consequently, we obtain a new *extended* instance  $\mathcal{J}'_f$  on the remaining clients  $\mathcal{C}'$ ; the family  $\mathcal{C}' \subseteq 2^{\mathcal{C}'}$  of client subsets in  $\mathcal{J}'_f$  consists of all  $m'$ -subsets of  $\mathcal{C}'$ , that is,  $\mathcal{C}' = \{C' \subseteq \mathcal{C}' : |C'| = m'\}$  for some fixed parameter  $m' \leq m$ ;  $\mathcal{J}'_f$  also requires that the pre-selected facilities in  $S_0$  must be part of the solution. By exhaustive search, we show that there exists some  $\mathcal{J}'_f$  with certain “sparse” properties, which is easier to approximate using an LP relaxation. More specifically, there exists a solution  $(F, C)$  to  $\mathcal{J}'_f$  such that  $S_0 \subseteq F$  and  $(F, C)$  satisfies the following for two small constants  $\rho, \delta \in (0, 1)$  (see Theorem 7). Here,  $\text{opt} \geq 0$  is the optimum of the original  $\text{RO}k\text{Med}$  instance.

- For each facility  $i \in F \setminus S_0$ , the clients assigned to  $i$  contribute at most  $\rho \cdot \text{opt}$ ; that is,  $\sum_{j \in C \text{ assigned to } i} f(d(i, j)) \leq \rho \cdot \text{opt}$ .
- For each  $p \in \mathcal{F} \cup \mathcal{C}'$ , let  $c_p = d(p, F)$ . The product of (a)  $f((1 - \delta)c_p)$  and (b) the number of served clients in  $C$  within a distance of  $\delta c_p$  from  $p$  is at most  $\rho \cdot \text{opt}$ ; that is,

$$|\{j \in C : d(j, p) \leq \delta c_p\}| \cdot f((1 - \delta)c_p) \leq \rho \cdot \text{opt}.$$

Intuitively speaking, this means that after removing the clients  $\mathcal{C} \setminus \mathcal{C}'$ , there cannot be too many clients with “large” contributions inside any such closed ball.

Further, a straightforward greedy algorithm on the removed clients  $\mathcal{C} \setminus \mathcal{C}'$  recovers a good approximate solution to  $\mathcal{J}_f$ . The two basic instances  $\{\mathcal{J}'_f, \mathcal{J}_f\}$  will be useful in the analysis.

Second, we formulate a stronger relaxation  $\text{S-LP}(f_{\lambda_1})$  that has the same objective as  $\text{LP}(f_\lambda)$  except for using a larger coefficient  $\lambda_1$ . It has both the constraints of  $\text{LP}(f_\lambda)$ , and additional constraints that guarantee certain sparse properties in its solutions; in particular,  $\mathcal{J}'_f$  also conforms to these sparsity constraints. Thus, we show that any sparse solution to  $\mathcal{J}'_{f_{\lambda_1}}$  is also feasible to  $\text{S-LP}(f_{\lambda_1})$ . We emphasize that during the algorithm, we solve  $\text{S-LP}(f_{\lambda_1})$  instead of  $\text{LP}(f_\lambda)$ ;  $\text{LP}(f_\lambda)$  will only be used in the analysis of the algorithm.

**Iterative rounding.** After obtaining a fractional optimal solution to  $\text{S-LP}(f_{\lambda_1})$ , we use the iterative rounding algorithm and obtain an integral solution  $(\hat{F}, \hat{C}')$  to  $\mathcal{J}'_{f_{\lambda_1}}$ . As aforementioned, it is easy to extend  $(\hat{F}, \hat{C}')$  to another solution  $(\hat{F}, \hat{C})$  that is feasible to  $\mathcal{J}_{f_{\lambda_1}}$ . However, because the function  $f$  in the LP objective is superlinear and our rounding algorithm incurs multiplicative factors on the input of  $f$ , we cannot directly analyze the approximation guarantee via  $\mathcal{J}_{f_{\lambda_1}}$  and  $\text{S-LP}(f_{\lambda_1})$ . Nevertheless,  $(\hat{F}, \hat{C})$  is also feasible to  $\text{LP}(f_\lambda)$  where the coefficient  $\lambda$  is smaller than  $\lambda_1$ , making it possible for us to bound the objective of  $(\hat{F}, \hat{C})$  in the instance  $\mathcal{J}_{f_\lambda}$ . Finally, we invoke Theorem 3 on  $\mathcal{J}_{f_\lambda}$  and obtain the overall approximation ratio.

### 3.1.1 The Algorithm for Robust Ordered $k$ -Median

In this section, we present our constant-factor approximation algorithm for  $\text{RO}k\text{Med}$  and prove Theorem 1. Suppose we have a faithful function  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  via Theorem 3 and exhaustive search. Let the reduced instance be  $\mathcal{J} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, f)$ . Recall  $n_0 = |\mathcal{F} \cup \mathcal{C}|$ .

### 3.1.1.1 The Sparse Instance

Let  $(F^* \in \mathcal{F}, C^* \in \mathcal{C})$  be a fixed *unknown* optimal solution to the original *ROkMed* instance  $\mathcal{J} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, \mathbf{w})$  and  $\text{opt} \geq 0$  be the optimum thereof; define  $c_p^* = \min_{i \in F^*} d(p, i)$ ,  $\kappa_p^* = \arg \min_{i \in F^*} d(p, i)$  for each  $p \in \mathcal{F} \cup \mathcal{C}$  (ties broken arbitrarily), and closed balls  $\text{Ball}_S(p, R) = \{i \in S : d(i, p) \leq R\}$ . We guess  $U \in [V^*, (1 + \epsilon)V^*]$  via binary search, where  $V^*$  is the optimum of  $\mathcal{J}$ . We have  $V^* \leq (1 + O(\epsilon))\text{opt}$  using Theorem 3. We need the following theorems on pre-processing. The proofs are given in the appendix.

► **Theorem 7.** (Similar to [23]). *Given  $\rho, \delta \in (0, 1)$  and  $U$ , there exists an  $n_0^{O(1/\rho)}$ -time algorithm that finds an extended instance  $\mathcal{J}' = (\mathcal{F}, \mathcal{C}', d, \mathcal{F}, \mathcal{C}', f, S_0)$  satisfying the following.*

(7.1)  $\mathcal{C}' \subseteq \mathcal{C}$ ,  $\mathcal{C}' = \{C' \subseteq \mathcal{C} : |C'| = m' := |C^* \cap \mathcal{C}'|\}$  and  $S_0 \subseteq F^*$  with  $|S_0| = O(1/\rho)$ .

(7.2) Denote  $C'^* = C^* \cap \mathcal{C}'$ . For each  $i \in F^* \setminus S_0$ , we have  $\sum_{j \in C'^* : \kappa_j^* = i} f(c_j^*) \leq \rho U$ .

(7.3) For each  $p \in \mathcal{F} \cup \mathcal{C}'$ , we have  $|\text{Ball}_{C'^*}(p, \delta c_p^*)| \cdot f((1 - \delta)c_p^*) \leq \rho U$ .

(7.4) Denote  $U' = \sum_{j \in C'^*} f(c_j^*)$ . We have  $\sum_{j \in C^* \setminus \mathcal{C}'} f\left(\frac{1-\delta}{1+\delta}d(j, S_0)\right) + U' \leq U$ .

Roughly speaking, Theorem 7 says that after removing a constant number of facilities  $S_0$  from  $F^*$  and some clients  $\mathcal{C} \setminus \mathcal{C}'$  from  $C^*$ , the remaining solution has some nice sparse properties. Moreover, we can easily extend a solution on  $\mathcal{J}'$  to another solution on  $\mathcal{J}$  using (7.4) such that the objective can still be bounded in terms of  $U \leq (1 + O(\epsilon))\text{opt}$ .

► **Theorem 8.** (Similar to [23]). *Given the instance  $\mathcal{J}'$  found in Theorem 7, we can efficiently compute a set of upper bounds  $\{\hat{R}_j \geq 0 : j \in \mathcal{C}'\}$  satisfying the following.*

(8.1) *There exists a solution  $(F^*, C')$  to  $\mathcal{J}'$ , such that each  $j \in \mathcal{C}'$  is assigned to  $\kappa'_j \in F^*$  and  $c'_j := d(\kappa'_j, j) \leq (1 + 3\delta/4)\hat{R}_j$ . Moreover, one has*

$$\sum_{j \in \mathcal{C}'} f\left(\frac{2}{2+\delta}c'_j\right) \leq U'; \quad \sum_{j \in \mathcal{C}' : \kappa'_j = i} f\left(\frac{2}{2+\delta}c'_j\right) \leq \rho U, \quad \forall i \in F^* \setminus S_0.$$

(8.2) *For each  $t > 0$  and  $p \in \mathcal{F} \cup \mathcal{C}'$ , one has*

$$\left| \left\{ j \in \text{Ball}_{\mathcal{C}'}\left(p, \frac{\delta}{4}t\right) : \hat{R}_j \geq t \right\} \right| \leq \frac{\rho U}{f((1-\delta)(1-\delta/4)t)}.$$

Roughly speaking, Theorem 8 says that we can efficiently find an upper bound  $\hat{R}_j$  for each  $j \in \mathcal{C}'$  such that there exists a solution for  $\mathcal{J}'$  that roughly respects these upper bounds and exhibits a similar sparse property as (7.2). Moreover, (8.2) is a stronger but somewhat different version of (7.3); its parameterized form will be useful in the analysis of the approximation guarantee.

### 3.1.1.2 The Strengthened LP

Let  $R_j = (1 + 3\delta/4)\hat{R}_j$  in Theorem 8 and define the following stronger LP relaxation for  $0 < \lambda_1 \leq 2/(2 + \delta)$ . We note that S-LP( $f_{\lambda_1}$ ) is built on the new instance  $\mathcal{J}'$ , hence admits a more “regular” solution according to Theorem 8. In our algorithm, we solve S-LP( $f_{\lambda_1}$ )

instead of  $\text{LP}(f_\lambda)$ , and conduct iterative rounding on its solution.

$$\min \sum_{j \in \mathcal{C}'} \sum_{i \in \mathcal{F}} x_{ij} f_{\lambda_1}(d(i, j)) \quad (\text{S-LP}(f_{\lambda_1}))$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{C}'} \sum_{i \in \mathcal{F}} x_{ij} \geq m'$$

$$\sum_{i \in \mathcal{F}} x_{ij} \leq 1 \quad \forall j \in \mathcal{C}'$$

$$\sum_{i \in \mathcal{F}} y_i \leq k$$

$$0 \leq x_{ij} \leq y_i \leq 1 \quad \forall i \in \mathcal{F}, j \in \mathcal{C}'$$

$$y_i = 1 \quad \forall i \in S_0 \quad (\text{S-LP.5})$$

$$x_{ij} = 0 \quad d(i, j) > R_j \quad (\text{S-LP.6})$$

$$x_{ij} = 0 \quad \forall i \notin S_0, f(2d(i, j)/(2 + \delta)) > \rho U \quad (\text{S-LP.7})$$

$$\sum_{j \in \mathcal{C}'} f(2d(i, j)/(2 + \delta)) x_{ij} \leq \rho U y_i \quad \forall i \notin S_0. \quad (\text{S-LP.8})$$

► **Lemma 9.** *The optimal objective value of  $\text{S-LP}(f_{\lambda_1})$  is at most  $\frac{\lambda_1(2+\delta)}{2} U'$ .*

**Proof.** Using (8.1), there exists an integral solution with an objective of at most  $U'$  when  $\lambda_1 = 2/(2 + \delta)$ . For  $\lambda_1 \leq 2/(2 + \delta)$ , the same solution is still feasible because the constraints are independent of  $\lambda_1$ . For  $\alpha \leq 1, z > 0$ , we have  $f(\alpha z) \leq \alpha f(z)$  because  $f$  is non-decreasing and superlinear, thus  $\sum_{j \in \mathcal{C}'} f(\lambda_1 c'_j) \leq \frac{\lambda_1(2+\delta)}{2} \sum_{j \in \mathcal{C}'} f\left(\frac{2}{2+\delta} c'_j\right) \leq \frac{\lambda_1(2+\delta)}{2} U'$ . ◀

After we solve  $\text{S-LP}(f_{\lambda_1})$  and obtain an optimal solution  $(x^*, y^*)$ , to eliminate the  $x^*$  variables and work with an auxiliary LP that is purely on the  $y^*$  variables, we need the following lemma due to [23]. Note that this is different from simple facility duplication [11], since we need a certain sparse property of the modified solution (see 5), which helps us bound the additional rounding cost in the final analysis.

► **Lemma 10.** *We can add co-located copies to  $\mathcal{F}$ , create a vector  $y^* \in [0, 1]^{\mathcal{F}}$  and define subsets  $F_j \subseteq \text{Ball}_{\mathcal{F}}(j, R_j)$  for each client  $j \in \mathcal{C}'$ , such that the following holds.*

(10.1)  $y^*(F_j) \leq 1$  for each  $j \in \mathcal{C}'$  and  $\sum_{j \in \mathcal{C}'} (\sum_{i \in F_j} y_i^*) \geq m'$ .

(10.2)  $\sum_{i \in \mathcal{F}} y_i^* \leq k$ .

(10.3) For each  $i \in S_0$ ,  $\sum_{i' \text{ co-located with } i} y_{i'}^* = 1$ .

(10.4)  $\sum_{j \in \mathcal{C}'} \sum_{i \in F_j} y_i^* f_{\lambda_1}(d(i, j)) \leq \frac{\lambda_1(2+\delta)}{2} U'$ .

(10.5) For each  $i$  not co-located with  $S_0$ ,  $\sum_{j \in \mathcal{C}': i \in F_j} f\left(\frac{2}{2+\delta} d(i, j)\right) \leq 2\rho U$ .

**Proof.** We start with an optimal solution  $(x^*, y^*)$  to  $\text{S-LP}(f_{\lambda_1})$  with objective at most  $\frac{\lambda_1(2+\delta)}{2} U'$  according to Lemma 9. To avoid confusion in notation, we create a copy  $\mathcal{F}' = \mathcal{F}$ , define  $F_j = \{i \in \mathcal{F}' : x_{ij}^* > 0\}$  and  $\bar{y}^* \leftarrow y^*$  both supported on  $\mathcal{F}'$ . For each copy  $i' \in \mathcal{F}'$  of  $i \in \mathcal{F}$ , define its *star cost* as  $\sum_{j \in \mathcal{C}': i' \in F_j} f(\frac{2}{2+\delta} d(i, j))$ .

We iteratively perform the following procedures. For each  $i \in \mathcal{F}$  and  $j \in \mathcal{C}'$  such that  $x_{ij}^* > 0$ , we sort all copies of  $i$  in  $\mathcal{F}'$  in non-decreasing order of their current star costs, and choose the first several copies such that their  $\bar{y}^*$  values add up to *exactly*  $x_{ij}^*$ . If we need to split a facility  $i'$  into two copies to make the sum exact, we replace  $i'$  with  $\{i'_1, i'_2\}$  in  $\mathcal{F}'$ , set  $\bar{y}_{i'_1}^*$  to whichever value is needed and  $\bar{y}_{i'_2}^* \leftarrow \bar{y}_{i'}^* - \bar{y}_{i'_1}^*$ . Remove from  $F_j$  all copies of  $i$ , and add the selected copies to  $F_j$  again. For any other  $j' \neq j$ , if some  $i' \in F_{j'}$  is split in two,  $F_{j'} \leftarrow F_{j'} \setminus \{i'\} \cup \{i'_1, i'_2\}$ .

After the procedures, we set  $\mathcal{F} \leftarrow \mathcal{F}'$  and the corresponding  $y^* \leftarrow \bar{y}^*$ ,  $\{F_j\}_{j \in \mathcal{C}'}$  such that they are supported on  $\mathcal{F}$ . 1 to 4 are easy to verify, since the original solution to  $\text{S-LP}(f_{\lambda_1})$  is preserved up to facility duplication. To see 5, consider each (original) facility  $i$  and all clients  $j$  such that  $x_{ij}^* > 0$ , denoted by  $J_i \subseteq \mathcal{C}'$ . It is easy to see each copy of  $i$  only appears in  $\bigcup_{j \in J_i} F_j$ . We use induction to show that, after each iteration, the difference between the maximum and minimum star costs among all copies of  $i$  is at most  $\rho U$ .

The copies of  $i$  and their star costs may only change after an iteration where  $i$  is selected. Suppose  $J_i = \{j_1, \dots, j_\ell\}$  and we consider the iterations in the order of  $(i, j_1), \dots, (i, j_\ell)$ . As the base case, before  $(i, j_1)$  is considered, the claim is true because  $i$  has only one copy in  $\mathcal{F}'$ .

Suppose the claim is true after  $(i, j_{t-1})$ ,  $t \geq 1$ . In the start of the iteration on  $(i, j_t)$ , we sort the copies of  $i$  in non-decreasing order of their current star costs; each client  $j_s$ ,  $s \geq t$  contributes equally to the star cost of each copy of  $i$ , including  $j_t$  in particular, and the difference between the maximum and minimum is at most  $\rho U$ , using the induction hypothesis. During this iteration, we remove the contributions of  $j_t$  to all copies, and add them back to copies that have the smallest star costs. Since  $f(\frac{2}{2+\delta}d(i, j_t)) \leq \rho U$  by (S-LP.7), it is easy to verify that the difference between the maximum and minimum after the iteration is still at most  $\rho U$ . This finishes the induction.

For facility  $i$ , we let  $\mathcal{F}(i) \subseteq \mathcal{F}'$  be the copies of  $i$  after the procedures. It follows that

$$\begin{aligned} \sum_{i' \in \mathcal{F}(i)} \bar{y}_{i'}^* \sum_{j \in \mathcal{C}': i' \in F_j} f\left(\frac{2}{2+\delta}d(i, j)\right) &= \sum_{j \in J_i} f\left(\frac{2}{2+\delta}d(i, j)\right) \sum_{i' \in \mathcal{F}(i) \cap F_j} \bar{y}_{i'}^* \\ &= \sum_{j \in J_i} x_{ij}^* f\left(\frac{2}{2+\delta}d(i, j)\right) \leq \rho U y_i^*, \end{aligned}$$

where the last inequality is due to (S-LP.8). Hence, the minimum star cost is at most  $\rho U y_i^* / \sum_{i' \in \mathcal{F}(i)} \bar{y}_{i'}^* = \rho U$ , and the maximum is at most  $2\rho U$ , yielding 5.  $\blacktriangleleft$

### 3.1.1.3 Iterative Rounding

We obtain  $y^* \in [0, 1]^{\mathcal{F}}$  and  $\{F_j\}_{j \in \mathcal{C}'}$  using Lemma 10. To optimize our approximation factor, we use the following *deterministic* metric discretization. Fix  $\tau > 1$ ; define  $D_{-2} = -1$ ,  $D_{-1} = 0$  and  $D_l = \tau^l$  for each  $l \geq 0$ ; let  $d'(i, j) = \min\{D_l \geq d(i, j) : l \geq -2\}$ . For each  $j \in \mathcal{C}'$ , we call  $F_j$  its *outer ball*, define its *radius level*  $l_j \in \mathbb{Z}$  such that  $D_{l_j} = \max_{i \in F_j} d'(i, j)$ , and define its *inner ball*  $B_j = \{i \in F_j : d'(i, j) \leq D_{l_j-1}\}$ . For  $0 < \lambda_2 \leq 1/\tau$ , we define an auxiliary LP.

$$\begin{aligned} \min \quad & \sum_{j \in C_{\text{part}}} \sum_{i \in F_j} y_i f_{\lambda_2}(d'(i, j)) + \sum_{j \in C_{\text{full}}} \left( \sum_{i \in B_j} y_i f_{\lambda_2}(d'(i, j)) + (1 - y(B_j)) f_{\lambda_2}(D_{l_j}) \right) \\ & \hspace{15em} (\text{A-LP}(f_{\lambda_2})) \\ \text{s.t.} \quad & y(F_j) = 1 \quad \forall j \in C_{\text{core}} \quad (\text{A-LP.1}) \\ & 0 \leq y_i \leq 1 \quad \forall i \in \mathcal{F} \quad (\text{A-LP.2}) \\ & y(B_j) \leq 1 \quad \forall j \in C_{\text{full}} \quad (\text{A-LP.3}) \\ & y(F_j) \leq 1 \quad \forall j \in C_{\text{part}} \quad (\text{A-LP.4}) \\ & y(\mathcal{F}) \leq k \quad (\text{A-LP.5}) \\ & |C_{\text{full}}| + \sum_{j \in C_{\text{part}}} y(F_j) \geq m'. \quad (\text{A-LP.6}) \end{aligned}$$

The objective of  $\text{A-LP}(f_{\lambda_2})$  is determined by three subsets of clients  $C_{\text{full}}$ ,  $C_{\text{part}}$ , and  $C_{\text{core}}$ , such that  $C_{\text{full}} \cup C_{\text{part}} = \mathcal{C}'$ ; each client in  $C_{\text{full}}$  is to be assigned an open facility relatively close to it, and  $C_{\text{core}}$  is used for placing these facilities. Initially, we set  $C_{\text{full}} = \emptyset$ ,



## 34:12 Ordered $k$ -Median with Outliers

$C_{\text{part}} = \mathcal{C}'$  and  $C_{\text{core}} = S_0$ ; each  $i \in S_0$  is called a *virtual client* and its initial radius level is  $-1$ , since  $\sum_{i' \text{ co-located with } i} y_{i'}^* = 1$  by 3 and  $D_{-1} = 0$ . We use the following Algorithm 1 to iteratively change  $y^*$  and A-LP( $f_{\lambda_2}$ ).

■ **Algorithm 1** Iterative Rounding [23].

---

**Input** : outer balls  $\{F_j : j \in \mathcal{C}'\}$ , radius levels  $\{l_j : j \in \mathcal{C}'\}$ , inner balls  $\{B_j : j \in \mathcal{C}'\}$ ,  $S_0$

**Output** : an output solution  $y'$

```

1 $C_{\text{full}} \leftarrow \emptyset, C_{\text{part}} \leftarrow \mathcal{C}', C_{\text{core}} \leftarrow S_0$
2 while true do
3 find an optimal basic feasible solution y' to A-LP(f_{λ_2})
4 if there exists $j \in C_{\text{part}}$ such that $y'(F_j) = 1$ then
5 $C_{\text{part}} \leftarrow C_{\text{part}} \setminus \{j\}, C_{\text{full}} \leftarrow C_{\text{full}} \cup \{j\}, B_j \leftarrow \{i \in F_j : d'(i, j) \leq D_{l_j-1}\},$
 update- $C_{\text{core}}(j)$
6 else if there exists $j \in C_{\text{full}}$ such that $y'(B_j) = 1$ then
7 $l_j \leftarrow l_j - 1, F_j \leftarrow B_j, B_j \leftarrow \{i \in F_j : d'(i, j) \leq D_{l_j-1}\},$ update- $C_{\text{core}}(j)$
8 else break
9 return y'
10 update- $C_{\text{core}}(j)$
11 if there exists no $j' \in C_{\text{core}}$ with $l_{j'} \leq l_j$ and $F_j \cap F_{j'} \neq \emptyset$ then
12 $\text{remove from } C_{\text{core}} \text{ all } j' \text{ such that } F_j \cap F_{j'} \neq \emptyset, C_{\text{core}} \leftarrow C_{\text{core}} \cup \{j\}$

```

---

► **Lemma 11.** *In each iteration,  $y'$  is feasible after modifying the LP. The objective value of  $y'$  is non-increasing throughout the algorithm.*

**Proof.** There are two cases. The first is when we move some  $j$  from  $C_{\text{part}}$  to  $C_{\text{full}}$  when  $y'(F_j) = 1$ . Since  $B_j \subseteq F_j$ , it satisfies the new constraints in (A-LP.3) and (A-LP.1), if it is added to  $C_{\text{core}}$ . Since  $F_j = B_j \cup (F_j \setminus B_j)$ , the contribution of  $j$  to the new objective is the same as when it is in  $C_{\text{part}}$ , because each  $i \in F_j \setminus B_j$  satisfies  $d'(i, j) = D_{l_j}$  by definition.

The second case is when we decrease the radius level  $l_j$  and invoke the subroutine on  $j \in C_{\text{full}}$ , where  $y'(B_j) = 1$ . Comparing the contributions of  $j$  before and after the iteration, they are equal since the old contribution has  $1 - y'(B_j) = 0$ , and we can partition the new outer ball  $F_j \leftarrow B_j$  in the same way as above.

In both cases, the objective of  $y'$  does not change during an iteration. At the beginning of each iteration, we solve for an optimal basic feasible solution, thus the lemma follows. ◀

► **Property 12.** *After each iteration of Algorithm 1, the following properties hold.*

(12.1)  $C_{\text{full}}$  and  $C_{\text{part}}$  form a partition of  $\mathcal{C}'$ ,  $S_0 \subseteq C_{\text{core}}$  and  $C_{\text{core}} \setminus S_0 \subseteq C_{\text{full}}$ .

(12.2)  $\{F_j : j \in C_{\text{core}}\}$  are mutually disjoint.

(12.3) For each  $j \in \mathcal{C}'$ ,  $D_{l_j} \leq \tau R_j$ .

(12.4) For each  $j \in \mathcal{C}'$ ,  $l_j \geq -1$ .

(12.5) For each  $i$  not co-located with  $S_0$ ,  $\sum_{j \in \mathcal{C}' : i \in F_j} f(\frac{2}{2+\delta} d(i, j)) \leq 2\rho U$ .

**Proof.** First, by iteratively decreasing the radius levels, we claim that no client can have a radius level of  $-2$  or smaller. This is because when  $l_j = -1$ , its inner ball is  $B_j = \{i \in F_j : d'(i, j) \leq D_{-2} = -1\} = \emptyset$ , the constraint  $y(B_j) \leq 1$  cannot be tight, and we will not invoke the subroutine **update-** $C_{\text{core}}$  on  $j$ . This shows (12.4).

To see (12.1), we only need to show that virtual clients in  $S_0$  cannot be removed from  $C_{\text{core}}$ . From the subroutine **update- $C_{\text{core}}$** ,  $j'$  can be removed by  $j$  only when  $l_j < l_{j'}$ . But each virtual client starts with a radius level of  $-1$ , and removing any such virtual client means a radius level of  $-2$ , a contradiction.

(12.2) clearly follows by the definition of the subroutine. (12.3) is due to  $F_j \subseteq \text{Ball}_{\mathcal{F}}(j, R_j)$  at the beginning of iterative rounding, hence  $D_{l_j} \leq \max_{i \in F_j} d'(i, j) \leq \tau \max_{i \in F_j} d(i, j) \leq \tau R_j$ . Lastly, since each  $F_j$ ,  $j \in \mathcal{C}'$  is inclusion-wise non-increasing during Algorithm 1, the sum in (12.5), being the star cost of  $i$ , is also non-increasing and at most  $2\rho U$  due to 5. This yields (12.5).  $\blacktriangleleft$

We now establish the connection between  $\text{S-LP}(f_{\lambda_1})$  and  $\text{A-LP}(f_{\lambda_2})$ , making it possible to compare their objectives, before and after the iterative rounding process.

► **Lemma 13.** *For each  $\lambda_1 \in (0, 1]$  and  $\lambda_2 = \lambda_1/\tau$ , the solution  $y^*$  obtained in Lemma 10 is feasible to  $\text{A-LP}(f_{\lambda_2})$  with its objective not increased.*

**Proof.**  $y^*$  is the same as the optimal solution for  $\text{S-LP}(f_{\lambda_1})$  up to facility duplication. Before Algorithm 1, we have  $C_{\text{core}} = S_0$ . Because we require  $y_i = 1$  for each  $i \in S_0$  in  $\text{S-LP}(f_{\lambda_1})$  and we can let  $F_i$  consist of all copies of  $i$  (as a virtual client), (A-LP.1) is satisfied by  $y^*$ . Initially,  $C_{\text{full}}$  is empty and we only have  $y(F_j) \leq 1$  for  $j \in \mathcal{C}'$  (i.e., (A-LP.4)) and  $\sum_{j \in \mathcal{C}'} y(F_j) \geq m'$  (i.e., (A-LP.6)), which are also satisfied by  $y^*$  due to Lemma 10. Therefore,  $y^*$  is indeed feasible to  $\text{A-LP}(f_{\lambda_2})$ .

In  $\text{S-LP}(f_{\lambda_1})$ , each facility-client pair  $(i, j)$  has a contribution of  $x_{ij}^* f_{\lambda_1}(d(i, j))$ . In  $\text{A-LP}(f_{\lambda_2})$ , because  $C_{\text{full}} = \emptyset$ , its contribution is  $y_i^* f_{\lambda_2}(d'(i, j))$  only when  $i \in F_j$  and zero otherwise. Since  $d'$  is rounded-up by a factor of at most  $\tau$  and  $\lambda_1 = \tau\lambda_2$ , we further obtain

$$y_i^* f_{\lambda_2}(d'(i, j)) \leq y_i^* f_{\lambda_1}(d(i, j)),$$

thus for  $y^*$ , the objective of  $\text{A-LP}(f_{\lambda_2})$  is at most the objective of  $\text{S-LP}(f_{\lambda_1})$ .  $\blacktriangleleft$

► **Lemma 14.** *There are at most two fractional variables in the output solution  $y'$ . At the conclusion of the algorithm, for each  $j \in C_{\text{full}}$ ,*

$$\sum_{i \in \mathcal{F}: d(i, j) \leq \frac{3\tau-1}{\tau-1} D_{l_j}} y'_i \geq 1.$$

**Proof.** Since  $y'$  is an optimal basic feasible solution, if it has  $t > 0$  strictly fractional variables, there are at least  $t$  non-trivial (i.e., not in the form of  $y_i \geq 0$  or  $y_i \leq 1$ ) and independent constraints of  $\text{A-LP}(f_{\lambda_2})$  that are tight at  $y'$  (see, e.g., [25]). The remaining constraints form a knapsack constraint (A-LP.6), and a laminar family (A-LP.1) plus (A-LP.5), according to (12.2). The number of such tight independent constraints is therefore at most  $t/2 + 1$ . This means that  $t/2 + 1 \geq t$ , and thus  $t \in \{1, 2\}$ .

To show the second assertion, we first use induction to show that, for each  $j$  that is added to  $C_{\text{core}}$  during Algorithm 1 with radius level  $l$ , the final solution satisfies  $\sum_{i \in \mathcal{F}: d(i, j) \leq \frac{\tau+1}{\tau-1} D_l} y'_i \geq 1$ . The base case is simple for  $l = -1$ , since we know such  $j$  cannot be removed from  $C_{\text{core}}$ , and  $y'$  satisfies the inequality due to (A-LP.1). Suppose the claim is true up to  $l - 1$ ,  $l \geq 0$ . For  $j$  added to  $C_{\text{core}}$  with radius level  $l$ , if it is not later removed from  $C_{\text{core}}$ , the claim directly follows from (A-LP.1). Otherwise, if  $j$  is later removed by  $j'$  with  $l_{j'} < l_j = l$  and  $F_{j'} \cap F_j \neq \emptyset$ , using the induction hypothesis, the inequality holds for  $j'$  and  $l_{j'}$ , where  $D_{l_{j'}} \leq D_l/\tau$ . Using the triangle inequality, all these facilities are at a distance at most  $\frac{\tau+1}{\tau-1} D_{l_{j'}} + D_{l_{j'}} + D_l \leq (\frac{\tau+1}{\tau(\tau-1)} + \frac{1}{\tau} + 1) D_l = \frac{\tau+1}{\tau-1} D_l$  from  $j$ , showing the induction step.

Back to the proof of the lemma. When we invoke  $j$  on its *final* radius  $l_j$ , if we can indeed add  $j$  to  $C_{\text{core}}$ , the claim above is sufficient since  $\frac{\tau+1}{\tau-1} \leq \frac{3\tau-1}{\tau-1}$ . If it cannot be added to  $C_{\text{core}}$ , it is because there exists  $j' \in C_{\text{core}}$  with  $F_{j'} \cap F_j \neq \emptyset$  and  $l_{j'} \leq l_j$ . Using the claim on the iteration when we add  $j'$  to  $C_{\text{core}}$  with radius level  $l_{j'}$ , and using the triangle inequality, all the facilities in the sum are at a distance at most  $\frac{\tau+1}{\tau-1}D_{l_{j'}} + D_{l_{j'}} + D_{l_j} \leq \frac{3\tau-1}{\tau-1}D_{l_j}$ , whence the lemma follows.  $\blacktriangleleft$

The following lemma is concerned with the objective value of  $y'$  in  $\text{LP}(f_\lambda)$ , where we replace  $(m, \mathcal{C})$  with  $(m', \mathcal{C}')$  and use the name  $\text{LP}(f_\lambda)$  here with a slight abuse of notation.

► **Lemma 15.** *Let  $0 < \lambda \leq \frac{2\tau-2}{\tau(3\tau-1)(2+\delta)}$ . Let  $y^*$  and outer balls  $\{F_j\}_{j \in \mathcal{C}'}$  be obtained from Lemma 10 and  $\text{S-LP}(f_{\lambda_1})$  where  $\lambda_1 = \frac{\tau(3\tau-1)}{\tau-1}\lambda$ . The iterative rounding algorithm on  $\text{A-LP}(f_{\lambda_2})$  where  $\lambda_2 = \frac{3\tau-1}{\tau-1}\lambda$  returns a solution  $y'$  with at most two fractions. Moreover,  $y'$  is a feasible solution to  $\text{LP}(f_\lambda)$  with objective at most  $\frac{\lambda\tau(3\tau-1)(2+\delta)}{2\tau-2}U'$ .*

**Proof.** From Lemma 9 and Lemma 13,  $y^*$  is feasible to  $\text{A-LP}(f_{\lambda_2})$  and its objective value is upper bounded by  $\frac{\lambda_1(2+\delta)}{2}U' = \frac{\lambda\tau(3\tau-1)(2+\delta)}{2(\tau-1)}U'$ . From Lemma 14, the final solution  $y'$  has at most two fractional values; if we further take  $y'$  to  $\text{LP}(f_\lambda)$ , we can assign each client in  $C_{\text{full}} \setminus C_{\text{core}}$  to an extent of 1 to facilities at most  $\frac{3\tau-1}{\tau-1}D_{l_j}$  away; the feasibility of  $y'$  w.r.t.  $\text{LP}(f_\lambda)$  is guaranteed by Lemma 11 and Property 12. From Lemma 11 and Lemma 13, the objective value of  $y'$  in  $\text{LP}(f_\lambda)$  is upper-bounded by (recall that  $d' \geq d$  and  $C_{\text{full}} \cup C_{\text{part}}$  is a partition of  $\mathcal{C}'$ )

$$\sum_{j \in C_{\text{part}}} \sum_{i \in F_j} y'_i f_\lambda(d'(i, j)) + \sum_{j \in C_{\text{full}}} \left( \sum_{i \in B_j} y'_i f_\lambda(d'(i, j)) + (1 - y'(B_j)) f_\lambda \left( \frac{3\tau-1}{\tau-1} D_{l_j} \right) \right). \quad (2)$$

Because  $\lambda_2 = \lambda \frac{3\tau-1}{\tau-1}$ , (2) is at most the objective of  $\text{A-LP}(f_{\lambda_2})$  and  $\leq \frac{\lambda\tau(3\tau-1)(2+\delta)}{2\tau-2}U'$ .  $\blacktriangleleft$

The following theorem converts the almost-integral solution  $y'$  to an integral one  $\hat{y}$ .

► **Theorem 16.** *There exists  $\lambda > 0$  depending on  $\delta$  and  $\tau$  such that we can efficiently compute an integral solution  $\hat{y}$  to  $\text{LP}(f_\lambda)$  with objective value at most  $5\rho U$  larger than that of  $y'$ .*

**Proof.** The case of less than 2 fractions are easier thus omitted here; in the rest of the proof, suppose  $y'_{i_1}$  and  $y'_{i_2}$  are the two fractional variables. Because  $y'$  is a basic feasible solution, we must have  $y'_{i_1} + y'_{i_2} = 1$  since the tight constraints in  $\text{A-LP}(f_{\lambda_2})$  represent the intersection of a laminar family and a knapsack constraint. Let  $C_1 = \{j \in C_{\text{part}} : i_1 \in F_j, i_2 \notin F_j\}$  and  $C_2 = \{j \in C_{\text{part}} : i_1 \notin F_j, i_2 \in F_j\}$ . W.l.o.g., we assume  $|C_1| \geq |C_2|$ . One has  $|C_1| + |C_{\text{full}}| \geq y'_{i_1}|C_1| + y'_{i_2}|C_2| + |C_{\text{full}}| \geq m'$  using (A-LP.6). Define  $\hat{y}$  such that  $\hat{y}_{i_1} = 1$ ,  $\hat{y}_{i_2} = 0$  and  $\hat{y}_i = y'_i$  for  $i \in \mathcal{F} \setminus \{i_1, i_2\}$ . Let  $\hat{F} = \{i \in \mathcal{F} : \hat{y}_i = 1\}$  and  $\hat{C}' = C_1 \cup C_{\text{full}}$ .

Using (12.5), the extra cost of assigning all of  $C_1$  to  $i_1$  is at most

$$\sum_{j \in C_1} f_\lambda(d(i_1, j)) \leq \sum_{j \in C_1} f \left( \frac{2d(i_1, j)}{2 + \delta} \right) \leq \sum_{j: i_1 \in F_j} f \left( \frac{2d(i_1, j)}{2 + \delta} \right) \leq 2\rho U \Leftarrow \lambda \leq \frac{2}{2 + \delta}. \quad (3)$$

Next, because we reduce the extent of opening  $i_2$  to zero, it remains to bound the extra cost of re-assigning full clients that were assigned to  $i_2$ , defined as  $J = \{j \in C_{\text{full}} : i_2 \in B_j\}$ ; we choose  $J$  here because these are the *full* clients whose contributions are changed in (2). Let  $\gamma > 0$  be some constant which we will determine later. Let  $i^*$  be the nearest facility to  $i_2$  in  $\hat{F}$  and  $t' = d(i_2, i^*)$ . Let  $J_1 = \{j \in J : d(j, i_2) > \gamma t'\}$  and  $J_2 = \{j \in J : d(j, i_2) \leq \gamma t'\}$ . For  $j \in J_1$ , we have  $d(j, i^*) \leq d(j, i_2) + d(i_2, i^*) < (1 + \frac{1}{\gamma})d(j, i_2)$ , thus if we want the following upper bound on the extra assignment costs,

$$\sum_{j \in J_1} f(\lambda d(j, i^*)) \leq \sum_{j \in J_1} f\left(\frac{2}{2+\delta} d(j, i_2)\right) \leq 2\rho U \Leftarrow \lambda \leq \frac{2}{2+\delta} \cdot \frac{\gamma}{1+\gamma}. \quad (4)$$

For  $j \in J_2$ , let  $i'$  be the nearest open facility to  $j$ . It is easy to verify that  $d(j, i') \leq \frac{3\tau-1}{\tau-1} D_{i_j}$  using Lemma 14 and the definition of  $\hat{y}$ . Since  $i^*$  is the nearest to  $i_2$ , one has

$$d(j, i') \geq d(i_2, i') - d(j, i_2) \geq d(i_2, i^*) - \gamma t' = (1-\gamma)t'$$

using the triangle inequality, and thus  $R_j \geq D_{i_j}/\tau \geq \frac{\tau-1}{\tau(3\tau-1)}(1-\gamma)t'$ . Let  $t = \frac{\tau-1}{\tau(3\tau-1)}(1-\gamma)t'$ . Suppose that  $\frac{\delta}{4+3\delta}t \geq \gamma t'$ , then using (8.2) and recalling  $R_j = (1+3\delta/4)\hat{R}_j$ , we have

$$|J_2| \leq \left| \left\{ j \in \text{Ball}_{C'}\left(i_2, \frac{\delta}{4+3\delta}t\right) : R_j \geq t \right\} \right| \leq \frac{\rho U}{f\left(\frac{(1-\delta)(1-\delta/4)}{1+3\delta/4}t\right)}.$$

Using the triangle inequality, we have  $d(j, i^*) \leq (1+\gamma)t'$ . The total extra cost of assigning  $J_2$  to  $i^*$  is at most

$$\sum_{j \in J_2} f(\lambda d(j, i^*)) \leq f(\lambda(1+\gamma)t')|J_2| \leq \rho U \Leftarrow \lambda \leq \frac{(1-\delta)(1-\delta/4)}{(1+3\delta/4)} \cdot \frac{\tau-1}{\tau(3\tau-1)} \cdot \frac{1-\gamma}{1+\gamma}. \quad (5)$$

Denote  $\sigma = \frac{\tau-1}{\tau(3\tau-1)}$  and let  $\gamma = \frac{\delta\sigma}{4+3\delta+\delta\sigma}$  so that  $\frac{\delta}{4+3\delta}t = \gamma t'$ . By letting  $\lambda$  be the minimum of (3)(4)(5) and summing over the three cases, the increase of objective value w.r.t.  $\text{LP}(f_\lambda)$  is at most  $2\rho U + 2\rho U + \rho U = 5\rho U$ , thus the theorem follows.  $\blacktriangleleft$

### 3.1.1.4 Proof of Theorem 1

Let  $\frac{\tau-1}{\tau(3\tau-1)} = 0.101$  and  $\delta = 0.81765$ , thus  $\lambda \in (0.008856, 0.008857)$  (see the proof of Theorem 16). We fix  $\epsilon > 0$  and obtain a faithful function  $f$  using Theorem 3. Fix two small constants  $\delta, \rho > 0$ , compute  $C', m', S_0, \{R_j : j \in C'\}$  via Theorem 7 and Theorem 8, and solve  $\text{S-LP}(f_{\lambda_1})$  with  $\lambda_1 = \frac{\tau(3\tau-1)}{\tau-1}\lambda$ . Using iterative rounding, we obtain an almost-integral solution to  $\text{LP}(f_\lambda)$  using Lemma 15. Next, we compute an integral solution  $\hat{y}$  using Theorem 16, with the objective w.r.t.  $\text{LP}(f_\lambda)$  increased by at most  $5\rho U$ .

Let  $\hat{F} = \{i \in \mathcal{F} : \hat{y}_i = 1\}$  and  $\hat{C}' = C_1 \cup C_{\text{full}}$  as in the proof of Theorem 16. There are at least  $m'$  clients in  $\hat{C}'$ . We assume  $|\hat{C}'| \leq m$ , otherwise the following argument is simpler. We greedily connect  $m - |\hat{C}'| \leq m - m'$  clients in  $\mathcal{C} \setminus C'$  to their nearest open facilities in  $\hat{F}$ , minimizing  $f\left(\frac{1-\delta}{1+\delta}d(j, \hat{F})\right)$  for each of them and output the final solution  $(\hat{F}, \hat{C})$ . We consider the objective of  $(\hat{F}, \hat{C})$  in  $\mathcal{J}_{f_\lambda}$  on  $\hat{C} \setminus C'$  and  $\hat{C} \cap C'$  separately.

$$\begin{aligned} & \sum_{j \in \hat{C} \setminus C'} f(\lambda d(j, \hat{F})) + \sum_{j \in \hat{C} \cap C'} f(\lambda d(j, \hat{F})) \\ & \leq \sum_{j \in C^* \setminus C'} \frac{\lambda(1+\delta)}{1-\delta} f\left(\frac{1-\delta}{1+\delta}d(j, \hat{F})\right) + \left(\frac{\lambda\tau(3\tau-1)(2+\delta)}{2\tau-2}U' + 5\rho U\right) \\ & \leq \max\left\{\frac{\lambda(1+\delta)}{1-\delta}, \frac{\lambda\tau(3\tau-1)(2+\delta)}{2\tau-2}\right\} \left(\sum_{j \in C^* \setminus C'} f\left(\frac{1-\delta}{1+\delta}d(j, S_0)\right) + U'\right) + 5\rho U \\ & \leq 0.12354U + 5\rho U. \end{aligned} \quad (6)$$

In the above, the first inequality is due to Lemma 15, Theorem 16 and the greedy selection of  $\hat{C} \setminus C'$ . The second is because  $S_0 \subseteq \hat{F}$ . The last is due to (7.4) and our choices of parameters.

Recall that  $U \leq (1+\epsilon)V^*$  where  $V^*$  is the optimal objective of  $\mathcal{J}_f$  and  $U \leq (1+O(\epsilon))\text{opt}$ . Using (6) and Theorem 3 again, the objective of  $(\hat{F}, \hat{C})$  in the original instance  $\mathcal{I}$  is at most  $(0.12354U + 5\rho U + (1 + O(\epsilon))\text{opt})/\lambda \leq (126.9 + O(\epsilon + \rho))\text{opt} \leq 127\text{opt}$ , by choosing small enough  $\rho$  and  $\epsilon$ . The running time is obtained from the enumeration process and bounded by a polynomial.  $\blacktriangleleft$

### 3.2 Ordered Matroid/Knapsack Median

We consider the ordered matroid median problem (OMatMed) and the ordered knapsack median problem (OKnapMed). Formally, in OMatMed, we instantiate ORDCLST such that  $\mathcal{F}$  is the set of independent sets of an arbitrary matroid  $\mathcal{M} = (\mathcal{F}, \mathcal{F})$  and  $\mathcal{C} = \{\mathcal{C}\}$ ; in OKnapMed, we instantiate ORDCLST such that each facility  $i \in \mathcal{F}$  has a weight  $\text{wt}_i \geq 0$ ,  $\mathcal{F}$  is the set of facility subsets with total weight at most  $W$ , i.e.,  $\mathcal{F} = \{F \subseteq \mathcal{F} : \sum_{i \in F} \text{wt}_i \leq W\}$  and  $\mathcal{C} = \{\mathcal{C}\}$ . It is easy to see that OMatMed and OKnapMed generalize matroid center and matroid median, knapsack center and knapsack median, respectively. Moreover, since the cardinality constraint  $|F| \leq k$  is trivially recovered by the matroid and knapsack constraints, OKMED is also generalized by OMatMed and OKnapMed.

Theorem 2 is obtained using the same reduction by Theorem 3 and similar iterative rounding algorithms as RO $k$ Med. We provide the details of the algorithms and the proofs in Appendix B and Appendix C.

► **Remark.** We remark on the difficulties of OMatMed and OKnapMed under previous methods for OKMED. The integrality gap in the natural relaxation for matroid median is a constant (see, e.g., [22]), thus it is likely that the algorithm by Byrka et al. [5] could provide a constant-factor approximation for OMatMed after some modifications; this would hardly be surprising since our reduction algorithm also gives a simpler analysis for OMatMed, compared with RO $k$ Med. For OKnapMed, however, it appears that previous methods will fail due to the unbounded integrality gap in the natural relaxation for knapsack median (see, e.g., [24]); our reduction framework can circumvent this issue analogously to RO $k$ Med.

## 4 Conclusion

In this paper, we present a reduction framework for a class of clustering problems with ordered objectives, which preserves the approximation guarantee up to constant factors. This leads to the first polynomial-time constant-factor approximation algorithms for three natural clustering problems, namely, robust ordered  $k$ -median, ordered matroid median and ordered knapsack median. We find the problem of robust ordered  $k$ -median particularly interesting, since its objective exhibits a certain unimodal shape, which can be nicely motivated by real-world applications.

We list some open questions here that we find interesting.

- Our reduction framework is based upon the sparsification methods proposed by Aouad and Segev [1] and Byrka et al. [5]. On the ordered objective and symmetric monotone norms in general, there have been other approximation methods, e.g., [8, 19]. It would be interesting to see whether our approximation guarantees can be improved by leveraging other techniques and ideas in the literature.
- Although the objective of robust ordered  $k$ -median is distinctly unimodal in its shape (see Figure 1), there are still more general unimodal objective functions that are not captured by it. Obtaining an approximation algorithm for arbitrary unimodal vectors, even with an  $O(\log n)$ -factor approximation guarantee, is beyond the scope of our current framework, so it might require some brand new ideas to handle these objectives.

## References

- 1 Ali Aouad and Danny Segev. The ordered  $k$ -median problem: surrogate models and approximation algorithms. *Math. Program.*, 177(1-2):55–83, 2019. doi:10.1007/s10107-018-1259-3.
- 2 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for  $k$ -median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.
- 3 Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Joachim Spoerhase, Aravind Srinivasan, and Khoa Trinh. An improved approximation algorithm for knapsack median using sparsification. *Algorithmica*, 80(4):1093–1114, 2018. doi:10.1007/s00453-017-0294-4.
- 4 Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for  $k$ -median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017. doi:10.1145/2981561.
- 5 Jaroslaw Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered  $k$ -median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 620–631, 2018. doi:10.1145/3188745.3188930.
- 6 Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform  $k$ -center problem. *ACM Trans. Algorithms*, 16(4):46:1–46:19, 2020. doi:10.1145/3392720.
- 7 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between  $k$ -median and  $k$ -center: Approximation algorithms for ordered  $k$ -median. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs*, pages 29:1–29:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.29.
- 8 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 126–137, 2019. doi:10.1145/3313276.3316322.
- 9 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the  $k$ -median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002. doi:10.1006/jcss.2002.1882.
- 10 Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms*, pages 642–651, 2001. URL: <http://dl.acm.org/citation.cfm?id=365411.365555>.
- 11 Moses Charikar and Shi Li. A dependent LP-rounding approach for the  $k$ -median problem. In *Automata, Languages, and Programming - 39th International Colloquium, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 194–205, 2012. doi:10.1007/978-3-642-31594-7\_17.
- 12 Danny Z. Chen, Jian Li, Hongyu Liang, and Haitao Wang. Matroid and knapsack center problems. *Algorithmica*, 75(1):27–52, 2016. doi:10.1007/s00453-015-0010-1.
- 13 Ke Chen. A constant factor approximation algorithm for  $k$ -median clustering with outliers. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 826–835, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347173>.
- 14 Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for  $k$ -median. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms*, pages 1556–1612. SIAM, 2022. doi:10.1137/1.9781611977073.65.
- 15 Jack R. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26. Springer, 2001. doi:10.1007/3-540-36478-1\_2.
- 16 Anupam Gupta, Benjamin Moseley, and Rudy Zhou. Structural iterative rounding for generalized  $k$ -median problems. In *48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *LIPIcs*, pages 77:1–77:18, 2021. doi:10.4230/LIPIcs.ICALP.2021.77.



- 17 David G. Harris, Thomas W. Pensyl, Aravind Srinivasan, and Khoa Trinh. A lottery model for center-type problems with outliers. *ACM Transactions on Algorithms*, 15(3):36:1–36:25, 2019. doi:10.1145/3311953.
- 18 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.
- 19 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *LIPIcs*, pages 81:1–81:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.81.
- 20 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 731–740, 2002. doi:10.1145/509907.510012.
- 21 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and  $k$ -median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.
- 22 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1130, 2011. doi:10.1137/1.9781611973082.84.
- 23 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for  $k$ -median and  $k$ -means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 646–659, 2018. doi:10.1145/3188745.3188882.
- 24 Amit Kumar. Constant factor approximation algorithm for the knapsack median problem. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 824–832, 2012. doi:10.1137/1.9781611973099.66.
- 25 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.
- 26 Shi Li and Ola Svensson. Approximating  $k$ -median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016. doi:10.1137/130938645.
- 27 Chaitanya Swamy. Improved approximation algorithms for matroid and knapsack median problems and applications. *ACM Trans. Algorithms*, 12(4):49:1–49:22, 2016. doi:10.1145/2963170.

## A Missing Proofs for Robust Ordered $k$ -Median

**Proof of Theorem 7.** Let us first assume we know  $(F^*, C^*)$ ; we will remedy this assumption after the construction of  $\mathcal{J}'$ . Recall that  $U \in [V^*, (1 + \epsilon)V^*)$  and  $V^* = \sum_{j \in C^*} f(c_j^*)$ .

Set  $S_0 = \emptyset$ ,  $\mathcal{C}' = \mathcal{C}$ . Whenever there exists  $i \in F^* \setminus S_0$  such that  $\sum_{j \in C^*: \kappa_j^* = i} f(c_j^*) \geq \rho U$ , we set  $S_0 \leftarrow S_0 \cup \{i\}$ . This process can be repeated for at most  $O(1/\rho)$  times, because the subsets of clients assigned to the facilities in  $S_0$  are mutually disjoint by the definition of  $\kappa_j^*$ , and the overall sum of  $f$  values is at most  $V^* \leq U$ . The remaining facilities in  $F^* \setminus S_0$  will always satisfy (7.2) because we will only add facilities to  $S_0$  and remove clients from  $\mathcal{C}'$ .

Next, we put  $C'^* = C^* \cap \mathcal{C}'$  at all times. Whenever there exists  $p \in \mathcal{F} \cup \mathcal{C}'$  such that  $|\text{Ball}_{C'^*}(p, \delta c_p^*)| \cdot f((1 - \delta)c_p^*) \geq \rho U$ , set  $\mathcal{C}' \leftarrow \mathcal{C}' \setminus \text{Ball}_{\mathcal{C}'}(p, \delta c_p^*)$  and  $S_0 \leftarrow S_0 \cup \{\kappa_p^*\}$ . Each removed client  $j$  is from  $\mathcal{C}'$  and satisfies  $f(c_j^*) \geq f(c_p^* - d(j, p)) \geq f((1 - \delta)c_p^*)$  using the triangle inequality. Thus the total  $f$  value removed is at least  $|\text{Ball}_{C'^*}(p, \delta c_p^*)| \cdot f((1 - \delta)c_p^*) \geq \rho U$ . Using a similar argument, this process can also be repeated for at most  $O(1/\rho)$  times. The condition in (7.3) is then satisfied by definition.

Note that each such removed client  $j \in \mathcal{C} \setminus \mathcal{C}'$  has, by the triangle inequality again,

$$f\left(\frac{1 - \delta}{1 + \delta}d(j, S_0)\right) \leq f\left(\frac{1 - \delta}{1 + \delta}(d(j, p) + d(p, \kappa_p^*))\right) \leq f((1 - \delta)c_p^*) \leq f(c_j^*),$$



where the last inequality is because  $c_j^* \geq c_p^* - d(j, p) \geq (1 - \delta)c_p^*$ . Therefore, by summing over all  $j \in C^*$ , (7.4) follows since

$$\sum_{j \in C^* \setminus C'} f\left(\frac{1-\delta}{1+\delta}d(j, S_0)\right) + U' \leq \sum_{j \in C^* \setminus C'} f(c_j^*) + \sum_{j \in C^* \cap C'} f(c_j^*) = V^* \leq U.$$

Finally, we remove the dependence of the procedures on  $(F^*, C^*)$ , by noticing that  $|S_0| = O(1/\rho)$ , and  $C'$  is obtained from  $C$  by removing  $O(1/\rho)$  closed balls. Since  $m' = |C^* \cap C'|$  only takes values in  $[m]$ , the total number of possible outcomes is at most  $n_0^{O(1/\rho)}$ , and we can simply enumerate all possible configurations of  $(C', m', S_0)$ . (7.1) also follows by definition.  $\blacktriangleleft$

**Proof of Theorem 8.** We iteratively construct  $\{\hat{R}_j : j \in C'\}$  that always maintain (8.2), then prove (8.1). Initially, let  $\hat{R}_j = 0$  for each  $j \in C'$ . In each iteration  $k \geq 1$ , we try to assign the  $k$ -th largest distance  $t'$  in  $\{d(i, j) : i \in \mathcal{F}, j \in C'\} \setminus \{0\}$  *sequentially* to unassigned clients  $\{j \in C' : \hat{R}_j = 0\}$  without violating (8.2); it is easy to verify that (8.2) is always maintained, since it suffices to consider the case of  $t = t'$  for each  $p \in \mathcal{F} \cup C'$  (cf. [23]).

Recall that  $C'^* = C^* \cap C'$ . We construct a one-to-one mapping  $\phi : C'^* \rightarrow C'$  and show the solution  $(F^*, \phi(C'^*))$  satisfies (8.1). Initially, we let  $\phi$  be the identity function on  $C'^*$ . Consider the clients in  $\{j \in C'^* : c_j^* > (1 + 3\delta/4)\hat{R}_j\}$  in non-decreasing order of  $c_j^*$ . For each such  $j$ , we want to update  $\phi(j)$  to an “unused” client in the current  $C' \setminus \phi(C'^*)$  such that  $d(\phi(j), j) \leq \delta c_j^*/2$  and  $\hat{R}_{\phi(j)} \geq c_j^*$ . If such  $\phi(j)$  exists for each  $j \in C'^*$ , we assign  $\phi(j)$  to  $\kappa_j^*$ , define  $\kappa'_{\phi(j)} = \kappa_j^*$  and thus  $c'_{\phi(j)} = d(\phi(j), \kappa_j^*) \leq c_j^* + d(j, \phi(j)) \leq (1 + \delta/2)c_j^* \leq (1 + \delta/2)\hat{R}_{\phi(j)}$ . Moreover, one has

$$\forall i \in F^* \setminus S_0, \quad \sum_{j \in \phi(C'^*) : \kappa'_j = i} f\left(\frac{2}{2+\delta}c'_j\right) \leq \sum_{j \in C'^* : \kappa_j^* = i} f(c_j^*) \leq \rho U,$$

where the last inequality is due to (7.2). Similarly, one has

$$\sum_{j \in \phi(C'^*)} f\left(\frac{2}{2+\delta}c'_j\right) = \sum_{j \in C'^*} f\left(\frac{2}{2+\delta}c'_{\phi(j)}\right) \leq \sum_{j \in C'^*} f(c_j^*) = U',$$

therefore (8.1) is satisfied by  $(F^*, \phi(C'^*))$  in this case.

It remains to show that such an unused  $j' \in C' \setminus \phi(C'^*)$  can always be found for each  $j \in C'^*$  with  $c_j^* > (1 + 3\delta/4)\hat{R}_j$ . Notice that we have  $\hat{R}_j = 0$  when  $t' = c_j^*$  is considered during the construction, so setting  $\hat{R}_j = c_j^*$  would be a violation in (8.2); that is, there exists  $p \in \mathcal{F} \cup C'$  such that  $d(p, j) \leq \delta c_j^*/4$  and the set  $H_j = \{j' \in \text{Ball}_{C'}(p, \delta c_j^*/4) : \hat{R}_{j'} \geq c_j^*\}$  satisfies

$$|H_j \cup \{j\}| = |H_j| + 1 > \frac{\rho U}{f((1-\delta)(1-\delta/4)c_j^*)}. \quad (7)$$

Further, if there exists some  $j' \in H_j \setminus \phi(C'^*)$ , we can set  $\phi(j) = j'$  since  $\hat{R}_{j'} \geq c_j^*$  and  $d(j, j') \leq d(j', p) + d(p, j) \leq \delta c_j^*/2$  using the triangle inequality. Therefore, it suffices to prove  $H_j \not\subseteq \phi(C'^*)$ .

For the sake of contradiction, assume  $H_j \subseteq \phi(C'^*)$  when we want to update  $\phi(j)$ . For each  $\phi(\hat{j}) \in H_j$ , we have  $d(p, \hat{j}) \leq d(p, \phi(\hat{j})) + d(\hat{j}, \phi(\hat{j})) \leq 3c_j^*/4$ , because we consider the clients in non-decreasing order of  $c_j^*$  and hence  $d(\hat{j}, \phi(\hat{j})) \leq \delta c_j^*/2 \leq \delta c_j^*/2$  in earlier

iterations. This shows that (currently)  $\phi^{-1}(H_j) \subseteq \text{Ball}_{C'^*}(p, 3\delta c_j^*/4)$ . We further have  $c_p^* \geq c_j^* - d(p, j) \geq (1 - \delta/4)c_j^*$  and  $\delta c_p^* \geq \delta(1 - \delta/4)c_j^* \geq (3\delta/4)c_j^*$  as  $\delta < 1$ . Thus  $\phi^{-1}(H_j) \subseteq \text{Ball}_{C'^*}(p, 3\delta c_j^*/4) \subseteq \text{Ball}_{C'^*}(p, \delta c_p^*)$ . Because  $j \notin \phi^{-1}(H_j)$  and  $j \in \text{Ball}_{C'^*}(p, \delta c_p^*)$ , we have

$$\begin{aligned} |\text{Ball}_{C'^*}(p, \delta c_p^*)| \cdot f((1 - \delta)c_p^*) &\geq (|H_j| + 1)f((1 - \delta)c_p^*) \\ &\geq (|H_j| + 1)f((1 - \delta)(1 - \delta/4)c_j^*) >_{(7)} \rho U, \end{aligned}$$

which is a contradiction to (7.3). ◀

## B Ordered Matroid Median

In this section, we give the first constant-factor approximation for **OMatMed**. As is pointed out in [23], the natural LP relaxation for matroid median has a small integrality gap; we skip the pre-processing steps of **ROkMed** and provide a sketch on how the iterative rounding algorithm directly outputs the desired approximate solution. Suppose we have a faithful function  $f$  in what follows via Theorem 3 and exhaustive search.

1. Ignore the pre-processing steps (i.e., Theorem 7 and Theorem 8) in the **ROkMed** algorithm. To obtain a natural relaxation for any reduced instance  $\mathcal{J}_{f_\lambda}$ , replace the cardinality constraint  $\sum_{i \in \mathcal{F}} y_i \leq k$  in  $\text{LP}(f_\lambda)$  with  $\sum_{i \in S} y_i \leq r_{\mathcal{M}}(S)$ ,  $\forall S \subseteq \mathcal{F}$ ; here,  $r_{\mathcal{M}}$  is the rank function of the given matroid  $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ , and these matroid polytope constraints follow from a classic result by Edmonds [15].
2. We *no longer* have the stronger relaxation as the **ROkMed** case does. We solve the natural relaxation and proceed to the auxiliary LP, which is similar to  $\text{A-LP}(f_{\lambda_2})$  except for the matroid constraints; because each client is fully assigned to an extent of 1 in **OMatMed**, we also remove (A-LP.6) and change (A-LP.4) to equality constraints. We use Algorithm 1 for iterative rounding; since we do not have  $S_0$  or virtual clients, Algorithm 1 starts with  $C_{\text{part}} \leftarrow \mathcal{C}$  and  $C_{\text{core}} \leftarrow \emptyset$ .
3. Because we do not have any outliers and each client will end up in  $C_{\text{full}}$ , the remaining *tight* constraints after iterative rounding are either from a partition matroid (i.e.,  $y(F_j) = 1$  for each  $j \in C_{\text{core}}$ ) or from the input matroid (i.e.,  $\sum_{i \in S} y_i \leq r_{\mathcal{M}}(S)$  for each  $S \subseteq \mathcal{F}$ ). Therefore, the corresponding output solution  $y'$  is integral [15].
4. Using the same argument as Lemma 15, the objective value of  $y'$  in the natural relaxation  $\text{LP}(f_\lambda)$  (we use the same names as **ROkMed** here with a slight abuse of notation) is bounded by the objective of  $y'$  in  $\text{A-LP}(f_{\lambda_2})$  where  $\lambda_2 = \frac{3\tau-1}{\tau-1}\lambda$ , and further bounded by the *optimal* objective of  $\text{LP}(f_{\lambda_1})$  where  $\lambda_1 = \frac{\tau(3\tau-1)}{\tau-1}\lambda$ . Similar to Lemma 9, the optimum of  $\text{LP}(f_{\lambda_1})$  is at most  $\lambda_1(1 + O(\epsilon))\text{opt}$ , so the objective of  $y'$  in  $\text{LP}(f_\lambda)$  is also at most  $\lambda_1(1 + O(\epsilon))\text{opt}$ .
5. Using Theorem 3 on the reduced instance  $\mathcal{J}_{f_\lambda}$ , the integral solution induced by  $y'$  has an approximation ratio of

$$\frac{1}{\lambda} (\lambda_1(1 + O(\epsilon)) + (1 + O(\epsilon))) = (1 + O(\epsilon)) \left( \frac{\tau(3\tau-1)}{\tau-1} + \frac{1}{\lambda} \right),$$

where we have  $\lambda \leq \frac{\tau-1}{\tau(3\tau-1)} \in (0, 5 - 2\sqrt{6}]$  because  $\lambda_1 \leq 1$  in  $\text{LP}(f_{\lambda_1})$ . Therefore, the approximation ratio is minimized when  $\lambda = \frac{\tau-1}{\tau(3\tau-1)}$  and  $\frac{\tau(3\tau-1)}{\tau-1}$  is minimized, giving

$$10 + 4\sqrt{6} + O(\epsilon) \leq 19.798 + O(\epsilon) \leq 19.8,$$

where we choose a small enough constant  $\epsilon > 0$ .

## C Ordered Knapsack Median

In this section, we give the first constant-factor approximation for OKnapMed. We closely follow the procedures in [23] and use an iterative rounding algorithm akin to ROkMed. Suppose we have a faithful function  $f$  in what follows via Theorem 3 and exhaustive search. The following two theorems are similar to Theorem 7 and Theorem 8 in ROkMed. Let  $(F^*, C^* = \mathcal{C})$  be the optimal solution to the original OKnapMed instance  $\mathcal{J}$  with optimum  $\text{opt} \geq 0$ . Recall that we guess  $U \in [V^*, (1+\epsilon)V^*]$  via binary search, where  $V^*$  is the optimum of  $\mathcal{J} = (\mathcal{F}, \mathcal{C}, d, \mathcal{F}, \mathcal{C}, f)$ ; we have  $V^* \leq (1 + O(\epsilon))\text{opt}$  using Theorem 3.

► **Theorem 17.** *Given  $\rho, \delta \in (0, 1)$  and  $U$ , there exists an  $n_0^{O(1/\rho)}$ -time algorithm that finds an extended instance  $\mathcal{J}' = (\mathcal{F}, \mathcal{C}', d, \mathcal{F}, \mathcal{C}', f, S_0)$  satisfying the following.*

(17.1)  $\mathcal{C}' \subseteq \mathcal{C}$ ,  $\mathcal{C}' = \{\mathcal{C}'\}$  and  $S_0 \subseteq F^*$  with  $|S_0| = O(1/\rho)$ .

(17.2) For each  $i \in F^* \setminus S_0$ , we have  $\sum_{j \in \mathcal{C}': \kappa_j^* = i} f(c_j^*) \leq \rho U$ ,

(17.3) For each  $p \in \mathcal{F} \cup \mathcal{C}'$ , we have  $|\text{Ball}_{\mathcal{C}'}(p, \delta c_p^*)| \cdot f((1-\delta)c_p^*) < \rho U$ ,

(17.4) Denote  $U' = \sum_{j \in \mathcal{C}'} f(c_j^*)$ . We have  $\sum_{j \in \mathcal{C} \setminus \mathcal{C}'} f\left(\frac{1-\delta}{1+\delta}d(j, S_0)\right) + U' \leq U$ .

► **Theorem 18.** *Given the instance found in Theorem 17, we can efficiently compute a set of upper bounds  $\{R_j \geq 0 : j \in \mathcal{C}'\}$  such that for each  $j \in \mathcal{C}'$ , we have*

$$c_j^* \leq R_j = \max \{R > 0 : |\text{Ball}_{\mathcal{C}'}(j, \delta R)| \cdot f((1-\delta)R) \leq \rho U\}.$$

The two theorems above are almost identical to those for ROkMed, thus we omit their proofs here. By replacing the cardinality constraint  $y(F) \leq k$  with the relaxed knapsack constraint  $\sum_{i \in \mathcal{F}} \text{wt}_i \cdot y_i \leq W$ , and removing the coverage constraint for outliers, we consider a stronger LP similar to S-LP( $f_{\lambda_1}$ ). We also use iterative rounding on an auxiliary LP similar to A-LP( $f_{\lambda_2}$ ). Using a similar argument as in Lemma 14, we see that after iterative rounding, the resulting solution  $y'$  corresponds to the intersection of a laminar family and a knapsack constraint, hence it contains at most 2 fractional variables. We now focus on obtaining an integral solution  $\hat{y}$  from  $y'$ .

► **Theorem 19.** *There exists  $\lambda > 0$  depending on  $\delta$  and  $\tau$ , such that we can efficiently compute an integral solution  $\hat{y}$  to LP( $f_\lambda$ ) (in the knapsack case), and its objective value is at most  $3\rho U$  larger than that of  $y'$ .*

**Proof.** If there is only one fractional facility  $i_2$ , we close it. If there are two, suppose  $i_1, i_2$  are the two fractional facilities and  $i_1$  is the one with a smaller weight; because  $y'$  is a basic feasible solution, we again have  $y'_{i_1} + y'_{i_2} = 1$ ; we fully open  $i_1$  and close  $i_2$ . The set of open facilities  $\hat{F}$  is similarly defined as in Theorem 16. Because  $\text{wt}_{i_1} \leq \text{wt}_{i_2}$ , it is also easy to verify that  $\sum_{i \in \hat{F}} \text{wt}_i \leq W$ , thus  $\hat{F}$  is indeed a feasible solution.

Unlike ROkMed, each client is fully assigned, so it remains to bound the cost incurred from re-assigning clients that were assigned to  $i_2$ , that is,  $J = \{j \in \mathcal{C}_{\text{full}} : i_2 \in B_j\}$ . Let  $\gamma > 0$  be a constant that we determine later,  $i^*$  be the nearest open facility to  $i_2$  in  $\hat{F}$  and  $t' = d(i_2, i^*)$ . Let  $J_1 = \{j \in J : d(j, i_2) > \gamma t'\}$  and  $J_2 = \{j \in J : d(j, i_2) \leq \gamma t'\}$ . For  $j \in J_1$ , we have  $d(j, i^*) \leq d(j, i_2) + d(i_2, i^*) < (1 + \frac{1}{\gamma})d(j, i_2)$ , thus

$$\sum_{j \in J_1} f(\lambda d(j, i^*)) \leq \sum_{j \in J_1} f(d(j, i_2)) \leq 2\rho U \Leftarrow \lambda \leq \frac{\gamma}{1+\gamma}. \quad (8)$$

## 34:22 Ordered $k$ -Median with Outliers

Fix some  $j \in J_2$ . Similar as before, we have  $R_j \geq D_{l_j}/\tau \geq \frac{\tau-1}{\tau(3\tau-1)}(t' - d(j, i_2)) \geq \frac{\tau-1}{\tau(3\tau-1)}(1 - \gamma)t'$ . Suppose  $\delta R_j \geq 2\gamma t'$ , then by Theorem 18 and the triangle inequality,

$$|J_2| \leq |\text{Ball}_{\mathcal{C}'}(i_2, \gamma t')| \leq |\text{Ball}_{\mathcal{C}'}(j, 2\gamma t')| \leq |\text{Ball}_{\mathcal{C}'}(j, \delta R_j)| \leq \frac{\rho U}{f((1 - \delta)R_j)}.$$

Using the triangle inequality again, we have  $d(j, i^*) \leq (1 + \gamma)t'$  and the following total cost of assigning  $J_2$  to  $i^*$  is at most

$$\sum_{j \in J_2} f(\lambda d(j, i^*)) \leq |J_2| f(\lambda(1 + \gamma)t') \leq \rho U \Leftarrow \lambda \leq (1 - \delta) \cdot \frac{\tau - 1}{\tau(3\tau - 1)} \cdot \frac{1 - \gamma}{1 + \gamma}. \quad (9)$$

We let  $\sigma = \frac{\tau-1}{\tau(3\tau-1)}$  and let  $\gamma = \frac{\delta\sigma}{2+\delta\sigma}$  so that  $\delta R_j \geq 2\gamma t'$ . By letting  $\lambda$  be the minimum of (8)(9) and summing over the two cases, the increase of objective value w.r.t.  $\text{LP}(f_\lambda)$  is at most  $2\rho U + \rho U = 3\rho U$ , thus the theorem follows.  $\blacktriangleleft$

Let  $\delta = 2/3$  and thus  $\lambda = \frac{\sigma}{3+2\sigma}$ . Similar to Lemma 9, the objective value of  $\text{S-LP}(f_{\lambda_1})$ ,  $\lambda_1 = \frac{\tau(3\tau-1)}{\tau-1}\lambda$  is at most  $\lambda_1 U'$ . Using the same argument as Lemma 15, the objective value of  $y'$  in the original relaxation  $\text{LP}(f_\lambda)$  is at most that of  $\text{A-LP}(f_{\lambda_2})$ ,  $\lambda_2 = \frac{3\tau-1}{\tau-1}\lambda$ , which can be bounded by  $\lambda_1 U'$  akin to Lemma 13. Using Theorem 19, the objective of  $\hat{y}$  to  $\text{LP}(f_\lambda)$  is at most  $\lambda_1 U' + 3\rho U$ . Finally, using (17.4) and similarly to (6), the approximation ratio is

$$\left( \max \left\{ 5\lambda, \frac{\tau(3\tau-1)}{\tau-1}\lambda \right\} + 1 + O(\rho) \right) \frac{1 + \epsilon}{\lambda} = \left( \frac{\lambda}{\sigma} + 1 + O(\rho) \right) \frac{1 + \epsilon}{\lambda} = \frac{1}{\sigma} + \frac{1}{\lambda} + O(\epsilon + \rho).$$

By letting  $\tau = 1 + \sqrt{\frac{2}{3}}$  and  $\sigma = 5 - 2\sqrt{6}$ , the approximation ratio is at most

$$\frac{4}{\sigma} + 2 + O(\epsilon + \rho) = 22 + 8\sqrt{6} + O(\epsilon + \rho) \leq 41.596 + O(\epsilon + \rho) \leq 41.6,$$

where one chooses  $\epsilon$  and  $\rho$  that are small enough. The running time is obtained from the enumeration process and bounded by a polynomial.

# Sketching Approximability of (Weak) Monarchy Predicates

**Chi-Ning Chou**

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

**Alexander Golovnev**

Department of Computer Science, Georgetown University, Washington, D.C., USA

**Amirbehshad Shahrashbi**

Microsoft, Redmond, WA, USA

**Madhu Sudan**

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA USA

**Santhoshini Velusamy**

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

---

## Abstract

We analyze the sketching approximability of constraint satisfaction problems on Boolean domains, where the constraints are balanced linear threshold functions applied to literals. In particular, we explore the approximability of monarchy-like functions where the value of the function is determined by a weighted combination of the vote of the first variable (the president) and the sum of the votes of all remaining variables. The pure version of this function is when the president can only be overruled by when all remaining variables agree. For every  $k \geq 5$ , we show that CSPs where the underlying predicate is a pure monarchy function on  $k$  variables have no non-trivial sketching approximation algorithm in  $o(\sqrt{n})$  space. We also show infinitely many weaker monarchy functions for which CSPs using such constraints are non-trivially approximable by  $O(\log(n))$  space sketching algorithms. Moreover, we give the first example of sketching approximable asymmetric Boolean CSPs. Our results work within the framework of Chou, Golovnev, Sudan, and Velusamy (FOCS 2021) that characterizes the sketching approximability of all CSPs. Their framework can be applied naturally to get a computer-aided analysis of the approximability of any specific constraint satisfaction problem. The novelty of our work is in using their work to get an analysis that applies to *infinitely* many problems simultaneously.

**2012 ACM Subject Classification** Theory of computation → Sketching and sampling; Theory of computation → Approximation algorithms analysis

**Keywords and phrases** sketching algorithms, approximability, linear threshold functions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.35

**Category** APPROX

**Related Version** Full Version: <https://arxiv.org/abs/2205.02345>

**Funding** *Chi-Ning Chou*: Partially supported by NSF grants DMS-2134157 and CCF-1565264, DOE grant DE-SC0022199, and the Simons foundation.

*Amirbehshad Shahrashbi*: The author was with Harvard University and supported by CRA-CCC Computing Innovations Fellowship (CIFellowship 2020) during this work.

*Madhu Sudan*: Supported in part by a Simons Investigator Award and NSF Awards CCF 1715187 and CCF 2152413.

*Santhoshini Velusamy*: Supported in part by a Google Ph.D. Fellowship, a Simons Investigator Award to Madhu Sudan, and NSF Awards CCF 1715187 and CCF 2152413.

**Acknowledgements** We thank the anonymous reviewers for their helpful and constructive comments.



© Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrashbi, Madhu Sudan, and Santhoshini Velusamy;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 35; pp. 35:1–35:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

In this paper we consider the sketching complexity of solving constraint satisfaction problems (CSPs) approximately where the constraints are given by linear threshold functions over a collection of Boolean literals. We introduce these terms below.

### CSPs

Given a Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ , the Boolean CSP associated with  $f$ , denoted  $\text{Max-CSP}(f)$  is the following optimization problem. Given  $m$  constraints  $C_1, \dots, C_m$  on  $n$  Boolean variables  $X_1, \dots, X_n$ , where each constraint applies  $f$  to a sequence of  $k$  distinct literals from the set  $\{X_1, \dots, X_n, -X_1, \dots, -X_n\}$ , find the maximum fraction of constraints that can be satisfied by an assignment to the  $n$  variables. For an instance  $\Psi$  of  $\text{Max-CSP}(f)$  we use  $\text{val}_\Psi$  to denote this maximum value. We are interested in approximating  $\text{val}_\Psi$  and this task is known to be equivalent to solving a gapped decision version of  $\text{Max-CSP}(f)$ . For  $0 \leq \beta < \gamma \leq 1$  we define the  $(\gamma, \beta)$ -gapped version of  $\text{Max-CSP}(f)$ , abbreviated to  $(\gamma, \beta)$ - $\text{Max-CSP}(f)$ , to be the following promise decision problem: Given an instance  $\Psi$  satisfying  $\text{val}_\Psi \geq \gamma$  or  $\text{val}_\Psi < \beta$  decide which one of the two conditions holds.

### Sketching algorithms

The class of algorithms we consider (and rule out) are randomized sketching algorithms. Inputs to these algorithms arrive as a stream of elements, in our case a stream of constraints. We consider algorithms that use some bounded amount of space, denoted  $s(n)$ , to process the stream and maintain a sketch of their output. When the stream ends the algorithm outputs its verdict based on the current sketch. A key restriction of a sketching algorithm is that its sketch should satisfy the following composability property. Given two streams  $\sigma$  and  $\tau$  and a fixing of the randomness, the sketch of their concatenation  $S(\sigma \circ \tau)$  should be determined by their sketches  $S(\sigma)$  and  $S(\tau)$  alone.<sup>1</sup> Most existing algorithms for streaming CSPs are sketching algorithms. We say a sketching algorithm solves a (gapped) decision problem if on every input its answer is correct with probability at least  $2/3$ .

### Approximability and approximation resistance

For  $\alpha \in [0, 1]$ , we say an algorithm is an  $\alpha$ -approximation algorithm for  $\text{Max-CSP}(f)$  if the following holds: on every input instance  $\Psi$ , the algorithm outputs  $v$  such that  $\alpha \cdot \text{val}_\Psi \leq v \leq \text{val}_\Psi$  with probability at least  $2/3$ . Note that the existence of an  $\alpha$ -approximation algorithm is equivalent to the existence of an algorithm for solving  $(\gamma, \beta)$ - $\text{Max-CSP}(f)$  for every  $\gamma, \beta \in [0, 1]$  with  $\beta \leq \alpha \cdot \gamma$ .

For a function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ , define  $\rho(f) = 2^{-k} \cdot |\{x \in \{-1, 1\}^k \mid f(x) = 1\}|$ . For every  $f$  and every instance  $\Psi$  of  $\text{Max-CSP}(f)$ , a random assignment satisfies  $\rho(f)$  fraction of the constraints in expectation and so every  $\Psi$  satisfies  $\text{val}_\Psi \geq \rho(f)$ . Thus the  $(1, \rho(f))$ - $\text{Max-CSP}(f)$  problem is trivially solvable by the algorithm that always outputs  $\text{val}_\Psi \geq 1$  (since the set  $\{\Psi \mid \text{val}_\Psi < \rho(f)\}$  is empty). We say  $\text{Max-CSP}(f)$  is sketching approximable within space  $s(n)$  if there is an  $\varepsilon > 0$  and a sketching algorithm using at most  $s(n)$  space that solves  $(1 - \varepsilon, \rho(f) + \varepsilon)$ - $\text{Max-CSP}(f)$ . We say that  $\text{Max-CSP}(f)$  is approximation resistant to space  $s(n)$  if for every  $\varepsilon > 0$ , every sketching algorithm for  $(1, \rho(f) + \varepsilon)$ - $\text{Max-CSP}(f)$  requires  $\Omega(s(n))$  space.

<sup>1</sup> In contrast, a general streaming algorithm maintains a state  $S(\sigma \circ \tau)$  that may depend on  $S(\sigma)$  and all of  $\tau$ .

## 1.1 Motivation and related work

There has been an increasing interest in studying the approximability of CSPs in the streaming setting [16, 13, 14, 8, 7, 15, 6, 4, 5, 19, 2, 3]. In particular, recently Chou, Golovnev, Sudan, and Velusamy [4, 5] gave a dichotomy result for sketching approximability of all finite CSPs. Specifically, they proved the following theorem.

► **Theorem 1** ([5]). *For every  $k$ , every predicate  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  and every  $0 \leq \beta < \gamma \leq 1$  one of the following holds: (1)  $(\gamma, \beta)$ -Max-CSP( $f$ ) is solvable by an  $O(\log(n))$ -space sketching algorithm, or (2) for every  $\varepsilon > 0$ ,  $(\gamma - \varepsilon, \beta + \varepsilon)$ -Max-CSP( $f$ ) is not solvable by any  $o(\sqrt{n})$ -space sketching algorithm. Furthermore there is a decidable procedure that determines, given  $\mathcal{F}$ ,  $\gamma$  and  $\beta$ , which of the two conditions hold.*

We note that a followup paper by the same authors [4] extends the result to a more general setting: Specifically they allow non-Boolean variables, allow a set of predicates rather than a single function; and allow the predicates to be applied to variables rather than literals. While their result is more general all results in this paper work in the more restricted setting of [5] and so we will describe our results in their language (which can be somewhat simpler for problems that are expressible in their setting).

While the results of [5] imply a dichotomy, to explicitly get the optimal sketching approximation ratio for a given predicate  $f$ , they need to solve an optimization problem which in general needs computer-aided analysis. In order to get more explicit results one needs to restrict the families of functions considered, and even then it is unclear if there can be a closed-form expression. In the only example we are aware of, Boyland, Hwang, Prasad, Singer, and Velusamy [2] gave closed-form expressions for the optimal sketching approximation ratio of some *symmetric* Boolean CSPs. This still leaves the question of exploring the sketching approximability of other subfamilies of CSPs and extracting some qualitative results yielding necessary or sufficient conditions for non-trivial approximability.

## 1.2 Main results

In this paper we study sketching approximability of CSPs on linear threshold functions. Below we define the classes of linear threshold functions and balanced linear threshold functions.

► **Definition 2** (Linear threshold function). *A linear threshold function, or LTF, is a Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  of the form*

$$f(x) = \text{sign} \left( \sum_{i=1}^k w_i x_i + \theta \right),$$

where  $w_1, \dots, w_k, \theta \in \mathbb{R}$ . The function  $\text{sign}(z)$  has value 1 if  $z > 0$  and 0 if  $z \leq 0$ ;  $w_1, \dots, w_k$  are called the weights of  $f$  and  $\theta$  is the threshold.

► **Definition 3** (Balanced linear threshold function). *A balanced linear threshold function, or balanced LTF, is an LTF with threshold 0 and the additional restriction that for every  $x \in \{-1, 1\}^k$ , we have  $\sum_{i=1}^k w_i x_i \neq 0$ . Specifically, a balanced LTF  $f$  satisfies  $f(-x) = 1 - f(x)$  for every  $x$ .*

Note that for a balanced LTF  $f$ ,  $\rho(f) = 1/2$ , and the goal of approximability is to beat this factor. Balanced LTFs form a technically important class of functions to study visavis CSP approximability. For instance Potechin [18] studies them in the polynomial time regime giving a (somewhat complex) approximation-resistant function in this class. In the sketching



setting, interest in this class of functions comes from [5, Theorem 1.3] which shows that if a function  $f$  supports one-wise independence (i.e.,  $f^{-1}$  supports a distribution on  $\{-1, 1\}^k$  that is uniform on each of the  $k$  marginals) then  $\text{Max-CSP}(f)$  is approximation resistant to  $o(\sqrt{n})$  space streaming algorithms. Balanced LTFs are the most basic class of functions that *do not* support one-wise independence and hence are not covered by this theorem. Studying this class thus offers the possibility of finding new classes of CSPs that are approximation resistant to  $o(\sqrt{n})$ -space streaming algorithms.

Our first result shows that every balanced LTF on up to 4 variables is sketching approximable. (So to search for new approximation resistant functions we need to look at functions on more variables!) We note that there are only finitely many such LTFs, but already this theorem gives the first example of an asymmetric Boolean CSP which is approximable by sketching algorithms.<sup>2</sup>

► **Theorem 4.** *For every balanced LTF  $f$  on  $k \leq 4$  variables,  $\text{Max-CSP}(f)$  is sketching approximable in  $O(\log(n))$  space.*

Our next result shows that there do exist balanced LTFs functions on 5 or more variables that are sketching approximation resistant. The specific family of functions we show this for are the “Monarchy” functions. For  $k \in \mathbb{N}$ ,  $\text{MON}_k : \{-1, 1\}^k \rightarrow \{0, 1\}$  is given by  $\text{MON}_k(x_1, \dots, x_k) = \text{sign}((k-2)x_1 + x_2 + \dots + x_k)$ . It may be easily verified that  $\text{MON}_k$  is a balanced LTF. We have the following theorem.

► **Theorem 5.** *For every  $k \geq 5$ ,  $\text{Max-CSP}(\text{MON}_k)$  is sketching approximation resistant to space  $o(\sqrt{n})$ .*

Thus we get the first examples of functions that do not support one-wise independence that are approximation resistant to space  $o(\sqrt{n})$  sketching algorithms. In fact, the theorem gives infinitely many such examples. We suspect that the Balanced LTF constructed in [18] should also be approximation-resistant but so far we don’t have a proof. The monarchy functions, by virtue of the simplicity allow a simpler analytic proof, though admittedly even in this case we do not have great intuition for the proof and do not know how to extend it to other classes of functions.

Finally we also give an infinite subclass of balanced LTFs that are approximable using  $O(\log(n))$  space. The functions we consider here are what we call “weak monarchy” functions.<sup>3</sup> For  $j \leq k \in \mathbb{N}$ , let  $\text{WMON}_{k,j} : \{-1, 1\}^k \rightarrow \{0, 1\}$  be the function given by  $\text{WMON}_{k,j}(x_1, \dots, x_k) = \text{sign}(j \cdot x_1 + x_2 + \dots + x_k)$ . It may be easily verified that when  $j + k$  is even, then  $\text{WMON}_{k,j}$  is a balanced LTF. We have

► **Theorem 6.** *For all integers  $j \geq 2$  and  $k \geq 7j^3$  such that  $k+j$  is even,  $\text{Max-CSP}(\text{WMON}_{k,j})$  is sketching approximable in  $O(\log(n))$  space. In particular, for every  $j$ , there exist infinitely many  $k$  such that  $\text{Max-CSP}(\text{WMON}_{k,j})$  is sketching approximable.*

The results above give the first examples of asymmetric Boolean CSPs for which  $\text{Max-CSP}(f)$  is sketching approximable. Again we get an infinite family of such functions.

<sup>2</sup> Note that  $\text{Max-DICUT}$  (shown to be sketching approximable in [6, 4]) is not considered a Boolean CSP in [5] since the  $\text{Max-DICUT}$  constraints are applied on variables and not on literals.

<sup>3</sup> Such functions are also sometimes called presidential type predicates [10].

## Comparison to the polynomial time regime

Hast [9] proves that (a generalization of) Theorem 23 holds in the polynomial time regime (thus, implying an analogue of Theorem 6 in the polynomial time regime). Austrin, Benabbas, and Magen [1] prove that  $\text{MON}_k$  is approximable in polynomial time, which is in sharp contrast to the result of Theorem 5 in the sketching setting. Huang and Potechin [10] show that almost all  $\text{WMON}$  predicates are approximable in polynomial time. Finally, Potechin [18] gives a balanced LTF which is (conditionally) approximation resistant in the polynomial time regime.

## Organization of the paper

We start with giving formal definitions and stating relevant previous results in Section 2. The three main theorems are proved in Section 3, Section 4, and Section 5, respectively.

## 2 Preliminaries

We use  $\mathbb{N}$ ,  $\mathbb{R}$ , and  $\mathbb{R}_{\geq 0}$  to denote the sets of all natural, real, and non-negative real numbers, respectively. We use  $[n]$  to denote the set  $\{1, \dots, n\}$ . We write vector variables in boldface, e.g.,  $\mathbf{x}$ , and we use  $x_i$  to denote their  $i$ th entry. For two vectors of the same length  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$ ,  $\mathbf{x} \odot \mathbf{y} \in \mathbb{R}^k$  denotes the entry-wise product of  $\mathbf{x}$  and  $\mathbf{y}$ . For  $p \in [0, 1]$ ,  $\text{Bern}(p)$  denotes the Bernoulli distribution taking value 1 with probability  $p$ , and value  $-1$  with probability  $1 - p$ . We adopt the convention that  $\binom{n}{k} = 0$  for  $k < 0$  or  $k > n$ . By  $\sum_{i=0}^k \binom{n}{i}$  we denote the sum  $\sum_{i=0}^k \binom{n}{i}$ .

### 2.1 Sketching approximability and approximation resistance

For a function  $f: \{-1, 1\}^k \rightarrow \{0, 1\}$ , let  $\rho(f) = 2^{-k} \cdot |\{\mathbf{a} \in \{-1, 1\}^k \mid f(\mathbf{a}) = 1\}|$  denote the probability that a uniformly random assignment of the variables satisfies  $f$ .

► **Definition 7** (Sketching approximation resistance). *For a function  $f: \{-1, 1\}^k \rightarrow \{0, 1\}$ , we say that  $f$  is sketching approximation resistant to space  $s(n)$  if for every  $\varepsilon > 0$ , every sketching algorithm for  $(1, \rho(f) + \varepsilon)$ -Max-CSP( $f$ ) requires  $\Omega(\sqrt{n})$  space.*

► **Definition 8** (Sketching approximability). *For a function  $f: \{-1, 1\}^k \rightarrow \{0, 1\}$ , we say that  $f$  is sketching approximable in space  $s(n)$  if there exist  $\varepsilon > 0$  and a sketching algorithm that solves  $(1 - \varepsilon, \rho(f) + \varepsilon)$ -Max-CSP( $f$ ) using space  $s(n)$ .*

At first glance, it seems that if  $f$  is not sketching approximation resistant then it's not necessarily sketching approximable. Nonetheless, [5] proved that every  $f$  is either approximable or approximation resistant.<sup>4</sup>

### 2.2 Characterization of approximability from [5]

In this work, we focus on CSPs that use a single function  $f$  applied to literals. Thus, we will use the machinery from [5] instead of the more general (and more notationally-heavy) version in [4]. For a distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$ , by  $\boldsymbol{\mu}(\mathcal{D})$  we denote its marginals, i.e.,  $\boldsymbol{\mu}(\mathcal{D}) = (\mu_1, \dots, \mu_k)$  where  $\mu_i = \mathbb{E}_{\mathbf{b} \sim \mathcal{D}}[b_i]$  for all  $i \in [k]$ .

<sup>4</sup> Concretely, as the sets  $K^Y, K^N$  are closed (see Lemma 10), an algorithm for  $(1, \rho(f) + \varepsilon)$ -Max-CSP( $f$ ) implies an algorithm for  $(1 - \varepsilon', \rho(f) + \varepsilon)$ -Max-CSP( $f$ ) for some  $\varepsilon' > 0$ , which in turn implies that Max-CSP( $f$ ) is approximable.

## 35:6 Sketching Approximability of (Weak) Monarchy Predicates

► **Definition 9** ([5, Definitions 2.1 and 2.2]). For  $\gamma, \beta \in \mathbb{R}$ , we define the sets of distributions  $S_\gamma^Y$  and  $S_\beta^N$  as

$$S_\gamma^Y = S_\gamma^Y(f) = \{\mathcal{D}_Y \in \Delta(\{-1, 1\}^k) \mid \mathbb{E}_{\mathbf{b} \sim \mathcal{D}_Y} [f(\mathbf{b})] \geq \gamma\}$$

and

$$S_\beta^N = S_\beta^N(f) = \{\mathcal{D}_N \in \Delta(\{-1, 1\}^k) \mid \mathbb{E}_{\mathbf{b} \sim \mathcal{D}_N} \mathbb{E}_{\mathbf{a} \sim \text{Bern}(p)^k} [f(\mathbf{b} \odot \mathbf{a})] \leq \beta, \forall p \in [0, 1]\},$$

and the sets of marginals of these distributions

$$K_\gamma^Y = K_\gamma^Y(f) = \{\mu(\mathcal{D}_Y) \mid \mathcal{D}_Y \in S_\gamma^Y\}$$

and

$$K_\beta^N = K_\beta^N(f) = \{\mu(\mathcal{D}_N) \mid \mathcal{D}_N \in S_\beta^N\}.$$

We will use the following properties of the sets  $K_\gamma^Y$  and  $K_\beta^N$ .

► **Lemma 10** ([5, Lemma 2.4]). For every  $\gamma, \beta \in [0, 1]$  the sets  $K_\gamma^Y$  and  $K_\beta^N$  are bounded, closed and convex.

With these definitions, we are ready to present the approximability criteria from [5].<sup>5</sup>

► **Theorem 11** ([5, Corollary 1.2]). For every  $k \in \mathbb{N}$  and every function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ , if  $K_1^Y(f) \cap K_{\rho(f)}^N(f) = \emptyset$ , then  $f$  is sketching approximable within space  $O(\log(n))$ , if  $K_1^Y(f) \cap K_{\rho(f)}^N(f) \neq \emptyset$ , then  $f$  is sketching approximation resistant to space  $o(\sqrt{n})$ .

### 2.3 (Weak) Monarchy functions

► **Definition 12.** A monarchy predicate on  $k \geq 2$  variables  $\text{MON}_k : \{-1, 1\}^k \rightarrow \{0, 1\}$  is defined as

$$\text{MON}_k(x_1, \dots, x_k) = \text{sign} \left( (k-2)x_1 + \sum_{i=2}^k x_i \right).$$

Here  $x_1$  is commonly referred to as the president and the rest of  $x_i$ s are called citizens.

► **Definition 13** (Weak monarchy functions). A weak monarchy predicate of order  $j$  on  $k \geq 2$  variables  $\text{WMON}_{k,j} : \{-1, 1\}^k \rightarrow \{0, 1\}$  is defined as

$$\text{WMON}_{k,j}(x_1, \dots, x_k) = \text{sign} \left( j \cdot x_1 + \sum_{i=2}^k x_i \right).$$

Similar to ordinary monarchy functions,  $x_1$  is commonly referred to as the president and the rest of  $x_i$ s are called citizens.

It is straightforward to see that  $\text{MON}_k$  is a balanced LTF for every  $k \geq 2$  and  $\text{WMON}_{k,j}$  is a balanced LTF whenever  $k+j$  is even.

<sup>5</sup> Strictly speaking the statement in Corollary 1.2 in [5] is somewhat different, but their proof of Corollary 1.2 asserts this explicitly.

## 2.4 Fourier analysis of Boolean functions

We will need the following basic notions from Fourier analysis over the Boolean hypercube (see, for instance, [17]).

► **Definition 14** (Characteristic functions). *For every  $S \subseteq [k]$  such that  $|S| \geq 1$ , the characteristic function  $\chi_S : \{-1, 1\}^k \rightarrow \{-1, 1\}$  is defined as  $\chi_S(x) = \prod_{i \in S} x_i$ . The characteristic function corresponding to the empty set is defined as the constant function  $\chi_\emptyset(x) = 1$  for all  $x \in \{-1, 1\}^k$ .*

► **Definition 15** (Fourier expansions). *The Fourier expansion of a Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  is given by*

$$f = \sum_{S \subseteq [k]} \hat{f}(S) \cdot \chi_S,$$

where  $\hat{f}(S) = \mathbb{E}_{x \sim \text{Unif}(\{-1, 1\}^k)}[f(x) \cdot \chi_S(x)]$  and  $\text{Unif}(\{-1, 1\}^k)$  denotes the uniform distribution on  $\{-1, 1\}^k$ .

► **Definition 16** (Chow parameters). *The Chow parameters of a Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  are the degree-0 Fourier coefficient and the  $k$  degree-1 Fourier coefficients of  $f$ , i.e.,  $\hat{f}(\emptyset), \hat{f}(\{1\}), \dots, \hat{f}(\{k\})$ .*

► **Proposition 17.** *For every Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ ,*

1.  $\rho(f) = \hat{f}(\emptyset)$ ,
2. for every  $S \subseteq [k]$ ,  $|\hat{f}(S)| \leq \hat{f}(\emptyset)$ , and
3. for every  $x \in \{-1, 1\}^k$ ,  $-\hat{f}(\emptyset) \cdot k \leq \sum_{i=1}^k \hat{f}(\{i\}) \cdot x_i \leq \hat{f}(\emptyset) \cdot k$ .

**Proof.** The first statement of the proposition follows directly from the definition of  $\rho(f)$ :  $\rho(f) = \mathbb{E}_{x \sim \text{Unif}(\{-1, 1\}^k)}[f(x)] = \hat{f}(\emptyset)$ . For the second statement, observe that for all  $S \subseteq [k]$ ,

$$\begin{aligned} |\hat{f}(S)| &= |\mathbb{E}_{x \sim \text{Unif}(\{-1, 1\}^k)}[f(x) \cdot \chi_S(x)]| \\ &\leq \mathbb{E}_{x \sim \text{Unif}(\{-1, 1\}^k)}[|f(x) \cdot \chi_S(x)|] \\ &= \mathbb{E}_{x \sim \text{Unif}(\{-1, 1\}^k)}[f(x)] \\ &= \hat{f}(\emptyset). \end{aligned}$$

It immediately follows that for all  $x \in \{-1, 1\}^k$ ,

$$\left| \sum_{i=1}^k \hat{f}(\{i\}) \cdot x_i \right| \leq \sum_{i=1}^k |\hat{f}(\{i\}) \cdot x_i| \leq \hat{f}(\emptyset) \cdot k. \quad \blacktriangleleft$$

## 3 Approximability of Balanced LTFs on 4 variables

In this section, we show that all balanced LTFs on at most 4 variables are sketching approximable in  $O(\log(n))$  space. We start by proving that  $\text{Max-CSP}(\text{MON}_4)$  is approximable.

### 3.1 Approximability of $\text{MON}_4$

Recall that by Theorem 11, it suffices to show that  $K_1^Y(\text{MON}_4) \cap K_{1/2}^N(\text{MON}_4) = \emptyset$ . For  $k \geq 2$ , the inputs  $x_2, \dots, x_k$  are symmetric, and we will only consider distributions  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  where all vectors having the same sum of coordinates and the same value in the first

coordinate have the same probability masses. Concretely, for  $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^k$ , if  $x_1 = y_1$  and  $\sum_i x_i = \sum_i y_i$ , then  $\mathcal{D}(x) = \mathcal{D}(y)$ . Such a distribution  $\mathcal{D}$  is uniquely specified by a pair of vectors  $\mathbf{u} = (u_0, \dots, u_{k-1}), \mathbf{v} = (v_0, \dots, v_{k-1}) \in \mathbb{R}_{\geq 0}^k$  with  $\sum_i u_i + v_i = 1$ , where for  $0 \leq i \leq k-1$ ,

$$\begin{aligned} u_i &= \Pr\{x_1 = 1 \text{ and exactly } i \text{ of the rest of } x_i\text{'s are } 1\}, \\ v_i &= \Pr\{x_1 = -1 \text{ and exactly } i \text{ of the rest of } x_i\text{'s are } 1\}. \end{aligned}$$

Note that when  $\sum_i u_i + v_i = 1$ ,  $\mathbf{u}, \mathbf{v}$  define a distribution  $\mathcal{D}$  with marginals  $\boldsymbol{\mu}(\mathcal{D}) = (\mu_1, \mu', \dots, \mu')$  where

$$\mu_1 = \sum_{i=0}^{k-1} (u_i - v_i) \text{ and } \mu' = \sum_{i=0}^{k-1} \left(\frac{2i}{k-1} - 1\right)(u_i + v_i). \quad (1)$$

Next we show that for  $\text{MON}_k$  functions, restricting our attention to this class of distributions is without loss of generality.

► **Definition 18.** For  $\gamma, \beta \in \mathbb{R}$  and  $k \geq 2$ ,

$$\begin{aligned} \tilde{K}_\gamma^Y(\text{MON}_k) &= \{(\mu_1, \mu') \mid (\mu_1, \mu', \dots, \mu') \in K_\gamma^Y(\text{MON}_k)\} \\ \text{and } \tilde{K}_\beta^N(\text{MON}_k) &= \{(\mu_1, \mu') \mid (\mu_1, \mu', \dots, \mu') \in K_\beta^N(\text{MON}_k)\}. \end{aligned}$$

► **Lemma 19.** For  $\gamma, \beta \in \mathbb{R}$  and  $k \geq 2$ ,

$$K_\gamma^Y(\text{MON}_k) \cap K_\beta^N(\text{MON}_k) = \emptyset \text{ if and only if } \tilde{K}_\gamma^Y(\text{MON}_k) \cap \tilde{K}_\beta^N(\text{MON}_k) = \emptyset.$$

**Proof.** First, if  $(\mu_1, \mu', \dots, \mu') \in \tilde{K}_\gamma^Y(\text{MON}_k) \cap \tilde{K}_\beta^N(\text{MON}_k)$ , then by Definition 18,  $(\mu_1, \mu', \dots, \mu') \in K_\gamma^Y(\text{MON}_k) \cap K_\beta^N(\text{MON}_k)$ .

For the other direction. Assume that there is a vector  $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_k) \in K_\gamma^Y(\text{MON}_k) \cap K_\beta^N(\text{MON}_k)$ . Consider two distributions  $\mathcal{D}_Y \in S_\gamma^Y$  and  $\mathcal{D}_N \in S_\beta^N$  yielding the vector  $\boldsymbol{\mu} = \boldsymbol{\mu}(\mathcal{D}_Y) = \boldsymbol{\mu}(\mathcal{D}_N)$ . Given that the variables  $x_2, \dots, x_k$  are symmetric, any distribution that is yielded by permuting  $x_2, \dots, x_k$  in  $\mathcal{D}_Y$  (or  $\mathcal{D}_N$ ) is also in  $S_\gamma^Y$  (or  $S_\beta^N$ ). Note that the marginals of these distributions are also permutations of  $\boldsymbol{\mu}$ . By Lemma 10,  $K_\gamma^Y$  and  $K_\beta^N$  are convex, so they also contain the averages of these vectors:  $(\mu_1, \mu', \dots, \mu') \in K_\gamma^Y(\text{MON}_k) \cap K_\beta^N(\text{MON}_k)$  for  $\mu' = (\mu_2 + \dots + \mu_k)/(k-1)$ . Finally, by Definition 18,  $(\mu_1, \mu') \in \tilde{K}_\gamma^Y(\text{MON}_k) \cap \tilde{K}_\beta^N(\text{MON}_k)$ . ◀

Next, we characterize the set  $\tilde{K}_1^Y(\text{MON}_k)$ .

► **Lemma 20.** For every  $k \geq 2$ ,  $\tilde{K}_1^Y(\text{MON}_k) = \{(\mu_1, \mu') \in [-1, 1]^2 : \mu_1(k-2) + \mu'(k-1) \geq 1\}$ .

**Proof.** For  $\mu_1, \mu' \in [-1, 1]$  satisfying  $\mu_1(k-2) + \mu'(k-1) \geq 1$ , consider the distribution  $\mathcal{D}_Y$  given by  $u_1 = \frac{(k-1)(1-\mu')}{2(k-2)}$ ,  $u_{k-1} = \frac{(k-1)\mu' + (k-2)\mu_1 - 1}{2(k-2)}$ ,  $v_{k-1} = (1 - \mu_1)/2$ , and  $u_i = 0$  for  $i \notin \{1, k-1\}$  and  $v_j = 0$  for  $j \neq k-1$ . Note that  $u_1, v_{k-1} \geq 0$  from  $\mu_1, \mu' \in [-1, 1]$ , and  $u_{k-1} \geq 0$  from  $\mu_1(k-2) + \mu'(k-1) \geq 1$ . It is also easy to check that  $u_1 + u_{k-1} + v_{k-1} = 1$  which implies that  $\mathcal{D}_Y$  is a distribution, and that it is supported on the preimages of 1 under  $\text{MON}_k$ . Therefore  $(\mu_1, \mu') \in \tilde{K}_1^Y(\text{MON}_k)$ .

For the other direction, a distribution  $\mathcal{D}_Y$  supported on the preimages of 1 under  $\text{MON}_k$  satisfies  $u_1 + \dots + u_{k-1} + v_{k-1} = 1$ . Then, from (1),

$$\begin{aligned} \mu_1(k-2) + \mu'(k-1) &= (k-2) \sum_{i=0}^{k-1} (u_i - v_i) + \sum_{i=0}^{k-1} (2i - k + 1)(u_i + v_i) \\ &= \sum_{i=1}^{k-1} (2i-1)u_i + v_{k-1} \\ &\geq \sum_{i=1}^{k-1} u_i + v_{k-1} = 1, \end{aligned}$$

where the second equality uses that  $u_0 = 0$  and  $v_j = 0$  for  $j < k-1$ . This concludes the proof of the lemma.  $\blacktriangleleft$

Now we show that for the  $\text{MON}_4$  function,  $\tilde{K}_1^Y$  and  $\tilde{K}_{1/2}^N$  are disjoint, and, thus,  $\text{MON}_4$  is approximable in  $O(\log(n))$  space.

► **Lemma 21.** *Max-CSP( $\text{MON}_4$ ) is sketching approximable in  $O(\log(n))$  space.*

**Proof.** Note that Lemma 20 gives that  $\tilde{K}_1^Y(\text{MON}_4) = \{(\mu_1, \mu') \in [-1, 1]^2 : 2\mu_1 + 3\mu' \geq 1\}$ . We show that  $\tilde{K}_1^Y$  and  $\tilde{K}_{1/2}^N$  are disjoint, and then Lemma 19 and Theorem 11 imply that  $\text{Max-CSP}(\text{MON}_4)$  is sketching approximable in space  $O(\log(n))$ . Next, we prove that no distribution  $\mathcal{D} \in S_{1/2}^N$  has marginals that lie in  $\tilde{K}_1^Y$ .

We start by characterizing  $K_{1/2}^N$  (for general  $\text{MON}_k$ ). Take a distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$ . In order for  $\mathcal{D}$  to lie within  $S_{1/2}^N$ , the following needs to be satisfied:

$$\mathbb{E}_{\mathbf{b} \sim \mathcal{D}_N} \mathbb{E}_{\mathbf{a} \sim \text{Bern}(p)^k} [f(\mathbf{b} \odot \mathbf{a})] \leq \beta, \forall p. \quad (2)$$

Let the function  $h_{\mathcal{D}}(p)$  denote the probability of an assignment from  $\mathcal{D}$  that has undergone bit flips with respect to  $\text{Bern}(p)^k$  to satisfy the monarchy predicate with the probability of  $\beta = 1/2$  or less. With this definition,  $\mathcal{D} \in S_{1/2}^N$  if and only if  $h_{\mathcal{D}}(p) \leq \frac{1}{2}$  for all  $0 \leq p \leq 1$ . Note that negating all variables  $x_i$  flips the output of the monarchy predicate. Therefore, the negation of a “true” assignment is “false” and vice versa. This gives that  $h_{\mathcal{D}}(p) = 1 - h_{\mathcal{D}}(1-p)$  for all  $0 \leq p \leq 1$  which implies that  $\mathcal{D} \in S_{1/2}^N$  if and only if for all  $0 \leq p \leq 1$

$$h_{\mathcal{D}}(p) = \frac{1}{2}.$$

We now write down the coefficients of the polynomial  $h_{\mathcal{D}}(p)$  in terms of  $u_i$  and  $v_i$  describing the distribution (as used earlier in this section).

If one draws an assignment from  $\mathcal{D}$  where  $x_1 = 1$  and exactly  $i$  of the rest of the variables are 1, the probability of the resulting assignment satisfying the monarchy predicate after the Bernoulli flipping is

$$p(1 - (1-p)^i p^{k-1-i}) + (1-p)^{k-i} p^i.$$

Similarly, if  $x_1 = -1$  and exactly  $i$  of the rest of the variables are 1, the probability of the resulting assignment satisfying the monarchy predicate after the Bernoulli flipping is

$$(1-p)(1 - (1-p)^i p^{k-1-i}) + (1-p)^{k-1-i} p^{i+1}.$$

### 35:10 Sketching Approximability of (Weak) Monarchy Predicates

This gives that

$$\begin{aligned} h_{\mathcal{D}}(p) &= \sum_{i=0}^{k-1} u_i [p(1 - (1-p)^i p^{k-1-i}) + (1-p)^{k-i} p^i] \\ &\quad + \sum_{i=0}^{k-1} v_i [(1-p)(1 - (1-p)^i p^{k-1-i}) + (1-p)^{k-1-i} p^{i+1}] \end{aligned} \quad (3)$$

To prove this lemma, we form the polynomial  $h_{\mathcal{D}}(p)$  for  $k = 4$  and show that no set of  $u_i$ s and  $v_i$ s satisfy both  $h_{\mathcal{D}}(p) = \frac{1}{2}$  and  $2\mu_1 + 3\mu' \geq 1$  (where, by (1),  $\mu_1 = \sum_{i=0}^3 (u_i - v_i)$  and  $\mu' = \sum_{i=0}^3 (\frac{2i}{3} - 1)(u_i + v_i)$ .)

$$\begin{aligned} h_{\mathcal{D}}(p) &= u_0 [p(1 - p^3) + (1-p)^4] \\ &\quad + u_1 [p(1 - (1-p)p^2) + (1-p)^3 p] \\ &\quad + u_2 [p(1 - (1-p)^2 p) + (1-p)^2 p^2] \\ &\quad + u_3 [p(1 - (1-p)^3) + (1-p)p^3] \\ &\quad + v_0 [(1-p)(1 - p^3) + (1-p)^3 p] \\ &\quad + v_1 [(1-p)(1 - (1-p)p^2) + (1-p)^2 p^2] \\ &\quad + v_2 [(1-p)(1 - (1-p)^2 p) + (1-p)p^3] \\ &\quad + v_3 [(1-p)(1 - (1-p)^3) + p^4] \\ &= u_0 + v_0 + v_1 + v_2 \\ &\quad + p \cdot (-3u_0 + 2u_1 + u_2 - v_1 - 2v_2 + 3v_3) \\ &\quad + p^2 \cdot (6u_0 - 3u_1 + 3u_3 - 3v_0 + 3v_2 - 6v_3) \\ &\quad + p^3 \cdot (-4u_0 + 2u_1 - 2u_3 + 2v_0 - 2v_2 + 4v_3) \end{aligned}$$

Every distribution (whose marginals are) in  $\tilde{K}_{1/2}^N(\text{MON}_4)$  must satisfy the following system of equations and inequalities, where (4)–(7) are equivalent to  $h_{\mathcal{D}}(p) = \frac{1}{2}$ , and (8)–(10) guarantee that  $u_i$ s and  $v_i$ s describe a distribution.

$$u_0 + v_0 + v_1 + v_2 = \frac{1}{2} \quad (4)$$

$$-3u_0 + 2u_1 + u_2 - v_1 - 2v_2 + 3v_3 = 0 \quad (5)$$

$$6u_0 - 3u_1 + 3u_3 - 3v_0 + 3v_2 - 6v_3 = 0 \quad (6)$$

$$-4u_0 + 2u_1 - 2u_3 + 2v_0 - 2v_2 + 4v_3 = 0 \quad (7)$$

$$\sum_{i=0}^3 (u_i + v_i) = 1 \quad (8)$$

$$u_i \geq 0, \quad \forall 0 \leq i \leq 3 \quad (9)$$

$$v_i \geq 0, \quad \forall 0 \leq i \leq 3 \quad (10)$$

Summing up (5) multiplied by 3, (7) multiplied by  $-13/6$ , and (8) multiplied by  $2/3$ , we have that

$$\begin{aligned} 2/3 &= u_0/3 + 7u_1/3 + 11u_2/3 + 5u_3 - 11v_0/3 - 7v_1/3 - v_2 + v_3 \\ &\geq -u_0 + u_1 + 3u_2 + 5u_3 - 5v_0 - 3v_1 - v_2 + v_3 \\ &= 2\mu_1 + 3\mu', \end{aligned}$$

where the last equality uses (1). By Lemma 20,  $\tilde{K}_1^Y(\text{MON}_4) = \{(\mu_1, \mu') \in [-1, 1]^2 : 2\mu_1 + 3\mu' \geq 1\}$ , and from the above inequality every vector  $(\mu_1, \mu') \in \tilde{K}_{1/2}^N(\text{MON}_4)$  satisfies  $2\mu_1 + 3\mu' \leq 2/3$ . This implies that  $\tilde{K}_1^Y(\text{MON}_4) \cap \tilde{K}_{1/2}^N(\text{MON}_4) = \emptyset$ , and finishes the proof.  $\blacktriangleleft$



### 3.2 Balanced LTFs on 4 variables

In this section, we prove Theorem 4.

► **Theorem 4.** *For every balanced LTF  $f$  on  $k \leq 4$  variables,  $\text{Max-CSP}(f)$  is sketching approximable in  $O(\log(n))$  space.*

We remark that there are non-balanced LTFs on fewer than four variables that are approximation resistant. For example, if  $f(x_1, x_2) = x_1 \text{ OR } x_2$ , then  $\text{Max-CSP}(f)$  is approximation resistant to space  $o(n)$  even in the larger class of streaming algorithms (see, e.g., Corollary 4.2 in [6]).

**Proof of Theorem 4.** After relabeling and negating some of the variables of  $f$ , we can assume that  $f(x_1, x_2, x_3, x_4) = \text{sign}(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4)$ , where  $w_1 \geq w_2 \geq w_3 \geq w_4 \geq 0$  (if  $f$  depends on  $i < 4$  variables, then we set  $w_{i+1} = \dots = w_4 = 0$ ). Since  $f$  is balanced,  $\xi_1w_1 + \xi_2w_2 + \xi_3w_3 + \xi_4w_4 \neq 0$  for all  $\xi_i \in \{-1, 1\}$ . Now consider the following three cases.

- If  $w_1 > w_2 + w_3 + w_4$ , then  $f = \text{sign}(x_1)$  is a dictator function, so  $\text{Max-CSP}(f)$  can be trivially  $(1 - \varepsilon)$ -approximated in  $O(\log(n)/\varepsilon^2)$  space by an  $\ell_1$ -sketch algorithm [11, 12].
- If  $w_2 + w_3 - w_4 < w_1 < w_2 + w_3 + w_4$ , then  $f = \text{MON}_4$  is a monarchy function on  $k = 4$  variables. Indeed, in this case only the sum of the votes of the three last variables overrules the vote of the first variable. By Lemma 21,  $\text{Max-CSP}(f)$  is sketching approximable in  $O(\log(n))$  space.
- If  $w_1 < w_2 + w_3 - w_4$ , then  $f = \text{MAJ}(x_1, x_2, x_3)$  is the majority function on 3 variables. Indeed, the sum of any two weights of the first three variables outweighs the sum of the remaining weights. In this case,  $\text{Max-CSP}(f)$  is known to be sketching approximable in space  $O(\log(n))$  (this follows from the characterization of sketching approximable symmetric functions in [5, Lemma 2.14] and the fact that a balanced LTF doesn't support one-wise independent distributions).

Another way to see that the majority function is sketching approximable is via Theorem 25. Indeed, since majority is a symmetric function, the (non-empty) Chow parameters of the majority function are all equal and non-zero (see, e.g., [17, Theorem 5.19] for the exact values of the Fourier coefficients of the majority function). Then the Chow parameters define the majority function itself, and, by Theorem 25,  $\text{Max-CSP}(f)$  is sketching approximable in space  $O(\log(n))$ . ◀

## 4 Approximation resistance of Monarchy Functions

In this section, we prove Theorem 5: we show that for  $k \geq 5$ , the  $\text{MON}_k$  function is approximation resistant. Recall that by Lemma 19 it suffices to show that  $\tilde{K}_1^Y(\text{MON}_k) \cap \tilde{K}_{1/2}^N(\text{MON}_k) \neq \emptyset$  for  $k \geq 5$ .

In the following we show that for  $k \geq 5$ , there exist vectors  $(\mathbf{u}, \mathbf{v})$  with certain properties that will be useful in showing that  $\tilde{K}_1^Y(\text{MON}_k) \cap \tilde{K}_{1/2}^N(\text{MON}_k) \neq \emptyset$ .

We defer the proof of Lemma 22 to the full version of the paper.

► **Lemma 22.** *For every  $k \geq 5$ , there exists  $\mathbf{u}, \mathbf{v} \in \mathbb{R}_{\geq 0}^k$  satisfying the following conditions.*

- (i)  $\sum_i (u_i + v_i) = 1$ , i.e.,  $\mathbf{u}, \mathbf{v}$  define a distribution  $\mathcal{D}$ . In particular, the marginals of  $\mathcal{D}$  is  $(\mu_1, \mu', \dots, \mu')$  where  $\mu_1 = \sum_i (u_i - v_i)$ , and  $\mu' = \sum_i (\frac{2i}{k-1} - 1)(u_i + v_i)$ .

(ii)  $\mathbf{u}$  and  $\mathbf{v}$  satisfy

$$\begin{aligned}
 & (1/2 - \delta) \sum_{i=0}^{k-1} u_i + (1/2 + \delta) \sum_{i=0}^{k-1} v_i \\
 & + \sum_{i=0}^{k-1} u_i \left( -(1/2 + \delta)^i (1/2 - \delta)^{k-i} + (1/2 - \delta)^i (1/2 + \delta)^{k-i} \right) \\
 & + \sum_{i=0}^{k-1} v_i \left( -(1/2 + \delta)^{i+1} (1/2 - \delta)^{k-1-i} + (1/2 - \delta)^{i+1} (1/2 + \delta)^{k-1-i} \right) \\
 & = 1/2
 \end{aligned}$$

for every  $\delta \in [-1/2, 1/2]$ . In particular, this implies that  $\mathcal{D} \in S_{1/2}^N$ .

(iii)  $p' \geq 1 - \frac{k-2}{k-1} p_1$  where  $p' = \Pr_{\mathbf{x} \sim \mathcal{D}}[x_2 = 1] = \frac{1}{k-1} (\sum_i i u_i + \sum_i i v_i)$  and  $p_1 = \Pr_{\mathbf{x} \sim \mathcal{D}}[x_1 = 1] = \sum_i u_i$ . In particular, this implies the existence of  $\mathcal{D}_Y \in S_1^Y$  and  $\mu(\mathcal{D}_Y) = (\mu_1, \mu', \dots, \mu')$ .

Now, we are ready to prove Theorem 5 using Lemma 22 and Theorem 11.

► **Theorem 5.** For every  $k \geq 5$ ,  $\text{Max-CSP}(\text{MON}_k)$  is sketching approximation resistant to space  $o(\sqrt{n})$ .

**Proof.** For every  $k \geq 5$ , let  $\mathbf{u}, \mathbf{v} \in \mathbb{R}_{\geq 0}^k$ , and  $\mu_1, \mu' \in [-1, 1]$  be the vectors given by Lemma 22. Note that condition (i) guarantees that  $\mathbf{u}, \mathbf{v}$  define a distribution  $\mathcal{D}$  with marginal  $(\mu_1, \mu', \dots, \mu')$ .

First, we show that condition (ii) is a sufficient condition for  $(\mu_1, \mu') \in \tilde{K}_{1/2}^N$ . Recall that  $\mathcal{D}_N \in S_{1/2}^N(\text{MON}_k)$  if for every  $\delta \in [-1/2, 1/2]$ ,  $\mathbb{E}_{\mathbf{b} \in \mathcal{D}_N} \mathbb{E}_{\mathbf{a} \sim \text{Bern}(1/2 + \delta)}[\text{MON}_k(\mathbf{b} \odot \mathbf{a})] = 1/2$ . Since  $\Pr_{\mathbf{x}}[\text{MON}_k(\mathbf{x}) = 1] = \Pr_{\mathbf{x}}[x_1 = 1] - \Pr_{\mathbf{x}}[\mathbf{x} = 10^{k-1}] + \Pr_{\mathbf{x}}[\mathbf{x} = 01^{k-1}]$ , we have that

$$\mathbb{E}_{\mathbf{b} \in \mathcal{D}_N} \mathbb{E}_{\mathbf{a} \sim \text{Bern}(1/2 + \delta)}[\text{MON}_k(\mathbf{b} \odot \mathbf{a})] = \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b}_1 \odot \mathbf{a}_1 = 1] - \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = 1(-1)^{k-1}] + \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = (-1)1^{k-1}].$$

We compute these three probabilities in terms of  $\mathbf{u}, \mathbf{v}, \delta$ .

$$\begin{aligned}
 \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b}_1 \odot \mathbf{a}_1 = 1] &= (1/2 - \delta) \sum_{i=0}^{k-1} u_i + (1/2 + \delta) \sum_{i=0}^{k-1} v_i, \\
 \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = 1(-1)^{k-1}] &= \sum_{i=0}^{k-1} u_i (1/2 + \delta)^i (1/2 - \delta)^{k-i} + \sum_{i=0}^{k-1} v_i (1/2 + \delta)^{i+1} (1/2 - \delta)^{k-1-i}, \\
 \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = (-1)1^{k-1}] &= \sum_{i=0}^{k-1} u_i (1/2 - \delta)^i (1/2 + \delta)^{k-i} + \sum_{i=0}^{k-1} v_i (1/2 - \delta)^{i+1} (1/2 + \delta)^{k-1-i}.
 \end{aligned}$$

Note that condition (ii) implies that

$$\Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b}_1 \odot \mathbf{a}_1 = 1] + \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = 1(-1)^{k-1}] + \Pr_{\mathbf{b}, \mathbf{a}}[\mathbf{b} \odot \mathbf{a} = (-1)1^{k-1}] = \frac{1}{2}$$

for every  $\delta \in [-1/2, 1/2]$  as desired. This implies that  $\mathcal{D} \in S_{1/2}^N(\text{MON}_k)$ . As condition (i) gives  $\mu(\mathcal{D}_N) = (\mu_1, \mu', \dots, \mu')$ , we have  $(\mu_1, \mu') \in \tilde{K}_{1/2}^N$  as desired.

Next, as  $p' = \frac{\mu' + 1}{2}$  and  $1 - \frac{k-2}{k-1} p_1 = 1 - \frac{(k-2)(\mu_1 + 1)}{2(k-1)}$ , condition (iii) implies  $\mu_1(k-2) + \mu'(k-1) \geq 1$ . By Lemma 20, this implies that  $(\mu_1, \mu') \in \tilde{K}_1^Y(\text{MON}_k)$  as desired.

To sum up, Lemma 22 gives us  $(\mu_1, \mu') \in \tilde{K}_1^Y \cap \tilde{K}_{1/2}^N$  for every  $k \geq 5$  and Lemma 19 implies  $(\mu_1, \mu', \dots, \mu') \in K_1^Y \cap K_{1/2}^N$ . By Theorem 11, we conclude that  $\text{MON}_k$  is sketching approximation resistant to space  $o(\sqrt{n})$  and, hence, complete the proof of Theorem 5. ◀

## 5 Chow parameters and the approximability of weak monarchies

In this section, we prove that infinitely many weak monarchy functions are sketching approximable within  $O(\log(n))$  space. We first prove in Section 5.1 that every LTF defined by its Chow parameters (i.e., degree-1 Fourier coefficients as weights and threshold 0) is sketching approximable within  $O(\log(n))$  space. And later in Section 5.2, we prove that infinitely many weak monarchy functions are balanced LTFs defined by their Chow parameters.

### 5.1 Approximability of LTFs defined by their Chow parameters

► **Theorem 23.** *For every Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  of the form*

$$f(x) = \text{sign} \left( \sum_{i=1}^k \widehat{f}(\{i\}) x_i \right),$$

*Max-CSP( $f$ ) is sketching approximable in  $O(\log(n))$  space.*

► **Definition 24.** *Define  $\varepsilon_0(f) = \min\{\sum_{i=1}^k \widehat{f}(\{i\}) \cdot x_i : f(x) = 1\}$ . Define  $\varepsilon^*(f) = \min\{\frac{\varepsilon_0(f)}{3k}, \frac{2\varepsilon_0(f)^2}{9\rho(f)k^2}\}$ .*

We will use the following theorem to prove Theorem 23.

► **Theorem 25.** *For every Boolean function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  and every  $\varepsilon > 0$ , there exists an  $O(\log(n))$  space  $(\rho(f) + \varepsilon^*(f) - \varepsilon)$ -approximation algorithm for Max-CSP( $f$ ).*

First we show how to prove Theorem 23 using Theorem 25.

**Proof of Theorem 23.** If  $f(x)$  is the constant zero function, then it's trivially approximable in  $O(\log(n))$  space. Otherwise, when  $f(x) = \text{sign}(\sum_{i=1}^k \widehat{f}(\{i\}) \cdot x_i)$ , we have  $\varepsilon_0(f) = \min\{\sum_{i=1}^k \widehat{f}(\{i\}) \cdot x_i : f(x) = 1\} > 0$  and hence  $\varepsilon^*(f) > 0$  by their definitions. Now for  $\varepsilon = \varepsilon^*(f)/2$ , Theorem 25 implies that there is a  $(\rho(f) + \varepsilon^*(f)/2)$ -approximation algorithm for Max-CSP( $f$ ), and finishes the proof. ◀

Before we prove Theorem 25, we will describe some useful definitions and lemmas from [5].

Let  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  be a Boolean constraint function of arity  $k$  and  $X_1, \dots, X_n$  be variables. A constraint  $C$  consists of  $\mathbf{j} = (j_1, \dots, j_k) \in [n]^k$  and  $\mathbf{b} = (b_1, \dots, b_k) \in \{-1, 1\}^k$  where the  $j_i$ 's are distinct. The constraint  $C$  reads as requiring  $f(\mathbf{b} \odot \mathbf{X}|_{\mathbf{j}}) = f(b_1 X_{j_1}, \dots, b_k X_{j_k}) = 1$ . A Max-CSP( $f$ ) instance  $\Psi$  contains  $m$  constraints  $C_1, \dots, C_m$  with non-negative weights  $w_1, \dots, w_m$  where  $C_i = (\mathbf{j}(i), \mathbf{b}(i))$  and  $w_i \in \mathbb{R}$  for each  $i \in [m]$ . For an assignment  $\sigma \in \{-1, 1\}^n$ , the value  $\text{val}_{\Psi}(\sigma)$  of  $\sigma$  on  $\Psi$  is the fraction of weight of constraints satisfied by  $\sigma$ , i.e.,  $\text{val}_{\Psi}(\sigma) = \frac{1}{W} \sum_{i \in [m]} w_i \cdot f(\mathbf{b}(i) \odot \sigma|_{\mathbf{j}(i)})$ , where  $W = \sum_{i=1}^m w_i$ . The optimal value of  $\Psi$  is defined as  $\text{val}_{\Psi} = \max_{\sigma \in \{-1, 1\}^n} \text{val}_{\Psi}(\sigma)$ .

► **Definition 26** (Bias (vector)). *For  $\lambda = (\lambda_1, \dots, \lambda_k) \in \mathbb{R}^k$ , and instance  $\Psi = (C_1, \dots, C_m; w_1, \dots, w_m)$  of Max-CSP( $f$ ) where  $C_i = (\mathbf{j}(i), \mathbf{b}(i))$  and  $w_i \geq 0$ , we let the  $\lambda$ -bias vector of  $\Psi$ , denoted  $\text{bias}_{\lambda}(\Psi)$ , be the vector in  $\mathbb{R}^n$  given by*

$$\text{bias}_{\lambda}(\Psi)_{\ell} = \frac{1}{W} \cdot \sum_{i \in [m], t \in [k]: j(i)_t = \ell} \lambda_t w_i \cdot b(i)_t,$$

for  $\ell \in [n]$ , where  $W = \sum_{i \in [m]} w_i$ . The  $\lambda$ -bias of  $\Psi$ , denoted  $B_{\lambda}(\Psi)$ , is the  $\ell_1$  norm of  $\text{bias}_{\lambda}(\Psi)$ , i.e.,  $B_{\lambda}(\Psi) = \sum_{\ell=1}^n |\text{bias}_{\lambda}(\Psi)_{\ell}|$ .

► **Lemma 27** ([5, Lemma 4.7]). For every  $\lambda \in \mathbb{R}^k$ , we have  $B_\lambda(\Psi) = \max_{a \in \{-1,1\}^n} \langle a, \text{bias}_\lambda(\Psi) \rangle$ .

► **Lemma 28** ([5, Lemma 4.4]). For every vector  $\lambda \in \mathbb{R}^k$  and  $\varepsilon > 0$ , there exists a  $O(\log(n))$  space sketching algorithm  $\mathcal{A}$  that on input a stream  $\sigma_1, \dots, \sigma_\ell$ , representing an instance  $\Psi = (C_1, \dots, C_m; w_1, \dots, w_m)$ , outputs a  $(1 \pm \varepsilon)$ -approximation to  $B_\lambda(\Psi)$ , i.e., for every  $\Psi$ ,  $(1 - \varepsilon)B_\lambda(\Psi) \leq \mathcal{A}(\Psi) \leq (1 + \varepsilon)B_\lambda(\Psi)$ , with probability at least  $2/3$ .

Below, we describe Algorithm 1 and show that it is an  $O(\log(n))$  space  $(\rho(f) + \varepsilon^*(f) - \varepsilon)$ -approximation algorithm for  $\text{Max-CSP}(f)$ .

■ **Algorithm 1** A sketching  $(\rho(f) + \varepsilon^*(f) - \varepsilon)$ -approximation algorithm for  $\text{Max-CSP}(f)$ .

---

**Input:** a stream  $\sigma_1, \dots, \sigma_\ell$  representing an instance  $\Psi$  of  $\text{Max-CSP}(f)$  where  $\sigma_i = ((j(i), \mathbf{b}(i)), w_i)$ .

- 1: Let  $\lambda = (\hat{f}(\{1\}), \dots, \hat{f}(\{k\})) \in \mathbb{R}^k$  and  $\varepsilon' = \varepsilon/8$ .
- 2: Use the algorithm  $\mathcal{A}$  from Lemma 28 to compute  $\tilde{B}$  to be a  $(1 \pm \varepsilon')$  approximation to  $B_\lambda(\Psi)$ , i.e.,  $(1 - \varepsilon')B_\lambda(\Psi) \leq \tilde{B} \leq (1 + \varepsilon')B_\lambda(\Psi)$  with probability at least  $2/3$ .
- 3: Let  $\tilde{\delta} = \min\{\frac{1}{3k}, \frac{2\tilde{B}}{9\rho(f)k^2}\}$ .
- 4: **Output:**  $v = \rho(f) + \frac{\tilde{B}\tilde{\delta}}{(1+\varepsilon')^2}$ .

---

It is clear that the algorithm above runs in  $O(\log(n))$  space (in particular by Lemma 28 for Step 2). We now turn to analyzing the correctness of the algorithm.

### 5.1.1 Analysis of the correctness of Algorithm 1

Before we analyse Algorithm 1, we establish some upper and lower bounds on  $\text{val}_\Psi$  in terms of  $B_\lambda(\Psi)$  where  $\lambda = (\hat{f}(\{1\}), \dots, \hat{f}(\{k\}))$ .

► **Lemma 29** (Lower bound on  $\text{val}_\Psi$ ). Let  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  be a Boolean function, and  $\Psi$  be an instance of  $\text{Max-CSP}(f)$ . Then

$$\text{val}_\Psi \geq \rho(f) + B_\lambda(\Psi)\delta(\Psi),$$

where  $\lambda = (\hat{f}(\{1\}), \dots, \hat{f}(\{k\}))$  and  $\delta(\Psi) = \min\{\frac{1}{3k}, \frac{2B_\lambda(\Psi)}{9\rho(f)k^2}\}$ .

► **Lemma 30** (Upper bound on  $\text{val}_\Psi$ ). Let  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  be a Boolean function,  $\varepsilon_0(f)$  be as defined in Definition 24, and  $\Psi$  be an instance of  $\text{Max-CSP}(f)$ . Then

$$\text{val}_\Psi \leq \frac{B_\lambda(\Psi) + \rho(f) \cdot k}{\varepsilon_0(f) + \rho(f) \cdot k},$$

where  $\lambda = (\hat{f}(\{1\}), \dots, \hat{f}(\{k\}))$ .

We defer the proofs of Lemma 29 and Lemma 30 to the full version of the paper. We now show the correctness of Algorithm 1 using these lemmas.

### 5.1.2 Proof of Theorem 25

**Proof of Theorem 25.** First, by Lemma 28, with probability at least  $2/3$ ,  $\tilde{B}$  is a  $(1 \pm \varepsilon')$  approximation to  $B_\lambda(\Psi)$ , i.e.,  $(1 - \varepsilon')B_\lambda(\Psi) \leq \tilde{B} \leq (1 + \varepsilon')B_\lambda(\Psi)$ . Next, we show that with probability at least  $2/3$ , (i)  $v \leq \text{val}_\Psi$  and (ii)  $v \geq (\rho(f) + \varepsilon^*(f) - \varepsilon) \cdot \text{val}_\Psi$ .

(i)  $v \leq \text{val}_\Psi$

We have

$$v = \rho(f) + \frac{\tilde{B}\tilde{\delta}}{(1 + \varepsilon')^2} \leq \rho(f) + B_\lambda(\Psi)\delta(\Psi) \leq \text{val}_\Psi,$$

where the last inequality follows from Lemma 29.

(ii)  $v \geq (\rho(f) + \varepsilon^*(f) - \varepsilon) \cdot \text{val}_\Psi$

We have

$$v = \rho(f) + \frac{\tilde{B}\tilde{\delta}}{(1 + \varepsilon')^2} \geq \rho(f) + B_\lambda(\Psi)\delta(\Psi) \left( \frac{1 - \varepsilon'}{1 + \varepsilon'} \right)^2 \geq \rho(f) + B_\lambda(\Psi)\delta(\Psi)(1 - \varepsilon), \quad (11)$$

where the last inequality follows from the choice of  $\varepsilon'$ . Let us first consider the case when  $B_\lambda(\Psi) \geq \varepsilon_0(f)$ . We have

$$B_\lambda(\Psi)\delta(\Psi) \geq \varepsilon_0(f) \cdot \min \left\{ \frac{1}{3k}, \frac{2\varepsilon_0(f)}{9\rho(f)k^2} \right\} \geq \varepsilon^*, \quad (12)$$

where the last equality follows from the definition of  $\varepsilon^*(f)$  in Definition 24.

Combining Equation (11) and Equation (12), we get

$$v \geq \rho(f) + \varepsilon^*(f)(1 - \varepsilon) \geq (\rho(f) + \varepsilon^*(f) - \varepsilon)\text{val}_\Psi,$$

where the last inequality follows from  $\text{val}_\Psi \leq 1$ .

Now, let us consider the case when  $B_\lambda(\Psi) < \varepsilon_0(f)$ . It follows from Proposition 17 that  $\varepsilon_0(f) \leq \rho(f)k$ . Therefore,

$$\frac{2B_\lambda(\Psi)}{9\rho(f)k^2} \leq \frac{2\varepsilon_0(f)}{9\rho(f)k^2} \leq \frac{2}{9k} < \frac{1}{3k},$$

and so  $\delta(\Psi) = \frac{2B_\lambda(\Psi)}{9\rho(f)k^2}$ . Combining Equation (11) and Lemma 30, we have

$$\frac{v}{\text{val}_\Psi} \geq (1 - \varepsilon) \left( \frac{\rho(f) + \frac{2B_\lambda(\Psi)^2}{9\rho(f)k^2}}{\rho(f) + \frac{B_\lambda(\Psi)}{k}} \right) \left( \rho(f) + \frac{\varepsilon_0(f)}{k} \right).$$

We show that for  $0 \leq B_\lambda(\Psi) \leq \varepsilon_0(f)$ ,

$$\frac{\rho(f) + \frac{2B_\lambda(\Psi)^2}{9\rho(f)k^2}}{\rho(f) + \frac{B_\lambda(\Psi)}{k}} \geq \frac{\rho(f) + \frac{2\varepsilon_0(f)^2}{9\rho(f)k^2}}{\rho(f) + \frac{\varepsilon_0(f)}{k}}. \quad (13)$$

This immediately implies that

$$\frac{v}{\text{val}_\Psi} \geq (1 - \varepsilon) \left( \rho(f) + \frac{2\varepsilon_0(f)^2}{9\rho(f)k^2} \right) \geq (1 - \varepsilon)(\rho(f) + \varepsilon^*(f)) > \rho(f) + \varepsilon^*(f) - \varepsilon.$$

Consider the function  $g(p) = \frac{\rho(f) + \frac{2p^2}{9\rho(f)}}{\rho(f) + p}$ . In order to show Equation (13), it suffices to show that in the range  $p \in [0, \frac{\varepsilon_0(f)}{k}]$ ,  $g(p)$  attains the minimum value at  $p = \frac{\varepsilon_0(f)}{k}$ , i.e.,  $g'(p) < 0$  in

this range. We have  $g'(p) = \frac{\left( \frac{2(p + \rho(f))^2}{9\rho(f)} - \frac{11\rho(f)}{9} \right)}{(\rho(f) + p)^2}$  and for  $p \in [0, \frac{\varepsilon_0(f)}{k}]$ , we have

$$\left( \frac{2(p + \rho(f))^2}{9\rho(f)} - \frac{11\rho(f)}{9} \right) \leq \left( \frac{2(\varepsilon_0(f)/k + \rho(f))^2}{9\rho(f)} - \frac{11\rho(f)}{9} \right) \leq \frac{8\rho(f)}{9} - \frac{11\rho(f)}{9} = -\frac{\rho(f)}{3} < 0.$$

This completes the proof of Theorem 25.  $\blacktriangleleft$

## 5.2 Approximability of weak monarchy functions

In this section, we analyze the streaming approximability of  $\text{Max-CSP}(f)$  where  $f$  is a weak monarchy function. Note that in order for  $\text{WMON}_{k,j}$  to be a balanced LTF, the total number of votes, i.e.,  $j + k - 1$ , needs to be odd. Therefore, we make such assumption throughout the rest of this section. We defer the proof of Lemma 31 to the full version of the paper.

► **Lemma 31.** *For all integers  $j \geq 2$  and  $k \geq 7j^3$  such that  $k + j$  is even,*

$$\text{WMON}_{k,j}(x) = \text{sign} \left( \sum_{i=1}^k \widehat{\text{WMON}_{k,j}}(\{i\})x_i \right).$$

Note that Lemma 31 along with Theorem 23 directly conclude Theorem 6 restated below.

► **Theorem 6.** *For all integers  $j \geq 2$  and  $k \geq 7j^3$  such that  $k + j$  is even,  $\text{Max-CSP}(\text{WMON}_{k,j})$  is sketching approximable in  $O(\log(n))$  space. In particular, for every  $j$ , there exist infinitely many  $k$  such that  $\text{Max-CSP}(\text{WMON}_{k,j})$  is sketching approximable.*

---

## References



- 1 Per Austrin, Siavosh Benabbas, and Avner Magen. On quadratic threshold csps. In *LATIN 2010*, pages 332–343. Springer, 2010.
- 2 Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. Closed-form expressions for the sketching approximability of (some) symmetric Boolean CSPs. *CoRR*, abs/2112.06319, February 2022.
- 3 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs. In *STOC 2022*, 2022. To appear.
- 4 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. In *FOCS 2021*, pages 1197–1208. IEEE, 2021. doi:10.1109/FOCS52979.2021.00117.
- 5 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs with linear sketches. *CoRR*, abs/2102.12351v8, 11th february 2022.
- 6 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max- $k$ SAT. In *FOCS 2020*, pages 330–341. IEEE, 2020. doi:10.1109/FOCS46700.2020.00039.
- 7 Venkatesan Guruswami and Runzhou Tao. Streaming Hardness of Unique Games. In *APPROX 2019*, pages 5:1–5:12. Schloss Dagstuhl, 2019.
- 8 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In *APPROX 2017*, pages 8:1–8:19. Schloss Dagstuhl, 2017.
- 9 Gustav Hast. *Beating a random assignment: Approximating constraint satisfaction problems*. PhD thesis, KTH, 2005.
- 10 Neng Huang and Aaron Potechin. On the approximability of presidential type predicates. In *APPROX 2020*, pages 58:1–58:20. Schloss Dagstuhl, 2020.
- 11 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *FOCS 2000*, pages 189–197. IEEE, 2000.
- 12 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA 2010*, pages 1161–1178. SIAM, 2010.
- 13 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *SODA 2015*, pages 1263–1282. SIAM, 2015.

- 14 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker.  $(1 + \omega(1))$ -approximation to MAX-CUT requires linear space. In *SODA 2017*, pages 1703–1722. SIAM, 2017.
- 15 Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In *STOC 2019*, pages 277–288. ACM, 2019. doi:10.1145/3313276.3316364.
- 16 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *ITCS 2015*, pages 367–376. ACM, 2015.
- 17 Ryan O’Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.
- 18 Aaron Potechin. On the approximation resistance of balanced linear threshold functions. In *STOC 2019*, pages 430–441. ACM, 2019.
- 19 Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP. In *APPROX 2021*, pages 17:1–17:19. Schloss Dagstuhl, 2021.





# Integrality Gap of Time-Indexed Linear Programming Relaxation for Coflow Scheduling

Takuro Fukunaga  

Faculty of Science and Engineering, Chuo University, Tokyo, Japan

---

## Abstract

Coflow is a set of related parallel data flows in a network. The goal of the coflow scheduling is to process all the demands of the given coflows while minimizing the weighted completion time. It is known that the coflow scheduling problem admits several polynomial-time 5-approximation algorithms that compute solutions by rounding linear programming (LP) relaxations of the problem. In this paper, we investigate the time-indexed LP relaxation for coflow scheduling. We show that the integrality gap of the time-indexed LP relaxation is at most 4. We also show that yet another polynomial-time 5-approximation algorithm can be obtained by rounding the solutions to the time-indexed LP relaxation.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** coflow scheduling, hypergraph matching, approximation algorithm

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.36

**Category** APPROX

**Funding** Takuro Fukunaga: JSPS KAKENHI Grant Numbers JP20H05965, JP21K11759, and JP21H03397, Japan.

## 1 Introduction

*Coflow scheduling* was introduced by Chowdhury and Stoica [7]. It is motivated by cluster computation frameworks such as MapReduce and Hadoop. Because these frameworks involve a huge amount of communication within a computer cluster, it is crucial to efficiently schedule this communication to achieve high computation performance. Coflow is an abstraction of data flow created by the processing of a task within the computer cluster. The goal of coflow scheduling is to find the most efficient scheduling of coflows.

Among the many variations of the coflow scheduling problem, weighted completion minimization under a bipartite matching model is the most extensively studied setting. In this setting, a coflow is represented as a bipartite undirected multigraph. An edge in the coflow represents the demand of sending one unit of data from one node to another. We are given a set of coflows  $F_1, \dots, F_k$ , all of which are on the same bipartition  $(X, Y)$  of the node set. Each coflow  $F_i$  is associated with a weight  $w_i \geq 0$  and a release time  $r_i \in \mathbb{Z}_+$ , where  $\mathbb{Z}_+$  is the set of non-negative integers. The required task is to schedule all demands of the coflows under the congestion constraint and the release time constraint. The congestion constraint requires all nodes to send or receive at most one unit of data at any moment, and the release time constraint requires the demand of coflow  $F_i$  to not be processed before release time  $r_i$ . The completion time  $C_i$  of coflow  $F_i$  is defined as the time at which all demands of  $F_i$  have been processed. The objective of the problem is to minimize the weighted completion time, defined as  $\sum_{i=1}^k w_i C_i$ . More information on the problem setting is given in Section 2.

This coflow scheduling problem includes the *concurrent open shop scheduling problem*, which corresponds to the special case where  $X = \{x_1, \dots, x_n\}$ ,  $Y = \{y_1, \dots, y_n\}$  and each edge of the given coflows joins nodes  $x_i$  and  $y_i$  for some  $i \in \{1, \dots, n\}$ . For concurrent open shop scheduling, achieving  $(2 - \epsilon)$ -approximation for any  $\epsilon > 0$  is known to be NP-hard [16].



© Takuro Fukunaga;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 36; pp. 36:1–36:13



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Thus, the same approximation hardness holds for coflow scheduling. The best approximation factor for coflow scheduling is achieved by the algorithms proposed by Shafiee and Ghaderi [17] and Ahmadi et al. [2], respectively. The factor is 4 when the release times for all given coflows are identical and 5 when they are not identical. Narrowing the gap between the upper and lower bounds of the approximation factor is an interesting open problem.

The above approximation algorithms [2, 17] for coflow scheduling are both based on linear programming (LP) relaxations of the problem. The algorithm of Shafiee and Ghaderi [17] uses a relaxation with ordering variables and that of Ahmadi et al. [2] uses a relaxation with parallel inequalities. These relaxations are commonly used in the machine scheduling literature. Their algorithms also give upper bounds on the integrality gap of these LP relaxations.

### 1.1 Our contribution

Our contribution is to investigate the *time-indexed LP relaxation*, which is another standard formulation of LP relaxations for machine scheduling problems. We show that the integrality gap of the time-indexed LP relaxation is at most 4 even for non-identical release times, which is better than the known upper bounds on the integrality gap of other LP relaxations. Our integrality gap analysis relies on Hall's theorem [1] on existence of perfect matchings in bipartite hypergraphs. We show that a 4-approximate solution is obtained by finding a perfect matching in a bipartite hypergraph constructed from an optimal solution to the time-indexed LP relaxation.

Unfortunately, our integrality gap analysis does not provide a polynomial-time algorithm of approximation factor that matches the integrality gap bound because there are no known polynomial-time algorithms for computing hypergraph perfect matchings implied by Hall's theorem. Nevertheless, we believe that our analysis is useful for obtaining an improved polynomial-time approximation algorithm in the future.

We would also like to point out that our analysis is a new interesting application of the Hall's theorem on hypergraphs. Previously Hall's theorem on hypergraphs has been used for developing approximation algorithms for min-max allocation problems in a series of studies (see e.g., [4, 5]). We note that this line of studies was initiated by Asadpour, Feige, and Saberi [5], the algorithm given in which is not a polynomial-time algorithm.

In addition to the integrality gap bound, we give a polynomial-time rounding algorithm for the time-indexed LP relaxation. Although our algorithm does not improve upon the currently best approximation algorithms [2, 17], we prove that our algorithm achieves the same approximation factors as them. Namely, its approximation factor is 4 for the identical release times, and 5 for non-identical release times.

Inspired by our polynomial-time rounding algorithm, we also observe that, if a hypergraph is constructed from the coflow scheduling with identical release times, then a perfect matching can be found in polynomial-time time. This gives an alternative 4-approximation algorithm for the coflow scheduling with identical release times.

Summing up, our contributions can be summarized as follows.

- We show that the rounding of a solution to the time-indexed LP relaxation can be reduced to finding a perfect matching in a hypergraph. This implies that the integrality gap of the time-indexed LP relaxation is at most 4, which improves on the integrality gap upper bounds on LP relaxations for non-identical release times.
- We propose a polynomial-time rounding algorithm for the time-indexed LP relaxation. Its approximation factor is 4 for identical release times and 5 for non-identical release times. These factors match those of the currently known best approximation algorithms for coflow scheduling.

- We propose a polynomial-time algorithm for computing perfect matchings in hypergraphs constructed in the reduction of rounding solutions to the time-indexed LP relaxation with identical release times.

## 1.2 Organization

The rest of this paper is organized as follows. Section 2 introduces preliminary facts and related studies on coflow scheduling and hypergraph perfect matching. Section 3 formulates the time-indexed LP relaxation. Section 4 presents the analysis of the integrality gap of the time-indexed LP relaxation. Section 5 describes the proposed polynomial-time rounding algorithm for the time-indexed LP relaxation. Section 6 describes the proposed polynomial-time algorithm for computing perfect matchings in hypergraphs constructed from the coflow scheduling with identical release times. Section 7 concludes this work.

## 2 Preliminary facts and related studies

### 2.1 Coflow scheduling

Throughout this paper, an edge between two nodes  $x$  and  $y$  is denoted by  $xy$ . The set of integers  $1, \dots, n$  is denoted by  $[n]$ . For an edge set  $I$  and a node  $v$ , the set of edges in  $I$  incident to  $v$  is denoted by  $\delta_I(v)$ . The subscript is omitted when the edge set is clear from the context. The maximum degree of a graph  $G$  is denoted by  $\Delta(G)$ .

As mentioned in Section 1, the inputs of the bipartite matching model of the coflow scheduling problem are coflows  $F_1, \dots, F_k$  with weights  $w_1, \dots, w_k \geq 0$  and release times  $r_1, \dots, r_k \in \mathbb{Z}_+$ , where coflows are bipartite multigraphs on the bipartition  $(X, Y)$  of the node set. We usually identify a graph with the set of edges. Let  $F$  denote  $\bigcup_{i=1}^k F_i$ .

We denote the time horizon of schedules by  $T$ . In this paper, we consider finding a *discrete-time integer* schedule, for which the time interval  $[0, T)$  is divided into intervals  $[0, 1), [1, 2), \dots, [T-1, T)$  and the data flow does not vary within an interval. We refer to interval  $[t-1, t)$  as the  $t$ -th round. In contrast to a discrete-time schedule, a *continuous-time* schedule can change the data flow at any moment. In an integer schedule, data flow forms a matching in each round by the congestion constraint. Thus, a schedule is equivalent to a sequence  $(M_1, \dots, M_T)$  of matchings such that  $\bigcup_{t=1}^T M_t = F$  and  $M_t \cap F_i = \emptyset$  for each  $i \in [k]$  and  $t \in [r_i]$ . The completion time  $C_i$  of coflow  $F_i$  in the schedule is given by  $\max\{t: M_t \cap F_i \neq \emptyset\}$ . The objective of the problem is to minimize the weighted completion time  $\sum_{i=1}^k w_i C_i$ . In addition to integer schedules, we can also consider a *fractional* schedule, where data flow within a round forms a fractional matching, i.e., a vector  $x \in [0, 1]^E$  such that  $\sum_{e \in \delta(v)} x(e) \leq 1$  for each  $v \in X \cup Y$ .

Since its introduction by Chowdhury and Stoica [7], coflow scheduling has been extensively studied from both practical and theoretical viewpoints [2, 8, 9, 14, 17, 18]. Several extensions of the problem setting have been presented. For example, Im et al. [13] considered the matroid coflow scheduling problem, which replaces the congestion constraint with a constraint that requires the set of elements scheduled in a round to be independent in a given matroid. Note that the bipartite matching model cannot be modeled by the matroid coflow, and hence the result of Im et al. cannot be applied to the bipartite matching model. Chowdhury et al. [6] considered flows in general graphs instead of bipartite matchings in the congestion constraint. Their model is a generalization of the bipartite matching model. However, the algorithm of Chowdhury et al. outputs only a fractional schedule, and thus it cannot be used for computing an integer schedule.

## 2.2 Hypergraph perfect matching

Let  $H = (V, E)$  be a hypergraph with node set  $V$  and hyperedge set  $E$ . Here, we regard each hyperedge as a set of nodes.

A *matching*  $M$  in a hypergraph  $H = (V, E)$  is a subset of  $E$  such that  $|\delta_M(v)| \leq 1$  for all  $v \in V$ , where we naturally extend the notation  $\delta$  to hypergraphs. A *transversal*  $U$  of  $H = (V, E)$  is a subset of  $V$  such that  $U \cap e \neq \emptyset$  for all  $e \in E$ . The maximum size of matchings and the minimum size of transversals of  $H$  are called the *matching number* and the *transversal number* of  $H$ , denoted by  $\nu(H)$  and  $\tau(H)$ , respectively. A *fractional matching* is a function  $x: E \rightarrow [0, 1]$  such that  $\sum_{e \in \delta(v)} x(e) \leq 1$  for each  $v \in V$ . The maximum value of  $\sum_{e \in E} x(e)$  among all fractional matchings  $x$  in  $H$  is called the *fractional matching number* of  $H$  and is denoted by  $\nu^*(H)$ . Note that  $\nu(H) \leq \nu^*(H) \leq \tau(H)$  holds for any hypergraph  $H$ .

$H$  is said to be *r-uniform* if  $|e| = r$  for each  $e \in E$ , and is said to be *bipartite* if its node set has a bipartition  $(A, B)$  such that  $|A \cap e| = 1$  for all  $e \in E$ . Hereafter, we suppose that  $H$  is an *r-uniform bipartite hypergraph* with bipartition  $(A, B)$ . We denote the nodes in  $A$  by *A-nodes* and those in  $B$  by *B-nodes*. A *perfect matching* in  $H$  is a matching whose size is  $|A|$  (i.e., all *A-nodes* are covered by some hyperedge in the matching). For  $X \subseteq A$ , let  $H_X$  represent the hypergraph with the node set  $B$  and the hyperedge set  $E_X := \{e \setminus A: e \in E, e \cap X \neq \emptyset\}$ . The following sufficient conditions for the existence of perfect matching are known.

► **Theorem 1** (Haxell [11]). *If an  $r$ -uniform bipartite hypergraph  $H$  with the node set bipartition  $(A, B)$  satisfies*

$$\tau(H_X) > (2r - 3)(|X| - 1) \text{ for any } X \subseteq A, \quad (1)$$

*then  $H$  has a perfect matching.*

► **Theorem 2** (Aharoni and Haxell [1]). *If an  $r$ -uniform bipartite hypergraph  $H$  with the node set bipartition  $(A, B)$  satisfies*

$$\nu(H_X) > (r - 1)(|X| - 1) \text{ for any } X \subseteq A, \quad (2)$$

*then  $H$  has a perfect matching.*

Note that these two theorems extend the sufficient condition implied by Hall's theorem to the existence of perfect matchings in bipartite graphs (although the condition in Hall's theorem is necessary and sufficient, the conditions in the above two theorems are not).

The proofs of these theorems are not algorithmic. Nevertheless, Annamalai [3] gave an algorithmic proof of Haxell's theorem by introducing a small amount of slack into the condition. More concretely, Annamalai showed that, if there exists a constant  $\epsilon > 0$  such that the hypergraph  $H$  satisfies  $\tau(H_X) > (2r - 3 + \epsilon)(|X| - 1)$  for any  $X \subseteq A$ , then there exists a polynomial-time algorithm for finding a perfect matching in  $H$ . There is no known polynomial-time algorithm for finding a perfect matching in a hypergraph that satisfies condition (2). Note that finding perfect matchings in 3-uniform bipartite hypergraphs is NP-hard in general because it includes 3-dimensional matching [15].

## 3 Time-indexed LP relaxation

In this section, we introduce the time-indexed LP relaxation for the coflow scheduling problem.

We set  $T$  to an upper bound on the time horizon of optimal coflow scheduling. For example,  $T$  can be set to  $|F|$ . Indeed, we can see that  $2\Delta(F) + \max_{i \in [k]} r_i$  is also an upper bound because of the observations explained below in Lemma 6.

In the time-indexed LP, we have a variable  $x_{t,e} \in [0, 1]$  for each  $t \in [T]$  and  $e \in F$ , and a variable  $c_i$  for each  $i \in [k]$ . When the variables take integer values, variable  $x_{t,e}$  indicates whether the demand  $e$  is processed in the  $t$ -th round (i.e., time interval  $[t-1, t)$ ), and variable  $c_i$  is the completion time of coflow  $F_i$ .

The time-indexed LP is formulated as follows.

$$\text{minimize} \quad \sum_{i \in [k]} w_i c_i$$

$$\text{subject to} \quad \sum_{t \in [T]} t x_{t,e} \leq c_i, \quad \forall i \in [k], \forall e \in F_i, \quad (3)$$

$$\sum_{e \in \delta_F^-(v)} x_{t,e} \leq 1, \quad \forall t \in [T], \forall v \in V, \quad (4)$$

$$\sum_{t \in [T]} x_{t,e} = 1, \quad \forall i \in [k], \forall e \in F_i, \quad (5)$$

$$\begin{aligned} x_{t,e} &= 0, & \forall i \in [k], \forall e \in F_i, \forall t \in [r_i], \\ x_{t,e} &\geq 0, & \forall e \in F, \forall t \in [T]. \end{aligned} \quad (6)$$

Constraint (3) requires  $c_i$  to be at least the time of processing  $e \in F_i$ . Constraint (4) requires at most one edge incident to a node  $v$  to be processed within the  $t$ -th round. Constraint (5) requires each demand  $e$  in coflow  $F_i$  to be processed in some round. Constraint (6) requires the demands in coflow  $F_i$  to not be processed before the release time  $r_i$ .

Each solution for the time-indexed LP relaxation represents a discrete-time fractional schedule that consists of fractional matchings  $x_1, \dots, x_T \in [0, 1]^F$ . Let  $C_i$  be the completion time of coflow  $F_i$  in this schedule, expressed as

$$C_i = \max\{t \in [T] : x_{t,e} > 0 \text{ for some } e \in F_i\}.$$

Thus, the weighted completion time of this fractional schedule is  $\sum_{i \in [k]} w_i C_i$ . Note that this value is possibly larger than the objective value  $\sum_{i \in [k]} w_i c_i$  of the relaxation.

In the bipartite matching model, the discrete-time fractional schedule can be transformed into a continuous-time integer schedule without increasing the completion time of each coflow as follows. By the integrality of the fractional matching polytope, the fractional matching  $x_t$  can be represented as a convex combination of (integer) matchings. Namely, there exists a set of matchings  $M_1, \dots, M_m$  and nonnegative numbers  $\lambda_1, \dots, \lambda_m$  such that  $x_t = \sum_{j=1}^m \lambda_j \chi_{M_j}$  and  $\sum_{j=1}^m \lambda_j = 1$  hold, where  $\chi_{M_j}$  is the characteristic vector of matching  $M_j$ . A continuous-time integer schedule is obtained by scheduling the matching  $M_j$  for time  $\lambda_j$  within the  $t$ -th round.

Conversely, a continuous-time integer schedule can be transformed into a discrete-time fractional schedule. Let  $\lambda_M$  be the time spent for processing a matching  $M$  in the  $t$ -th round of the integer schedule. Then, the convex combination of matchings with coefficients  $\lambda_M$  is a fractional matching. A discrete-time fractional schedule is obtained by scheduling this fractional matching in the  $t$ -th round. If the completion time of coflow  $F_i$  in the integer schedule is  $C'_i$ , the completion time of  $F_i$  in the constructed fractional schedule is  $\lceil C'_i \rceil$ .

► **Remark.** The size of the time-indexed LP linearly depends on  $T$ , and hence running time for solving the LP is at least a polynomial with regards to  $T$ . Although this running time is polynomial in the input size of the instance of the coflow scheduling problem, it may be a disadvantage compared with other LP relaxations such as those used in [2, 17]. However, the size of the time-indexed LP can be reduced using a commonly used technique (see e.g., [12]) so that it depends on  $O(\log T)$  with a loss of  $1 + \epsilon$  in the approximation factor for any constant  $\epsilon > 0$ .

## 4 Integrality gap analysis

This section proves that the integrality gap of the time-indexed LP relaxation is at most 4 for the bipartite matching model. In the proof, we first show that there exists a discrete-time fractional schedule whose weighted completion time is at most twice the optimal objective value of the relaxation. Then, this fractional schedule is rounded into an integer schedule that is subject to the completion time of each coflow being at most twice that in the fractional schedule. This rounding is done by finding a perfect matching in a hypergraph constructed from the fractional schedule.

### 4.1 Random stretching of fractional schedule

As mentioned in Section 3, a solution  $(x, c)$  for the time-indexed LP relaxation represents a discrete-time fractional schedule, but the completion time  $C_i$  of coflow  $F_i$  in this schedule is possibly larger than  $c_i$ . However, as studied in [6, 13], random stretching gives another fractional schedule wherein the expected completion time of  $F_i$  is at most  $2c_i$ . The details are as follows.

For  $e \in F$  and  $t \in [T]$ , let  $v_e(t) = \sum_{t' \in [t]} x_{t', e}$ . Furthermore, we extend the definition of  $v_e(t)$  to any  $t \in [0, T]$  via linear interpolation. Namely, if  $t \in [t' - 1, t')$  for some  $t' \in [T]$ , then  $v_e(t) := v_e(t' - 1) + (t - t' + 1)(v_e(t') - v_e(t' - 1))$ .

For  $i \in [k]$  and  $\theta \in [0, 1]$ , we define  $C_i(\theta)$  as the time at which  $\theta$ -fraction of coflow  $F_i$  is completed in the discrete-time fractional schedule implied by the solution  $(x, c)$  to the relaxation. That is,  $C_i(\theta)$  is the minimum value of  $t \in [0, T]$  such that  $v_e(t) \geq \theta$  for all  $e \in F_i$ .

In the random stretching operation, we randomly sample  $\theta$  from  $[0, 1]$  according to the probability density function  $f(\theta) := 2\theta$ . Then, we stretch the schedule by the factor  $1/\theta$ . This means that if a demand is processed in a time interval  $[t', t'']$ , then it is processed in  $[t'/\theta, t''/\theta]$ . The processing of a demand is truncated when the processing time reaches one unit of time. This gives a continuous-time fractional schedule such that the completion time of a coflow  $F_i$  is  $C_i(\theta)/\theta$ .

The continuous-time fractional schedule can be transformed into a discrete-time fractional schedule as follows. For  $t \in [T]$  and  $e \in F$ , let  $\bar{x}_{t,e}$  be the fraction of  $e$  processed in time  $[t-1, t)$  of the continuous-time fractional schedule. Then, it can be verified that  $\{\bar{x}_{t,e} : e \in F\}$  forms a fractional matching for any  $t \in [T]$ , and thus it gives a discrete-time fractional schedule. In this discrete-time schedule, the process of coflow  $F_i$  is within an interval  $[r_i, \lceil C_i(\theta)/\theta \rceil]$ .

We have thus obtained a discrete-time fractional schedule by stretching the schedule represented by the LP optimal solution. The following lemma shows that the expected completion time in this schedule can be bounded by twice the objective value of the time-indexed LP.

► **Lemma 3.** *For each  $i \in [k]$ ,  $\mathbb{E}[\lceil C_i(\theta)/\theta \rceil] \leq 2c_i$ .*

This lemma is proven in [6, 13] for other variations of the coflow scheduling problem, and these proofs also apply to our problem. We omit the proof of Lemma 3 in this paper.

In the rest of the paper, we let  $\bar{C}_i$  denote  $\lceil C_i(\theta)/\theta \rceil$ .



## 4.2 Reduction to hypergraph perfect matching

By Lemma 3, a schedule of processing coflow  $F_i$  within the interval  $[r_i, \bar{C}_i]$  achieves a weighted completion time that is at most twice the optimal objective value of the relaxation. Moreover, the discrete-time fractional schedule implied by  $\bar{x}$  does so. What remains is to round this fractional schedule into a discrete-time integer schedule.

For the matroid coflow scheduling problem, Im et al. [13] showed that this rounding process can be done without loss of the approximation factor. This is because the fractional schedule is included in the intersection of a matroid polytope and a base polytope, where the matroid polytope is defined based on a constraint that requires demands processed in each round to be independent in the given matroid. Because the intersection forms an integer polytope, the fractional schedule can be represented as a convex combination of integer schedules, any of which processes coflow  $F_i$  within  $[r_i, \bar{C}_i]$ . This approach is not available for our problem because bipartite matchings do not form a matroid but a matroid intersection; thus the set of the fractional schedules is the intersection of two matroid polytopes and a base polytope, that is not integer in general.

Instead, we reduce the rounding process to hypergraph perfect matching. We first construct a hypergraph as follows. We prepare  $T$  copies of the node set, each of which corresponds to a round. We let  $V_t$  denote the copy corresponding to the  $t$ -th round for each  $t \in [T]$ , and let  $v_t$  denote the node in  $V_t$  corresponding to  $v \in X \cup Y$ . In addition, we introduce a node  $a_e$  corresponding to each demand  $e \in F$ . Let  $A := \{a_e : e \in F\}$  and  $B := \bigcup_{t \in [T]} V_t$ . A hyperedge in the hypergraph is defined by an edge  $e = xy \in F_i$  and time  $t \in [r_i + 1, \bar{C}_i]$  as  $h_{e,t} := \{x_t, y_t, a_e\}$ . Let  $H = (V_H, E_H)$  denote the hypergraph with the node set  $V_H = A \cup B$  and the hyperedge set  $E_H = \{h_{e,t} : i \in [k], e \in F_i, t \in [r_i + 1, \bar{C}_i]\}$ . Note that  $H$  is a 3-uniform bipartite hypergraph with bipartition  $(A, B)$ .

From a perfect matching in  $H$ , we define a discrete-time integer schedule so that a demand  $e = uv$  is processed in the  $t$ -th round whenever the hyperedge  $h_{e,t}$  is included in the matching. Because each node in  $B$  is incident to at most one hyperedge in a matching, the demands processed in each round of the schedule form a matching. Moreover, because each node  $a_e \in A$  is covered by exactly one hyperedge in the perfect matching, and because all hyperedges incident to  $a_e$  are defined only for the  $t$ -th rounds with  $t \in [r_i + 1, \bar{C}_i]$  if  $e \in F_i$ , the demand  $e \in F_i$  is processed within an interval  $[r_i, \bar{C}_i]$  in the schedule. Therefore, the defined integer schedule is feasible.

Based on this discussion, it suffices to find a perfect matching in  $H$ . However, we do not know whether  $H$  has a perfect matching. To ensure the existence of a perfect matching, we modify  $H$  so as to satisfy the Aharoni-Haxell condition (2). For this purpose, let us bound  $\nu(H_X)$  for  $X \subseteq A$ . First, observe that  $H_X$  is a bipartite graph, with the node set  $B = \bigcup_{t \in [T]} V_t$  and the edge set  $\{u_t v_t : a_{uv} \in X, t \in [r_i + 1, \bar{C}_i] \text{ for } i \text{ with } uv \in F_i\}$ . Therefore,  $\tau(H_X) = \nu^*(H_X) = \nu(H_X)$ . Moreover,  $\bar{x}_{t,uv}$  can be regarded as a weight assigned to edge  $u_t v_t$  in  $H_X$ . It forms a fractional matching in  $H_X$ . Because  $\sum_{t \in [T]} \bar{x}_{t,uv} = 1$ ,  $H_X$  has a fractional matching of size  $|X|$ . These facts indicate that  $\nu(H_X) \geq |X|$ .

This bound is insufficient to satisfy the Aharoni-Haxell condition, which requires satisfying  $\nu(H_X) > 2(|X| - 1)$  since  $r = 3$  in our case. Thus, we modify  $H$  as follows. In the original definition, for each round  $t \in [T]$ , we have the corresponding node set  $V_t$ , and the node set of  $H$  is defined as  $A \cup (\bigcup_{t \in [T]} V_t)$ . For each  $i \in [k]$ ,  $e = uv \in F_i$ , and  $t \in [r_i + 1, \bar{C}_i]$ ,  $H$  has a hyperedge  $\{a_e, u_t, v_t\}$ . In the new definition, for each round  $t \in [T]$ , we define two node sets  $V_{2t-1}$  and  $V_{2t}$ , and define the node set as  $A \cup (\bigcup_{t \in [T]} V_{2t-1} \cup V_{2t})$ . Hyperedges  $\{a_e, u_{2t-1}, v_{2t-1}\}$  and  $\{a_e, u_{2t}, v_{2t}\}$  are defined for each  $i \in [k]$ ,  $e = uv \in F_i$ , and  $t \in [r_i + 1, \bar{C}_i]$ . Let  $H'$  denote the obtained hypergraph.

► **Lemma 4.**  *$H'$  has a perfect matching.*

**Proof.**  $H'$  is still a 3-uniform bipartite hypergraph, with the bipartition  $(A, \bigcup_{t=1}^{2T} V_t)$ . Let us show that  $H'_X$  satisfies  $\nu(H'_X) \geq 2|X|$  for any  $X \in A$ , which indicates the existence of a perfect matching in  $H'$  by Lemma 2.

Note that each edge  $u_tv_t$  in  $H'_X$  is defined by a hyperedge  $\{a_e, u_t, v_t\}$  incident to an  $A$ -node  $a_e \in X$ . We define  $x'_{u_tv_t}$  as  $\bar{x}_{\lceil t/2 \rceil, e}$  for each edge  $u_tv_t$  in  $H'_X$ . Then,  $x'$  is a fractional matching in  $H'_X$  because  $\bar{x}_t$  is a fractional matching for each  $t \in [T]$ . Moreover, because  $\sum_{t \in [T]} \bar{x}_{t,e} = 1$ ,  $\sum_{t \in [2T]} x'_{u_tv_t} = 2$  holds. Thus, the size of the fractional matching  $x'$  is  $2|X|$ , and hence  $\nu^*(H'_X) \geq 2|X|$ . Note that  $H'_X$  is a bipartite graph, and hence  $\nu(H'_X) = \nu^*(H'_X)$ . Therefore, the claim is proven. ◀

We can define a discrete-time integer schedule from a perfect matching in  $H'$ ; if  $a_e$  is covered by a hyperedge  $\{a_e, u_t, v_t\}$  in the perfect matching, then demand  $e$  is processed in the  $t$ -th round. Because each  $A$ -node  $a_e$  has incident hyperedges corresponding to rounds in  $[2(r_i + 1) - 1, 2\bar{C}_i]$  if  $e \in F_i$ , the constructed integer schedule satisfies the release time constraint and all demands of coflow  $F_i$  are completed by time  $2\bar{C}_i$ . Therefore, the weighted completion time of this schedule is at most  $2 \sum_{i \in F} w_i \bar{C}_i$ . This fact and Lemma 3 prove the following theorem.

► **Theorem 5.** *The integrality gap of the time-indexed LP relaxation is at most 4.*

As for a lower bound on the integrality gap of the time-indexed LP, the following simple instance shows that it is at least 2. Suppose that there is a single coflow that consists of  $M$  parallel edges, and its weight and release time are 1 and 0. The minimum weighted completion time of integer schedules for this instance is  $M$ . On the other hand, the fractional schedule that processes  $1/M$  unit of all edges in each round achieves the weighted completion time  $(M + 1)/2$ . The ratio of this value to  $M$  approaches 2 as  $M$  grows. We are aware of no instance that indicates integrality gap larger than 2.

As mentioned in Section 2.2, the Aharoni-Haxell condition ensures the existence of a perfect matching but does not provide a polynomial-time algorithm for finding it. The algorithm of Annamalai [3] finds a perfect matching in a hypergraph that satisfies the Haxell condition with a constant slack, i.e.,  $\tau(H'_X) > (2r - 3 + \epsilon)(|X| - 1)$  for any  $X \subseteq A$  and any constant  $\epsilon > 0$  (again, recall that  $r = 3$  in our case). Using this algorithm gives us a polynomial-time rounding algorithm, but making the hypergraph satisfy the condition results in an approximation factor of 6, which is worse than that for existing coflow scheduling algorithms.

## 5 Polynomial-time rounding algorithm

In this section, we present a polynomial-time rounding algorithm for the time-indexed LP. It achieves 4-approximation for identical release times and 5-approximation for non-identical release times.

The algorithm first sorts the coflows in the non-decreasing order of  $c$ . Then, it schedules the demands greedily, giving higher priority to demands of earlier coflows. The details of this algorithm are given in Algorithm 1.

► **Lemma 6.** *The completion time of coflow  $F_i$  in the schedule output by Algorithm 1 is at most  $r_i + 2\Delta(\bigcup_{j=1}^i F_j) - 1$  for each  $i \in [k]$ .*

---

**Algorithm 1** Rounding Algorithm.

---

```

1 solve the time-indexed LP to obtain an optimal solution (x, c) ;
2 sort the coflows so that $c_1 \leq c_2 \leq \dots \leq c_k$;
3 $M_t := \emptyset$ for each $t \in [T]$;
4 for $i = 1, \dots, k$ do
5 for $uv \in F_i$ do
6 find the minimum $t \in [r_i + 1, T]$ such that $\delta_{M_t}(u) = \delta_{M_t}(v) = \emptyset$;
7 $M_t := M_t \cup \{uv\}$
8 output (M_1, \dots, M_T)

```

---

**Proof.** Let  $uv$  be a demand in  $F_i$  that is processed last, and let  $t$  be the round in which  $uv$  is processed (i.e.,  $t$  is the completion time of  $F_i$ ). Then, in each round in  $[r_i + 1, \dots, t - 1]$ , a demand incident to  $u$  or  $v$  is processed. This means that  $t - 1 - r_i \leq |\delta_{\bigcup_{j=1}^i F_j}(u)| - 1 + |\delta_{\bigcup_{j=1}^i F_j}(v)| - 1 \leq 2\Delta(\bigcup_{j=1}^i F_j) - 2$  holds. Therefore, the completion time of  $F_i$  is at most  $r_i + 2\Delta(\bigcup_{j=1}^i F_j) - 1$ .  $\blacktriangleleft$

Now, we prove the following.

► **Lemma 7.** For each  $i \in [k]$ ,  $\Delta(\bigcup_{j=1}^i F_j) \leq 2c_i$ .

**Proof.** Suppose that the indices of coflows indicate those after sorting in line 3 of the algorithm. Namely,  $c_1 \leq c_2 \leq \dots \leq c_k$ . We fix  $i \in [k]$  and  $v \in X \cup Y$ , and we prove that the degree of  $v$  in the graph  $\bigcup_{j=1}^i F_j$  is at most  $2c_i$ .

Since  $\sum_{t \in [T]} x_{t,e} = 1$  holds for any  $e$  by (5), we have

$$\sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{t \in [T]} x_{t,e} = \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} 1 = \sum_{j \in [i]} |\delta_{F_j}(v)|.$$

It suffices to show that this value is at most  $2c_i$ . For arriving at a contradiction, suppose that this is more than  $2c_i$ , i.e.,

$$2c_i < \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{t \in [T]} x_{t,e}. \quad (7)$$

Let  $e \in F_j$  for some  $j \leq i$ . Then, (3) and the assumption of  $c_j \leq c_i$  show that

$$\sum_{t \in [T]} tx_{t,e} \leq c_j \leq c_i. \quad (8)$$

Moreover, since  $\sum_{t \in [T]} x_{t,e} = 1$  holds by (5), we have

$$\begin{aligned} c_i - \sum_{t \in [T]} tx_{t,e} &= \sum_{t \in [T]} c_i x_{t,e} - \sum_{t \in [T]} tx_{t,e} \\ &= \sum_{t \in [T]} (c_i - t)x_{t,e} \\ &= \sum_{1 \leq t \leq c_i} (c_i - t)x_{t,e} + \sum_{c_i < t \leq T} (c_i - t)x_{t,e}. \end{aligned}$$

### 36:10 Integrality Gap of Time-Indexed LP for Coflow Scheduling

Since (8) indicates that this is at least 0, we have

$$\sum_{c_i < t \leq T} (t - c_i)x_{t,e} \leq \sum_{1 \leq t \leq c_i} (c_i - t)x_{t,e}.$$

Summing this inequality over all  $j \in [i]$  and  $e \in \delta_{F_j}(v)$  gives

$$\sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{c_i < t \leq T} (t - c_i)x_{t,e} \leq \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{1 \leq t \leq c_i} (c_i - t)x_{t,e}. \quad (9)$$

Since  $\sum_{e \in \delta_F(v)} x_{t,e} \leq 1$  for each  $t \in [T]$  by (4), the right-hand side of (9) is bounded as

$$\sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{1 \leq t \leq c_i} (c_i - t)x_{t,e} \leq \sum_{1 \leq t \leq c_i} (c_i - t) \sum_{e \in \delta_F(v)} x_{t,e} \leq \sum_{1 \leq t \leq c_i} (c_i - t) = \frac{c_i(c_i - 1)}{2}. \quad (10)$$

On the other hand, from (7), we have

$$\sum_{c_i < t \leq T} \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} x_{e,t} > 2c_i - \sum_{1 \leq t \leq c_i} \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} x_{e,t} \geq c_i.$$

Thus the left-hand side of (9) is bounded as

$$\sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{c_i < t \leq T} (t - c_i)x_{t,e} = \sum_{c_i < t \leq T} \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} (t - c_i)x_{t,e} > \sum_{c_i < t \leq 2c_i} (t - c_i) = \frac{c_i(c_i + 1)}{2}. \quad (11)$$

(9), (10), and (11) give a contradiction.  $\blacktriangleleft$

Combining Lemmas 6 and 7 proves the following theorem.

► **Theorem 8.** *Algorithm 1 is a 4-approximation algorithm for identical release times and a 5-approximation algorithm for non-identical release times.*

**Proof.** By Lemmas 6 and 7, the schedule output by Algorithm 1 processes the coflow  $F_i$  by time  $r_i + 4c_i - 1$ . Note that  $r_i \leq c_i$  holds for each  $i \in [k]$ . Therefore, the weighted completion time of the schedule is at most  $5 \sum_{i \in [k]} w_i c_i$ , which means that the algorithm achieves 5-approximation. In the identical release time case, we can assume that  $r_i = 0$  for all  $i \in [k]$ . Then, the weighted completion time of the schedule is at most  $4 \sum_{i \in [k]} w_i c_i$ , which means that it achieves 4-approximation.  $\blacktriangleleft$

► **Remark.** The above analysis does not depend on the assumption that coflows  $F_1, \dots, F_k$  are bipartite. Thus, it applies to the general graph model, where given coflows are not bipartite graphs and the congestion constraint requires that the demands processed in each round form a (non-bipartite) matching. Although this is not mentioned in previous works, similar analysis shows that the approximation algorithms of [2, 17] can also work for the general graph model. In other words, these approximation algorithms do not make full use of the assumption that the coflows are bipartite. In contrast, the integrality gap analysis given in Section 4 uses the bipartiteness.

## 6 Finding perfect matchings in hypergraphs

We proved Theorem 5 by showing that the hypergraph  $H'$  (defined in Section 4.2) has a perfect matching. Unfortunately, we do not know how to find the perfect matching in polynomial time even though its existence is implied by Theorem 2. In this section, we present a polynomial-time algorithm for finding a perfect matching in  $H'$  when  $r_i = 0$  for all  $i \in [k]$ . This gives an alternative proof of the statement for identical release times in Theorem 8.

---

### Algorithm 2 Perfect Matching Algorithm.

---

```

1 sort the coflows so that $\bar{C}_1 \leq \bar{C}_2 \leq \dots \leq \bar{C}_k$;
2 $M := \emptyset$;
3 for $i = 1, \dots, k$ do
4 for $uv \in F_i$ do
5 find the minimum $t \in [2\bar{C}_i]$ such that both u_t and v_t have no incident
 hyperedge in M ;
6 add hyperedge $\{a_{uv}, u_t, v_t\}$ to M
7 output M

```

---

The algorithm is given in Algorithm 2. The next theorem shows that it finds a perfect matching.

► **Theorem 9.** *Algorithm 2 outputs a perfect matching in polynomial time.*

**Proof.** On line 5 of Algorithm 2, there always exists  $t \in [2\bar{C}_i]$  such that both  $u_t$  and  $v_t$  have no incident hyperedge in  $M$ . If this claim is true,  $M$  is a perfect matching in  $H'$  at the termination of the algorithm. Because the algorithm runs in polynomial time, this proves the theorem.

To prove the above claim, we first show that  $\sum_{j \in [i]} |\delta_{F_j}(v)| \leq \bar{C}_i$  holds for each  $i \in [k]$  and  $v \in V$ . Recall that there exists  $\bar{x}_{t,e} \in [0, 1]$  ( $e \in F_j$ ,  $t \in \bar{C}_j$ ) such that  $\sum_{t \in [\bar{C}_j]} \bar{x}_{t,e} = 1$  for each  $e \in F_j$ , and  $\sum_{j \in [k]} \sum_{e \in \delta_{F_j}(v)} \bar{x}_{t,e} \leq 1$  for each  $t \in [T]$  and  $v \in V$ . Then,

$$\begin{aligned}
 \sum_{j \in [i]} |\delta_{F_j}(v)| &= \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} 1 = \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \sum_{t \in [\bar{C}_j]} \bar{x}_{t,e} \\
 &= \sum_{t \in [\bar{C}_i]} \sum_{j \in [i]} \sum_{e \in \delta_{F_j}(v)} \bar{x}_{t,e} \leq \sum_{t \in [\bar{C}_i]} 1 = |\bar{C}_i|.
 \end{aligned}$$

Here, the third equality uses the fact that  $\bar{C}_j \leq \bar{C}_i$  for all  $j \in [i]$ .

Then, when  $uv \in F_i$  is chosen on line 4 of Algorithm 2, the number of hyperedges in  $M$  incident to nodes  $u_1, \dots, u_{2\bar{C}_i}$  is at most  $\sum_{j \in [i]} |\delta_{F_j}(u)| - 1 \leq \bar{C}_i - 1$ . Similarly, the number of hyperedges in  $M$  incident to nodes  $v_1, \dots, v_{2\bar{C}_i}$  is at most  $\sum_{j \in [i]} |\delta_{F_j}(v)| - 1 \leq \bar{C}_i - 1$ . Therefore, among  $2\bar{C}_i$  pairs of  $\{u_t, v_t\}$  ( $t \in [2\bar{C}_i]$ ), there exist at least  $2\bar{C}_i - 2(\bar{C}_i - 1) = 2$  pairs such that no hyperedge in  $M$  is incident to nodes in the pairs. ◀

## 7 Conclusion

We showed that the integrality gap of the time-indexed LP relaxation for the coflow scheduling problem is at most 4. We also proposed a polynomial-time rounding algorithm that achieves 4-approximation for identical release times and 5-approximation for non-identical release

times. In addition, we proposed a polynomial-time algorithm for finding a perfect matching in the bipartite hypergraph constructed from a solution for the time-indexed LP relaxation with identical release times.

There are many interesting directions of further study. One of them is to improve the approximation factor, in particular for non-identical release times. Based on our integrality gap analysis, this can be achieved by developing a polynomial-time algorithm for finding perfect matchings in 3-uniform bipartite hypergraphs that satisfy the Aharoni-Haxell condition (2). However, designing such an algorithm is regarded as a difficult problem. Indeed, it is mentioned in [10] as “Thus algorithmic versions of these results would also be very interesting and useful, but currently seem out of reach.” We believe that it is interesting to investigate algorithms for hypergraphs constructed in our rounding of solutions to the time-indexed LP relaxation with non-identical release times.

---

## References

---

- 1 Ron Aharoni and Penny Haxell. Hall’s theorem for hypergraphs. *Journal of Graph Theory*, 35(2):83–88, 2000.
- 2 Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. On scheduling coflows. *Algorithmica*, 82(12):3604–3629, 2020.
- 3 Chidambaram Annamalai. Finding perfect matchings in bipartite hypergraphs. *Combinatorica*, 38(6):1285–1307, 2018.
- 4 Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. *ACM Transactions on Algorithms*, 13(3):37:1–37:28, 2017.
- 5 Arash Asadpour, Uriel Feige, and Amin Saberi. Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012.
- 6 Mosharaf Chowdhury, Samir Khuller, Manish Purohit, Sheng Yang, and Jie You. Near optimal coflow scheduling in networks. In Christian Scheideler and Petra Berenbrink, editors, *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22–24, 2019*, pages 123–134. ACM, 2019.
- 7 Mosharaf Chowdhury and Ion Stoica. Coflow: a networking abstraction for cluster applications. In Srikanth Kandula, Jitendra Padhye, Emin Gün Sirer, and Ramesh Govindan, editors, *11th ACM Workshop on Hot Topics in Networks, HotNets-XI, Redmond, WA, USA – October 29 – 30, 2012*, pages 31–36. ACM, 2012.
- 8 Mosharaf Chowdhury and Ion Stoica. Efficient coflow scheduling without prior knowledge. In Steve Uhlig, Olaf Maennel, Brad Karp, and Jitendra Padhye, editors, *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17–21, 2015*, pages 393–406. ACM, 2015.
- 9 Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient coflow scheduling with Varys. In Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy, editors, *ACM SIGCOMM 2014 Conference, SIGCOMM’14, Chicago, IL, USA, August 17–22, 2014*, pages 443–454. ACM, 2014.
- 10 Alessandra Graf and Penny Haxell. Finding independent transversals efficiently. *Combinatorics, Probability & Computing*, 29(5):780–806, 2020.
- 11 Penny E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.
- 12 Sungjin Im and Shi Li. Better unrelated machine scheduling for weighted completion time via random offsets from non-uniform distributions. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9–11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 138–147. IEEE Computer Society, 2016.

- 13 Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Manish Purohit. Matroid coflow scheduling. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPICs*, pages 145:1–145:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 14 Hamidreza Jahanjou, Erez Kantor, and Rajmohan Rajaraman. Asymptotically optimal approximation algorithms for coflow scheduling. In Christian Scheideler and Mohammad Taghi Hajiaghayi, editors, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 45–54. ACM, 2017.
- 15 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.
- 16 Sushant Sachdeva and Rishi Saket. Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 219–229. IEEE Computer Society, 2013.
- 17 Mehrnoosh Shafiee and Javad Ghaderi. An improved bound for minimizing the total weighted completion time of coflows in datacenters. *IEEE/ACM Transactions on Networking*, 26(4):1674–1687, 2018.
- 18 Yue Zeng, Baoliu Ye, Bin Tang, Songtao Guo, and Zhihao Qu. Scheduling coflows of multi-stage jobs under network resource constraints. *Computer Networks*, 184:107686, 2021.





# Fair Correlation Clustering in General Graphs

Roy Schwartz ✉

The Henry and Marilyn Taub Faculty of Computer Science, Technion, Haifa, Israel

Roded Zats ✉

The Henry and Marilyn Taub Faculty of Computer Science, Technion, Haifa, Israel

---

## Abstract

We consider the family of Correlation Clustering optimization problems under fairness constraints. In Correlation Clustering we are given a graph whose every edge is labeled either with a  $+$  or a  $-$ , and the goal is to find a clustering that agrees the most with the labels:  $+$  edges within clusters and  $-$  edges across clusters. The notion of fairness implies that there is no over, or under, representation of vertices in the clustering: every vertex has a color and the distribution of colors within each cluster is required to be the same as the distribution of colors in the input graph. Previously, approximation algorithms were known only for fair disagreement minimization in complete unweighted graphs. We prove the following: (1) there is no finite approximation for fair disagreement minimization in general graphs unless  $P = NP$  (this hardness holds also for bicriteria algorithms); and (2) fair agreement maximization in general graphs admits a bicriteria approximation of  $\approx 0.591$  (an improved  $\approx 0.609$  true approximation is given for the special case of two uniformly distributed colors). Our algorithm is based on proving that the sticky Brownian motion rounding of [Abbasi Zadeh-Bansal-Guruganesh-Nikolov-Schwartz-Singh SODA'20] copes well with uncut edges.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering

**Keywords and phrases** Correlation Clustering, Approximation Algorithms, Semi-Definite Programming

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.37

**Category** APPROX

**Funding** This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 852870-ERC-SUBMODULAR.

## 1 Introduction

**Correlation-Clustering** is a family of clustering optimization problems in which the goal is to cluster objects given pairwise similarity/dissimilarity information over the objects. In **Correlation-Clustering** we are given a graph  $G = (V, E)$  equipped with edge weights  $w : E \rightarrow \mathbb{R}_+$ , whose vertices are the objects, and each edge  $e = (u, v) \in E$  is labeled either with a  $+$  or a  $-$ . A  $+$  indicates similarity of  $u$  and  $v$  and a  $-$  indicates dissimilarity of  $u$  and  $v$ . We denote by  $E^+$  the collection of edges labeled with a  $+$  and by  $E^-$  the collection of edges labeled with a  $-$ . The goal is to find a clustering  $\mathcal{C}$ , i.e.,  $\mathcal{C} = \{C_1, \dots, C_l\}$  is a partition of  $V$  with no restriction on  $l$ , that agrees as much as possible with the labeling of the edges. A  $+$  edge is in agreement if its endpoints are in the same cluster and a  $-$  edge is in agreement if its endpoints are in different clusters. Two natural objectives have attracted much attention since the introduction of **Correlation-Clustering** close to two decades ago by Bansal, Blum and Chawla [12]. The first, denoted as **Max-Agreement**, is to maximize the total weight of edges that are in agreement:

$$\max_{\mathcal{C}} \left\{ \sum_{e=(u,v) \in E^+ : \mathcal{C}(u)=\mathcal{C}(v)} w(e) + \sum_{e=(u,v) \in E^- : \mathcal{C}(u) \neq \mathcal{C}(v)} w(e) \right\},$$



© Roy Schwartz and Roded Zats;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 37; pp. 37:1–37:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

where  $\mathcal{C}(u)$  denotes the cluster in  $\mathcal{C}$  that vertex  $u$  belongs to. The second, denoted as **Min-Disagreement**, is to minimize the total weight of edges that are in disagreement:

$$\min_{\mathcal{C}} \left\{ \sum_{e=(u,v) \in E^+ : \mathcal{C}(u) \neq \mathcal{C}(v)} w(e) + \sum_{e=(u,v) \in E^- : \mathcal{C}(u) = \mathcal{C}(v)} w(e) \right\}.$$

**Correlation-Clustering** has attracted much attention [7, 9, 19, 23, 39, 5, 24, 25, 6, 18, 28], both from the theoretical and practical perspectives. From a theoretical perspective **Correlation-Clustering** captures some fundamental graph cut problems such as **Multicut** and **Multiway-Cut**. From a practical perspective, it has found numerous practical applications in a wide range of settings, e.g., image segmentation [41], cross lingual link detection [40], coreference resolution [35], to name a few (refer to the survey of Wirth [41] for additional details).

Chierichetti, Kumar, Lattanzi and Vassilvitskii [21] introduced the notion of fairness in clustering where they considered the  $k$ -**Center** and  $k$ -**Median** problems. Informally, in fair clustering problems, each vertex has a type and each cluster needs to contain not too many and not too few vertices from each type. In general, fairness in clustering has received much attention in recent years that goes beyond  $k$ -**Center** and  $k$ -**Median**, e.g., [2, 11, 14, 33, 38, 15, 1, 4] (refer to surveys [16, 20] for additional details). One of the main reasons for considering fairness in algorithms in general, and clustering in particular, arises from human-centric applications. The goal is to ensure that the solutions are not biased with respect to a sensitive feature such as gender or race. For example, clustering and learning algorithms used for college admissions, bank loans, job applications etc. might be biased [16, 20]. Thus, there is a lot of effort to develop fair clustering algorithms as seen in the literature referenced above.

In this work we consider fairness in **Correlation-Clustering**. Formally, each vertex  $v$  is associated with one of  $k$  given colors  $\{1, \dots, k\}$  and we denote  $v$ 's color by  $c(v)$ . Additionally, we denote by  $V_i$  all vertices of color  $i$ , i.e.,  $V_i \triangleq \{u : c(u) = i\}$  for every  $i = 1, \dots, k$ . We are also given the ratios of these colors in  $V$ , i.e., there exists  $h \in \mathbb{N}$  such that  $V$  contains  $h \cdot p_i$  vertices of the  $i^{\text{th}}$  color (where  $p_1, \dots, p_k \in \mathbb{N}^{\geq 1}$ ).<sup>1</sup> We denote these ratios by  $p_1 : \dots : p_k$ .<sup>2</sup> The fairness constraint on the clustering  $\mathcal{C}$  is that for every cluster in  $\mathcal{C}$  and for every two colors  $i$  and  $j$  the ratio of the number of vertices in the cluster of color  $i$  to the number of vertices in the cluster of color  $j$  equals  $p_i/p_j$ . Hence, every cluster in  $\mathcal{C}$  preserves the color ratios of the vertices in the input graph  $G$ . We denote the problem of **Min-Disagreement** with a fairness constraint as **Fair-Min-Disagreement** and the problem of **Max-Agreement** with a fairness constraint as **Fair-Max-Agreement**. Typically, all the above mentioned applications of fairness in clustering satisfy that  $\sum_{i=1}^k p_i = o(n)$ .

**Fair-Min-Disagreement** in complete unweighted graphs was considered by Ahmadi, Galhotra, Saha and Schwartz [1] and by Ahmadian, Epasto, Kumar and Mahdian [4]. For two colors and a ratio of  $1 : 1$  [1] present an approximation of  $(3\alpha + 4)$  where  $\alpha$  is the best known approximation for **Min-Disagreement** in complete unweighted graphs ( $\alpha = 2.06$  [19]). For two colors and a ratio of  $1 : p$  [4, 1] present an approximation of  $O(p^2)$ . For a general number of colors  $k$  and ratios  $1 : p_2 : \dots : p_k$  an approximation of  $k^2 \cdot \max_{i=2, \dots, k} \{p_i^2\}$  was also given by [4, 1], as well as relaxed bi-criteria guarantees. All the above results reduce the problem to **Min-Disagreement** (without any fairness requirements) by matching nodes of different colors and merging them.

<sup>1</sup> It is assumed without loss of generality that there does not exist a number  $s > 1$  such that  $p_i/s \in \mathbb{N}$  for all  $i = 1, \dots, k$ .

<sup>2</sup> Note that  $p_i/p_j = |V_i|/|V_j|$  for every  $i, j = 1, \dots, k$ .

To the best of our knowledge, no approximation algorithms are known for general instances of **Fair-Min-Disagreement** as the above results of [1, 4] apply only to complete unweighted graphs. Additionally, to the best of our knowledge, no approximation algorithms are known for **Fair-Max-Agreement**.

## 1.1 Our Results and Techniques

We show that **Fair-Min-Disagreement** is hard to approximate within any finite approximation factor. Moreover, we prove that this hardness holds even for the special case of only two colors and a ratio of 1 : 1. This is summarized in the following theorem.

► **Theorem 1.** *If **Fair-Min-Disagreement** with 2 colors and a ratio of 1 : 1 admits a polynomial time approximation algorithm with a finite approximation guarantee, then  $P = NP$ .*

This hardness result is extended to bi-criteria algorithms, and it holds even for the special case of only three colors and ratios of 1 : 1 : 1. We say that an algorithm for **Fair-Min-Disagreement** is a bi-criteria  $(\alpha, 1 + \varepsilon)$ -approximation if it outputs a clustering  $\mathcal{C} = \{C_1, \dots, C_l\}$  that satisfies: (1) the cost of  $\mathcal{C}$  is at most  $\alpha$  times the cost of an optimal solution; and (2) for each  $1 \leq r \leq l$  it holds that  $|C_r \cap V_i|/|C_r \cap V_j| \leq (1 + \varepsilon)p_i/p_j$  for every (ordered) pair of colors  $i$  and  $j$ . This is summarized in the following theorem.

► **Theorem 2.** *For every  $\alpha \geq 1$  and  $\varepsilon > 0$ , if **Fair-Min-Disagreement** with 3 colors and ratios of 1 : 1 : 1 admits a bi-criteria  $(\alpha, 1 + \varepsilon)$  polynomial time approximation algorithm, then  $P = NP$ .*

Let us focus now on **Fair-Max-Agreement**. Obtaining an approximation of (roughly)  $1/2$  is easy (see Section 2), and thus the challenge is improving it. In order to achieve such an improvement, we first notice that one can restrict attention to solutions that contain only two clusters. We prove that if one returns the best of: (1) an  $\alpha$ -approximate fair two-cluster solution; and (2) a suitably chosen solution that is comprised of the smallest possible fair clusters (note that each such cluster contains exactly  $p_i$  vertices of color  $i$ ), then we can obtain an approximation better than  $1/2$  for **Fair-Max-Agreement** assuming  $\alpha$  is sufficiently large. The resulting approximation for **Fair-Max-Agreement** depends on  $\alpha$ , thus the bulk of the effort is focused on obtaining a good approximation for the problem where the output is restricted to having only two clusters.

First, we consider a case study with two colors and a ratio of 1 : 1. For this case study problem, we prove that one can reduce the two-cluster problem to a cut maximization problem that captures both **Max-Bisection** and **Max- $\frac{n}{2}$ -Uncut** (see Section 1.3 for the exact definitions) with *no* fairness constraints. Thus, we use machinery developed for **Max-Bisection** and **Max- $\frac{n}{2}$ -Uncut**, that is based on rounding a Lassere SDP hierarchy relaxation (see [37, 10, 42]), to obtain the following theorem.

► **Theorem 3.** ***Fair-Max-Agreement** with two colors and a ratio of 1 : 1 admits a polynomial time 0.609-approximation algorithm.*

When considering general instances, it is not clear if (or how) one can reduce the two-cluster problem to a problem that has no fairness constraints. Hence, a different approach is needed. We adopt the sticky Brownian motion approach of Abbasi-Zadeh, Bansal, Guruganesh, Nikolov, Schwartz and Singh [43], which was successfully used for approximating **Max-Cut** with side constraints (see Section 2.2 for the definition). In order to apply this approach to

our problem, we prove that it can simultaneously handle both edges that cross between the two clusters and edges that do not cross between the two clusters (it is important to note that the work of [43] deals only with edges that cross the cut when considering **Max-Cut** with side constraints). However, this comes at a price of a slightly worse bi-criteria approximation when compared to the case study which has two colors and a ratio of 1 : 1.

We say that an algorithm for **Fair-Max-Agreement** is a bi-criteria  $(\alpha, \varepsilon)$ -approximation if it outputs a clustering  $\mathcal{C} = \{C_1, \dots, C_l\}$  that satisfies: (1) the value of  $\mathcal{C}$  is at least  $\alpha$  times the value of an optimal clustering; and (2) for every  $1 \leq j \leq l$  there exists a  $h_j \in \{1, \dots, n / \sum_{i=1}^k p_i\}$  such that  $||C_j \cap V_i| - h_j \cdot p_i| \leq \varepsilon n$  for all  $1 \leq i \leq k$ .

► **Theorem 4.** *Fair-Max-Agreement with  $k \geq 2$  colors and ratios of  $p_1 : \dots : p_k$  admits for every  $0 < \varepsilon < 1 - \sum_{i=1}^k p_i/n$  a bicriteria  $((0.591 - \varepsilon)(1 - \sum_{i=1}^k p_i/n), \varepsilon)$ -approximation whose running time is  $O(n^{\text{poly}(\log(k)/\varepsilon)})$ .*

Recalling that  $\sum_{i=1}^k p_i = o(n)$  is the typical case, the approximation in the above theorem is in fact  $0.591 - \varepsilon - o(1)$ . Moreover, if  $k = O(1)$  then the running time of the algorithm is polynomial.

## 1.2 Related Work

**Correlation-Clustering** has received a lot of attention since its introduction by Bansal, Blum, and Chawla [12] close to two decades ago. The best known approximation algorithm **Min-Disagreement** in general graphs obtains a  $O(\log n)$  approximation [24, 18]. For **Max-Agreement** the best known approximations are obtained by rounding the natural SDP relaxation and achieve a guarantee of 0.7666 [39] and 0.7664 [18]. For complete unweighted graphs **Max-Agreement** admits a PTAS [12] while **Min-Disagreement** has a long sequence of works [12, 19, 18, 6] where the current best known achieves an approximation of 2.06 [19].

Fairness in clustering has attracted much attention since the work of Chierichetti, Kumar, Lattanzi and Vassilvitskii [21] for  $k$ -**Center** and  $k$ -**Median**. It was followed by works on the same two problems [14, 15, 11], as well as  $k$ -**Means** [38, 27]. Moreover, [33] considered fairness in the context of spectral clustering. Related notions of fairness were also studied [3, 14]. Fairness in **Correlation-Clustering** was considered by [1, 4] and also extended to hierarchical clustering [2].

In this work we use Lasserre SDP hierarchy to formulate relaxations. The Lasserre hierarchy [34] has been used to develop approximation algorithms for numerous combinatorial optimization problems. Here we mention only few of the related works that directly relate to our problem. Focusing on **Max-Bisection**, Raghavendra and Tan [37] obtained a 0.85 approximation ratio using the Lasserre SDP hierarchy. Following their work, Austrin, Benabbas and Georgiou [10] improved this ratio to 0.8776 which almost matches the Goemans-Williamson approximation ratio for **Max-Cut** [29]. Wu, Du and Xu [42] considered other graph bisection maximization problems and generalized the algorithm of [10] and showed that **Max- $\frac{n}{2}$ -Uncut** admits an approximation ratio of 0.8776.

## 1.3 Preliminaries

We denote a cut as  $\mathcal{S} = \{S, V \setminus S\}$  where  $\delta(\mathcal{S}) = \{(u, v) \in E \mid (u \in S \wedge v \notin S) \vee (u \notin S \wedge v \in S)\}$  is the collection of edges crossing  $S$ , and  $E(\mathcal{S}) = \{(u, v) \in E \mid u, v \in S\}$  is the collection of edges that have both endpoints in  $S$ . For every  $X \subseteq E$  we denote  $w(X) = \sum_{e \in X} w(e)$  the sum of weights of edges in  $X$ . We similarly use the notations above for  $E^+$  and  $E^-$ , i.e.,  $w^-(X) = w(X \cap E^-)$  and  $w^+(X) = w(X \cap E^+)$ . For the **Max-Agreement** objective and a

clustering  $\mathcal{C}$  we denote the weight of edges in agreement in  $\mathcal{C}$  as  $v(\mathcal{C})$  (alternatively,  $v(\mathcal{C})$  is the value of the clustering  $\mathcal{C}$ ). Additionally, we denote an optimal clustering by  $\mathcal{C}^*$  and its value by  $OPT = v(\mathcal{C}^*)$ . Moreover, we denote by  $v^+(\mathcal{C})$  and  $v^-(\mathcal{C})$  the contribution of the  $+$  and  $-$  edges to  $v(\mathcal{C})$ , respectively. Let us now define the variant of the problem where the number of clusters is bounded.

► **Definition 5.** *Fair-Max-Agreement* where the numbers of clusters in a solution is required to be at most  $r$  is denoted as *Fair-Max-Agreement* [ $r$ ].

Note that *Fair-Max-Agreement* [ $n$ ] is essentially *Fair-Max-Agreement* with no restriction on the number of clusters in the output. *Fair-Max-Agreement* [2] is related to *Max-Bisection* and *Max- $\frac{n}{2}$ -Uncut* problems which are defined as follows. Given a graph  $G = (V, E)$  the goal is to find a cut  $S \subseteq V$  where  $|S| = n/2$  such that  $w(\delta(S))$  is maximized for *Max-Bisection* and  $w(E(S)) + w(E(V \setminus S))$  is maximized for *Max- $\frac{n}{2}$ -Uncut*.

In this work we use Lasserre SDP hierarchy relaxations, which contain vectors  $\mathbf{v}_S$  for subsets  $S \subseteq V$ . We use the following abbreviated notations:  $\mathbf{v}_i = \mathbf{v}_{\{i\}}$  for the singleton set  $\{i\}$ ,  $\mathbf{v}_\emptyset = \mathbf{v}_\emptyset$  for the empty set,  $\mu_i = \mathbf{v}_i \cdot \mathbf{v}_\emptyset$  denotes the “marginal probability” of vertex  $i$ , and  $\rho_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j$  is the covariance between vertices  $i$  and  $j$ . Additionally,  $\tilde{\mathbf{w}}_i = \mathbf{v}_i - \mu_i \mathbf{v}_\emptyset$  is the component of  $\mathbf{v}_i$  in the linear subspace that is orthogonal to  $\mathbf{v}_\emptyset$ , and  $\mathbf{w}_i = \tilde{\mathbf{w}}_i / \|\tilde{\mathbf{w}}_i\|_2$  is its normalized vector.

## 2 Algorithms for Fair-Max-Agreement

We split this section into two parts. In the first part we consider the case study with two colors and a uniform ratio of  $1 : 1$ . In the second part we consider general instances with  $k \geq 2$  colors and ratios  $p_1 : \dots : p_k$ .

### 2.1 Case Study – Two Colors with Ratio 1:1

#### 2.1.1 A Simple $(1/2)$ -Approximation

► **Observation 6.** *Let  $G = (V_1 \cup V_2, E^+ \cup E^-)$  be an instance with two colors and a ratio of  $1 : 1$ . Let  $f : V_1 \rightarrow V_2$  be a bijection such that  $M^- \triangleq \{(u, f(u)) : u \in V_1\}$  minimizes  $w(M^- \cap E^-)$ . Then every clustering  $\mathcal{C}$  satisfies  $v^-(\mathcal{C}) \leq w(E^-) - w(M^- \cap E^-)$ .<sup>3</sup>*

The proof of the following theorem, which is based on Observation 6, appears in Appendix A.

► **Theorem 7.** *There is a polynomial time  $(1/2)$ -approximation algorithm for *Fair-Max-Agreement* in general weighted graphs with two colors and a ratio of  $1 : 1$ .*

#### 2.1.2 Beating the $(1/2)$ -Approximation Ratio

The following lemma shows that there is a solution with only two clusters whose value is sufficiently large (a similar idea was used in, .e.g., Charikar and Wirth [17]). Its proof appears in Appendix B.

<sup>3</sup> Intuitively, for every clustering  $\mathcal{C}$  we create a matching  $M$  between  $V_1$  and  $V_2$  such that every matched pair of nodes appears in the same cluster in  $\mathcal{C}$  (note that there can be more than one such matching). Thus, every clustering  $\mathcal{C}$  must incur a loss due to the  $-$  edges whose value is at least the total weight of  $-$  edges in  $M$ , i.e.,  $w(M \cap E^-)$ .

► **Lemma 8.** *For every clustering  $\mathcal{C}$  there is a clustering  $\mathcal{S} = \{S, \bar{S}\}$  satisfying:  $v(\mathcal{S}) \geq v^+(\mathcal{C}) + \frac{1}{2}v^-(\mathcal{C})$ .*

The following lemma reduces **Fair-Max-Agreement** to **Fair-Max-Agreement**[2] with bounded loss in the approximation factor (its proof appears in Appendix C).

► **Lemma 9.** *If there is an  $\alpha$ -approximation algorithm for **Fair-Max-Agreement**[2] with two colors and a ratio of  $1 : 1$ , then there is a  $(2\alpha)/(2 + \alpha)$ -approximation algorithm for **Fair-Max-Agreement** with two colors and a ratio of  $1 : 1$ .*

We note that if  $\alpha > 2/3$  then Lemma 9 implies an approximation better than  $1/2$  for **Fair-Max-Agreement**. Therefore, we focus our attention now on presenting an approximation that is strictly better than  $2/3$  for **Fair-Max-Agreement**[2] assuming two colors and a ratio of  $1 : 1$ . To achieve this goal we define the following optimization problem.

► **Definition 10.** *The **Max-Agreement-Bisection** problem is defined as follows. Given an edge weighted graph  $G = (V, E)$  equipped with non-negative edge weights  $w : E \rightarrow \mathbb{R}_+$ , where each edge is labeled either  $+$  or  $-$ , the task is to partition the nodes into two clusters of equal size so as to maximize the overall agreement, i.e.,*

$$\max_{S \subseteq V : |S| = n/2} \{w^-(\delta(S)) + w^+(E(S)) + w^+(E(\bar{S}))\}.$$

It is important to note that in **Max-Agreement-Bisection** there are no colors, therefore no fairness constraints. Nonetheless, relying on the fact that the number of colors is only two and the ratio is  $1 : 1$ , we present an approximation preserving reduction from **Fair-Max-Agreement**[2] to **Max-Agreement-Bisection**. This is summarized in the following lemma.

► **Lemma 11.** ***Fair-Max-Agreement**[2] with two colors and a ratio of  $1 : 1$  has an approximation preserving reduction to **Max-Agreement-Bisection**.*

**Proof.** We are given an instance of **Fair-Max-Agreement**[2] with two colors and a ratio of  $1 : 1$ . I.e., a graph  $G = (V_1 \cup V_2, E^+ \cup E^-)$  where  $|V_1| = |V_2|$ . We construct an instance for **Max-Agreement-Bisection** as follows. Consider the graph  $\tilde{G} = (V_1 \cup V_2, \tilde{E}^+ \cup \tilde{E}^-)$  where

$$\tilde{E}^+ \triangleq \{(u, v) \in E^+ \mid c(u) = c(v)\} \cup \{(u, v) \in E^- \mid c(u) \neq c(v)\}$$

$$\tilde{E}^- \triangleq \{(u, v) \in E^- \mid c(u) = c(v)\} \cup \{(u, v) \in E^+ \mid c(u) \neq c(v)\}.$$

For every solution  $\mathcal{S} = \{S, \bar{S}\}$  for **Max-Agreement-Bisection** we efficiently construct a solution  $\mathcal{S}' = \{S', \bar{S}'\}$  for **Fair-Max-Agreement**[2]. Let  $\mathcal{S} = \{S, \bar{S}\}$  be a solution to the former problem, we construct a clustering  $\mathcal{S}' = \{S', \bar{S}'\}$  as follows:  $S' = \{u \in S \mid c(u) = 1\} \cup \{u \in \bar{S} \mid c(u) = 2\}$ . One can note that  $S'$  is obtained from  $S$  by swapping the side of the cut all vertices of color 2 reside in.

Note that every edge  $e \in E$  which was in agreement in the solution  $\{S, \bar{S}\}$  for **Max-Agreement-Bisection**, has a corresponding edge  $\tilde{e} \in \tilde{E}$  which is in agreement in the solution  $\{S', \bar{S}'\}$  for **Fair-Max-Agreement**[2], and vice versa. Thus,  $v(\mathcal{S}) = v(\mathcal{S}')$ , i.e. the value of the solution remains the same. All that remains to prove is that  $\mathcal{S}'$  satisfies the fairness constraints. First, one can note that  $|V_1 \cap S| = n/2 - |V_2 \cap S|$  since  $|S| = n/2$ . Moreover,  $n/2 - |V_2 \cap S| = |V_2 \cap \bar{S}|$  since  $|V_2| = n/2$  (recall that the ratio is  $1 : 1$  and there are  $n$  vertices on total). From the definition of  $S'$  we can infer that:  $|V_2 \cap S'| = |V_2 \cap \bar{S}|$ . This proves  $|V_2 \cap S'| = |V_1 \cap S'|$ , i.e.,  $S'$  satisfies the fairness constraints (and therefore  $\bar{S}'$  also satisfies the fairness constraints). This concludes the proof. ◀



We emphasize that the above approach of reducing **Fair-Max-Agreement** [2] to a graph bisection problem, heavily relies on the fact that there are only two colors with a ratio of 1 : 1 and it fails for a general instance. Thus, for general instances a different approach is required.

In order to cope with **Max-Agreement-Bisection** we apply the approach of Raghavendra and Tan [37], and the subsequent works of [10, 42], which is based on rounding a Lasserre SDP hierarchy relaxation.

Following Halperin and Zwick [31] and Han, Ye and Zhang [32], we present a general graph bisection problem. This problem is parametrized by four coefficients  $c_0, c_1, c_2, c_3$  and is defined as follows (via a quadratic formulation):

$$\begin{aligned} \max \quad & \sum_{e=(i,j) \in E} w(e)(c_0 + c_1 x_i x_j + c_2 x_i x_j + c_3 x_i x_j) \\ \text{s.t.} \quad & \sum_{i \in V} x_i = 0 \\ & x_i^2 = 1 \quad 0 \leq i \leq n \end{aligned}$$

Note that in this problem,  $x_i \in \{\pm 1\}$  for every  $i \in V$ , since the last constraint is  $x_i^2 = 1$ . Therefore, the first constraint,  $\sum_{i \in V} x_i = 0$  is equivalent to the fact that exactly half the variables equal 1 and the other half equal  $-1$ .

The coefficients  $c_0, c_1, c_2, c_3$  depend on the exact graph bisection problem which we aim to solve. For example, when considering **Max-Bisection** the coefficients are  $c_0 = 1/2, c_1 = 0, c_2 = 0, c_3 = -1/2$ . Additionally, when considering **Max- $\frac{n}{2}$ -Uncut** the coefficients are  $c_0 = 1/2, c_1 = 0, c_2 = 0, c_3 = 1/2$ .

We note that **Max-Agreement-Bisection** resembles both **Max-Bisection** and **Max- $\frac{n}{2}$ -Uncut** since: (1) in all three problems we aim to find a cut that contains exactly half of the vertices; and (2) the objective of **Max-Agreement-Bisection** can be seen as the sum of the objectives of **Max-Bisection** and **Max- $\frac{n}{2}$ -Uncut** on two graphs over the same set of vertices  $V$  (the graph  $(V, E^-)$  corresponds to the **Max-Bisection** objective whereas  $(V, E^+)$  corresponds to the **Max- $\frac{n}{2}$ -Uncut** objective). Therefore, the objective for **Max-Agreement-Bisection** can be formally written as follows:

$$\sum_{e=(i,j) \in E^+} w(e) \left( \frac{1}{2} + \frac{1}{2} x_i x_j \right) + \sum_{e=(i,j) \in E^-} w(e) \left( \frac{1}{2} - \frac{1}{2} x_i x_j \right).$$

Equivalently, **Max-Agreement-Bisection** can be seen as an extension of the general graph bisection problem in which the  $c_0, \dots, c_3$  coefficients are not uniform over the edges of the graph. Specifically, for  $+$  edges the coefficients are  $c_0 = 1/2, c_1 = 0, c_2 = 0, c_3 = 1/2$  and for  $-$  edges the coefficients are  $c_0 = 1/2, c_1 = 0, c_2 = 0, c_3 = -1/2$ .

Our main observation is that the algorithm and analysis of [10] for **Max-Bisection**, and the followup work of [42] for the general graph bisection problem described above, can be extended to **Max-Agreement-Bisection** with virtually no change in the analysis. We refer to Appendix D to the high level details as to why **Max-Agreement-Bisection** admits an approximation of 0.8776 (an approximation of 0.8776 is the guarantee [10] proved for **Max-Bisection** and [42] for **Max- $\frac{n}{2}$ -Uncut**). This enables us to obtain the following Corollary.

► **Corollary 12.** *Fair-Max-Agreement [2] with two colors and a ratio of 1 : 1 is approximable in polynomial time to within a factor 0.8776.*

**Proof of Theorem 3.** Follows from Corollary 12 and Lemma 9. ◀

## 2.2 Approximating General Instances

For general instances (either  $k > 2$  or non-uniform ratios) the approximation guarantees we provide are slightly worse than for instances with two colors and a ratio of  $1 : 1$ . The main use of Lemma 8 is that there is a good solution that has only two clusters. It is important to note that this lemma holds for any number of colors and any ratios, hence it also applies to general instances. However, Lemma 9, which reduces the problem to the two cluster variant, does not hold for a general instance with the exact same guarantee. The reason is that even for the case of uniform ratios  $1 : \dots : 1$  and  $k$  colors we are required to find a min cost perfect matching in a  $k$ -partite graph (where the cost of a hyperedge is the total weight of edges between nodes in the hyperedge). Hence, we no longer can find in polynomial time a clustering  $\mathcal{C}$  which satisfies the condition  $v^-(\mathcal{C}) \geq v^-(\mathcal{C}^*)$ . To overcome the above difficulty we use a different approach in which we randomly choose a clustering that is based on a random  $k$ -partite matching between the colors. This approach incurs only a small loss in the approximation guarantee. Let us first describe the simple randomized approximation algorithm which obtains an approximation ratio of  $1/2 - o(1)$  and then describe how to improve upon this ratio.

### 2.2.1 Simple $(1/2 - o(1))$ -Approximation

Our random clustering algorithm is summarized in Algorithm 1.

■ **Algorithm 1** Random  $k$ -Partite Matching.

---

```

Input: $G = (V_1 \cup \dots \cup V_k, E)$, $\{p_i\}_{i=1}^k$;
 $\mathcal{C} \leftarrow \emptyset$;
while $V \neq \emptyset$ do
 $C \leftarrow \emptyset$;
 for $i \leftarrow 1$ to k do
 Let S_i be a uniform random set of p_i nodes from V_i ;
 $C \leftarrow C \cup S_i$;
 $V_i \leftarrow V_i \setminus S_i$;
 end
 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$;
end
return \mathcal{C} ;

```

---

► **Observation 13.** Let  $G = (V, E)$  be a graph with  $k \geq 2$  colors and ratios of  $p_1 : p_2 : \dots : p_k$ . Let  $\mathcal{C}$  be the output of Algorithm 1. Then  $\mathbb{E}[v^-(\mathcal{C})] \geq (1 - (\sum_{i=1}^k p_i)/n) \cdot w(E^-)$ .

The following extends Theorem 7, which provided an approximation of  $1/2$  for the case there are two colors and a ratio of  $1 : 1$ , to a general number of colors  $k$  and ratios  $p_1 : \dots : p_k$  while suffering a small loss of  $(\sum_{i=1}^k p_i)/(2n)$  in the approximation guarantee. Note that if  $\sum_{i=1}^k p_i = o(n)$  then this loss is at most  $o(1)$ . This is achieved by replacing the clustering that corresponds to the matching  $M^-$  of Observation 6 with the clustering  $\mathcal{C}$  generated by Algorithm 1.

► **Theorem 14.** There is a randomized polynomial time  $(1/2 - (\sum_{i=1}^k p_i)/(2n))$ -approximation algorithm for *Fair-Max-Agreement* in general weighted graphs with  $k \geq 2$  colors and ratios of  $p_1 : p_2 : \dots : p_k$ .

**Proof.** The algorithm chooses the best from the following two solutions: a single cluster containing all the nodes or the output of Algorithm 1. The value of the former clustering is the total weight of  $+$  edges, i.e.,  $w(E^+)$ . On the other hand, the expected value of the latter clustering is at least  $(1 - (\sum_{i=1}^k p_i)/n) \cdot w(E^-)$  (see Observation 13). Let us denote by  $\mathcal{C}_{\text{ALG}}$  the chosen clustering, i.e.,  $\mathcal{C}_{\text{ALG}} = \arg \max\{v(\mathcal{C}), v(\{V\})\}$ . One can note that  $\mathbb{E}[v(\mathcal{C}_{\text{ALG}})] \geq 1/2 \cdot (w(E^+) + (1 - (\sum_{i=1}^k p_i)/n) \cdot w(E^-)) \geq (1/2 - (\sum_{i=1}^k p_i)/(2n)) \cdot w(E) \geq (1/2 - (\sum_{i=1}^k p_i)/(2n)) \cdot \text{OPT}$ .  $\blacktriangleleft$

### 2.2.2 Beating the $(1/2 - o(1))$ -Approximation Ratio

The following lemma shows that one can reduce **Fair-Max-Agreement** to **Fair-Max-Agreement**[2], for general instances, with a small loss in the approximation guarantee (similarly to Lemma 9). If  $\sum_{i=1}^k p_i = o(n)$  then the approximation guarantee of the following lemma equals  $(2\alpha)/(2 + \alpha) - o(1)$ .

► **Lemma 15.** *If there is an  $\alpha$ -approximation algorithm for **Fair-Max-Agreement**[2] with  $k \geq 2$  colors and ratios of  $p_1 : \dots : p_k$ , then there is a  $(1 - \frac{\sum_{i=1}^k p_i}{n})(\frac{1}{\alpha}(\frac{2+\alpha}{2} - \frac{\sum_{i=1}^k p_i}{n}))^{-1}$ -approximation algorithm for **Fair-Max-Agreement** with  $k \geq 2$  colors and ratios of  $p_1 : \dots : p_k$ .*

**Proof.** The proof is similar to the proof of Lemma 9 except that instead of choosing the best of two solutions when one of them has a value of at least  $v^-(\mathcal{C}^*)$  for some optimal clustering  $\mathcal{C}^*$ , we need to settle for a solution with value  $(1 - (\sum_{i=1}^k p_i)/n) \cdot v^-(\mathcal{C}^*)$  (swapping the clustering Observation 6 guarantees with the clustering Observation 13 guarantees). The rest of the calculations are similar.  $\blacktriangleleft$

All that remains is to find a good approximation for **Fair-Max-Agreement**[2]. When considering an approximation better than  $1/2$ , we note that the approach taken for two colors and a ratio of  $1 : 1$ , which reduces **Fair-Max-Agreement**[2] to **Max-Agreement-Bisection**, does not work for general instances. Instead we take the approach of Abbasi-Zadeh, Bansal, Guruganesh, Nikolov, Schwartz and Singh [43] which presented the problem of **Max-Cut** with side constraints (denoted by **Max-Cut-Sc**) and an algorithm for it. In the **Max-Cut-Sc** problem we are given an  $n$ -vertex graph  $G = (V, E)$ , a collection  $\mathcal{F} = \{F_1, \dots, F_k\}$  of  $k$  subsets of  $V$ , and cardinality bounds  $b_1, \dots, b_k \in \mathbb{N}$ . The goal is to find a subset  $S \subseteq V$  that maximizes the total weight of edges that cross  $S$ , subject to satisfying  $|S \cap F_i| = b_i$  for all  $1 \leq i \leq k$ . Their algorithm uses a sticky Brownian motion for the rounding process of a suitable semi-definite relaxation for **Max-Cut-Sc**. In order to utilize this approach we: (1) present a generalization of **Max-Cut-Sc** which also handles uncut edges (this problem is denoted by **Max-Agreement-Sc**); and (2) prove that the rounding approach of [43] can handle uncut edges, i.e.,  $+$  edges, with the same approximation guarantee of the cut edges.<sup>4</sup> Formally, the input for the **Max-Agreement-Sc** problem is the same as **Max-Cut-Sc**, with the addition that every edge is labeled either with a  $+$  or a  $-$ . The goal is to find a subset  $S \subseteq V$  that maximizes  $w^-(\delta(S)) + w^+(E(S)) + w^+(E(\bar{S}))$  subject to the same constraints as in **Max-Cut-Sc**.

<sup>4</sup> The algorithm of [43] for **Max-Cut-Sc** in fact takes the best out of two solutions: the Brownian motion rounding and randomized rounding. The latter algorithm is needed for the case that the value of the optimal solution is small. In our case we prove that the instance for which we need to solve **Max-Agreement-Sc** cannot have an optimal solution of small value (see Lemma 17), thus our algorithm just utilizes the Brownian motion approach.

Following the above discussion, we show how one can handle **Fair-Max-Agreement** [2] by solving a sequence of **Max-Agreement-Sc** instances with an appropriate choice of cardinality bounds. Specifically, given an instance of **Fair-Max-Agreement** [2] with  $k$  colors, we choose  $\mathcal{F} = \{V_1, \dots, V_k\}$  and all the cardinality bounds  $b_i = h \cdot p_i$  for all  $i = 1, \dots, k$ . The sequence of **Max-Agreement-Sc** is defined by enumerating over all values of  $h$  in the range  $h = 1, \dots, n / \sum_{i=1}^k p_i$ . Finding a solution to this problem for all possible  $h$  values captures the fairness constraints. This is true since an optimal clustering of **Fair-Max-Agreement** [2] corresponds to some specific (unknown) value of  $h$  in the above range. Note that an instance as above to the **Max-Agreement-Sc** is always feasible for every value of  $h$  in the range between 1 and  $n / \sum_{i=1}^k p_i$ . This is summarized in Algorithm 2.

■ **Algorithm 2** **Fair-Max-Agreement** [2] via **Max-Agreement-Sc**.

---

**Input:**  $G = (V_1 \cup \dots \cup V_k, E^+ \cup E^-)$ ,  $\{p_i\}_{i=1}^k$ ;  
 $\mathcal{F} \leftarrow \{V_1, \dots, V_k\}$ ;  
**for**  $h \leftarrow 1$  **to**  $n / \sum_{i=1}^k p_i$  **do**  
     $b_i \leftarrow h \cdot p_i \quad \forall 1 \leq i \leq k$ ;  
    Let  $\mathcal{C}_h$  be the sticky Brownian motion solution for **Max-Agreement-Sc** with  
    bounds  $\{b_i\}_{i=1}^k$ ;  
**end**  
**return**  $\operatorname{argmax}\{v(\mathcal{C}_h) : h = 1, \dots, n / \sum_{i=1}^k p_i\}$ ;

---

The bulk of the effort is in solving **Max-Agreement-Sc** via the Brownian motion approach of [43]. Theorem 3 in [43] provides, for every  $\varepsilon > 0$ , an algorithm for **Max-Cut-Sc** whose running time is  $O(n^{\operatorname{poly}(\log(k)/\varepsilon)})$  which finds a solution  $S \subseteq V$  with the following properties (in what follows  $S^*$  is an optimal solution for **Max-Cut-Sc**): (1)  $\mathbb{E}[w^-(\delta(S))] \geq (0.843 - \varepsilon) \cdot w^-(\delta(S^*))$ ; and (2) for every color  $i = 1, \dots, k$   $|S \cap V_i| - b_i| \leq \varepsilon n$ . We prove that the rounding algorithm of [43] can also handle, in addition to the above two properties, the contribution of the  $+$  edges. Specifically, we show how one can easily change the SDP relaxation and then apply the rounding algorithm of [43] to also guarantee that

$$\mathbb{E}[w^+(E(S)) + w^+(E(\bar{S}))] \geq (0.843 - \varepsilon) \cdot (w^+(E(S^*)) + w^+(E(\bar{S}^*))),$$

where  $S^*$  here denotes an optimal solution to **Max-Agreement-Sc** for the correct choice of  $h$ .

First, let us start by formulating **Max-Agreement-Sc** as a quadratic optimization problem:

$$\begin{aligned} \max \quad & \sum_{e=(i,j) \in E^-} w(e) \cdot (x_i - x_j)^2 + \sum_{e=(i,j) \in E^+} w(e) \cdot (1 - (x_i - x_j)^2) \\ \text{s.t.} \quad & \sum_{j \in F_i} x_j = b_i & i = 1, \dots, k \\ & x_j \cdot (1 - x_j) = 0 & j = 1, \dots, n \end{aligned}$$

Recall that in the above  $b_i$  equals  $p_i \cdot h$  (for some value of  $h$ ).

We denote the above quadratic problem by  $Q$  and the solutions to the  $\ell$ -level Lasserre strengthening of the standard SDP relaxation of  $Q$  by  $SoS(Q)$ . A solution in  $SoS(Q)$  can be represented by a collection of unit vectors  $\mathbf{v}_S$  for all subsets  $S \subseteq V$  ( $|S| \leq \ell$ ). For completeness we write the  $\ell$ -round SDP relaxation for the problem:

$$\max \quad \sum_{e=(i,j) \in E^-} w(e) \cdot \|\mathbf{v}_i - \mathbf{v}_j\|^2 + \sum_{e=(i,j) \in E^+} w(e) \cdot (1 - \|\mathbf{v}_i - \mathbf{v}_j\|^2)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j \in F_i} \mathbf{v}_0 \cdot \mathbf{v}_j = b_i & i = 1, \dots, k \\
& \mathbf{v}_0 \cdot \mathbf{v}_0 = 1 \\
& \mathbf{v}_{S_1} \cdot \mathbf{v}_{S_2} = \mathbf{v}_{S_3} \cdot \mathbf{v}_{S_4} & \forall S_1, S_2, S_3, S_4 \subseteq V, \\
& & S_1 \cup S_2 = S_3 \cup S_4 \\
& & \text{and } |S_1 \cup S_2| \leq \ell \\
& \mathbf{v}_0 \cdot \mathbf{v}_i + \mathbf{v}_j \cdot \mathbf{v}_0 - \mathbf{v}_i \mathbf{v}_j \leq 1 & 1 \leq i, j \leq n \\
& \mathbf{v}_i \cdot \mathbf{v}_0 \geq \mathbf{v}_i \cdot \mathbf{v}_j & 1 \leq i, j \leq n \\
& \mathbf{v}_i \cdot \mathbf{v}_j \geq 0 & 1 \leq i, j \leq n
\end{aligned}$$

For completeness of presentation, let us now focus on defining the rounding algorithm of [43], as we require some of the notations in order to present the analysis of the uncut + edges. Recall that  $\tilde{\mathbf{w}}_i \triangleq \mathbf{v}_i - \mu_i \mathbf{v}_0$  and  $\mathbf{w}_i \triangleq \tilde{\mathbf{w}}_i / \|\tilde{\mathbf{w}}_i\|_2$ . Let  $\mathbf{W}$  and  $\tilde{\mathbf{W}}$  be the PSD correlation matrices defined by the above vectors, that is  $\mathbf{W}_{ij} = \mathbf{w}_i \cdot \mathbf{w}_j$  and  $\tilde{\mathbf{W}}_{ij} = \tilde{\mathbf{w}}_i \cdot \tilde{\mathbf{w}}_j$ , for every  $1 \leq i, j \leq n$ . The following lemma is used to obtain the input vectors to the rounding algorithm, this lemma is based on [13] and [30] and appears as Lemma 10 in [43]. We refer the reader to [13, 30, 43] for its proof.

► **Lemma 16.** *Let  $\varepsilon_0 \leq 1, \ell \geq 1/\varepsilon_0^4 + 2$ , for any solution in  $\text{SoS}_\ell(Q)$  where  $\ell \geq 1/\varepsilon_0^4 + 2$ , there exists an efficiently computable solution in  $\text{SoS}_{\ell-1/\varepsilon_0^4}(Q)$  such that  $\sum_{i=1}^n \sum_{j=1}^n \tilde{\mathbf{W}}_{ij}^2 \leq \varepsilon_0^4 n^2$ .*

Second, let us focus on the rounding algorithm. The input to the rounding algorithm is the vectors obtained by Lemma 16. To round the vectors, the algorithm performs a sticky Brownian motion inside the hypercube  $[0, 1]^n$ , that is the random process  $\{\mathbf{X}_t\}_{t \geq 0}$  which is defined as follows. The starting point of the random walk is  $\mathbf{X}_0$  such that  $(\mathbf{X}_0)_i = \mu_i$  for every  $1 \leq i \leq n$ . Denote  $\{\mathbf{B}_t\}_{t \geq 0}$  as the standard Brownian motion in  $\mathbb{R}^n$ . Let  $\tau_1 = \inf\{t : \mathbf{X}_0 + \mathbf{W}^{1/2} \mathbf{B}_t \notin [0, 1]^n\}$ , then for all  $0 \leq t \leq \tau_1$ :  $\mathbf{X}_t = \mathbf{X}_0 + \mathbf{W}^{1/2} \mathbf{B}_t$ . Let  $A_t = \{i | (\mathbf{X}_t)_i \neq 0, 1\}$  be the collection of active nodes at time  $t$ , and  $F_t = \{\mathbf{x} \in [0, 1]^n | x_i = (\mathbf{X}_t)_i, \forall i \notin A_t\}$ . The covariance matrix  $\mathbf{W}_t$  used for the random walk at time  $t$  is based on  $\mathbf{W}$  and an entry in this matrix is not 0 only for the indices in  $A_t$ , i.e.,  $(\mathbf{W}_t)_{ij} = \mathbf{W}_{ij}$  if  $i, j \in A_t$  (otherwise  $(\mathbf{W}_t)_{ij} = 0$ ). After time  $\tau_1$  the random process is changed to  $\mathbf{X}_t = \mathbf{X}_{\tau_1} + \mathbf{W}_{\tau_1}^{1/2} (\mathbf{B}_t - \mathbf{B}_{\tau_1})$ , it is defined for  $\tau_1 \leq t \leq \tau_2$  where  $\tau_2 = \inf\{t : \mathbf{X}_{\tau_1} + \mathbf{W}_{\tau_1}^{1/2} (\mathbf{B}_t - \mathbf{B}_{\tau_1}) \notin F_{\tau_1}\}$ . In general,  $\tau_i = \inf\{t : \mathbf{X}_{\tau_{i-1}} + \mathbf{W}_{\tau_{i-1}}^{1/2} (\mathbf{B}_t - \mathbf{B}_{\tau_{i-1}}) \notin F_{\tau_{i-1}}\}$  and when  $\tau_{i-1} \leq t \leq \tau_i$  the process is defined as follows:  $\mathbf{X}_t = \mathbf{X}_{\tau_{i-1}} + \mathbf{W}_{\tau_{i-1}}^{1/2} (\mathbf{B}_t - \mathbf{B}_{\tau_{i-1}})$ . The algorithm does not terminate at time  $\tau_n$  but it is stopped at a fixed pre-specified time  $\tau$  (which is chosen to be  $\Theta(\log(1/\varepsilon))$ ) and rounds to 1 the remaining nodes  $i \in A_\tau$  with probability  $(\mathbf{X}_\tau)_i$ . The output cut  $S \subseteq V$  contains all the nodes  $i$  for which  $(\mathbf{X}_\tau)_i = 1$ .

As previously mentioned, the algorithm for **Max-Cut-Sc** in [43] distinguishes between two cases. For instances with small optimal value a different approach was taken instead of the Brownian motion approach described above. However, since we use a sequence of **Max-Agreement-Sc** instances to solve **Fair-Max-Agreement** [2], this case is not possible due to the following lemma. It states that the optimal value of **Fair-Max-Agreement** [2] is not small, hence for the correct choice of  $h$  the optimal value of **Max-Agreement-Sc** is also not small. Thus, we can focus solely on instances whose optimal value is not small.

► **Lemma 17.** *The optimal value of an instance  $G = (V_1 \cup \dots \cup V_k, E^+ \cup E^-)$  to **Fair-Max-Agreement** [2] is at least  $(1/2 - \sum_{i=1}^k p_i / (2n))w(E)$ .*

**Proof.** Let  $\mathcal{C}$  be the output of Algorithm 1. The simple algorithm which outputs  $\mathcal{S} = \{S, \bar{S}\}$  by placing all the nodes of each cluster  $C \in \mathcal{C}$  together in  $S$  or  $\bar{S}$  with probability  $1/2$  (and independently over the different clusters in  $\mathcal{C}$ ) results in a solution to **Fair-Max-Agreement** [2] with an expected value of  $(1/2 - \sum_{i=1}^k p_i / (2n))w(E)$ . This is true since  $\mathbb{E}[v^+(\mathcal{S})] = 1/2 \cdot w(E^+)$  and  $\mathbb{E}[v^-(\mathcal{S})] = 1/2 \cdot (1 - \sum_{i=1}^k p_i / n)w(E^-)$  (the latter follows from Observation 13) ◀

The analysis of the Brownian motion based algorithm relies heavily on the following theorem which appears in [36]. Intuitively, this theorem captures the connection between diffusion processes and partial differential equations (see chapter 9 in [36]). We present it here since we require it for the analysis of the + edges.

► **Theorem 18** (Theorem 9 in [43] and Theorem 9.14 in [36]). *Given a domain  $D = (0, 1)^2 \subseteq \mathbb{R}^2$ , suppose  $L$  is uniformly elliptic in  $D$  of the form*

$$L = \sum_{i,j=1}^2 a_{ij}(x) \frac{\partial^2}{\partial x_i \partial x_j}$$

where  $\frac{1}{2}\sigma\sigma^T = [a_{ij}]$  for  $\sigma \in \mathbb{R}^{2 \times 2}$ . For  $x \in D$  consider the process  $X_t = X_0 + \sigma B_t$ . Denote  $\tau_D = \inf\{t > 0; X_t \notin D\}$  (the stopping time of  $X_t$ ). Let  $\phi$  be a bounded continuous function on  $\partial D$ . Put  $u(x) = E^x[\phi(X_{\tau_D})]$  where  $E^x$  denotes the expected value when  $X_0 = x$ . Then  $u$  solves the Dirichlet problem

1.  $Lu = 0$  in  $D$ .
2.  $\lim_{x \rightarrow y} u(x) = \phi(y)$  for all regular  $y \in \partial D$ .

Let us fix two nodes  $i$  and  $j$  and denote  $\bar{\mathbf{X}}_t$  as the projection of the random process  $\mathbf{X}_t$  to the coordinates  $i, j$  and let

$$\bar{\mathbf{W}} = \begin{pmatrix} 1 & \rho_{ij} \\ \rho_{ij} & 1 \end{pmatrix}.$$

In our analysis  $\sigma = \bar{\mathbf{W}}^{1/2}$ , i.e., the entries  $a_{ij}$  in the above theorem are the entries of  $\bar{\mathbf{W}}$ . When performing an edge-wise analysis we can consider the projection of  $\mathbf{X}_t$  we described above. We note that the first guarantee in the following lemma is identical to [43], but is included for completeness. The novelty of the following theorem lies in the second guarantee.<sup>5</sup>

► **Lemma 19.** *Let  $i, j \in V$  and  $\mathbf{v}_i, \mathbf{v}_j$  the corresponding vectors in the SDP solution. It holds that*

1.  $\Pr[\mathbf{X}_{\tau_{ni}} \neq \mathbf{X}_{\tau_{nj}}] \geq 0.843 \cdot \|\mathbf{v}_i - \mathbf{v}_j\|^2$ .
2.  $\Pr[\mathbf{X}_{\tau_{ni}} = \mathbf{X}_{\tau_{nj}}] \geq 0.843 \cdot (1 - \|\mathbf{v}_i - \mathbf{v}_j\|^2)$ .

**Proof.** Guarantee 1 of the lemma is identical to Lemma 11 in [43], and thus its proof is omitted.

Let us focus on guarantee 2 above. Let us denote  $\theta_{ij} = \arccos(\mathbf{w}_i \cdot \mathbf{w}_j)$ . Recall that  $\|\mathbf{v}_i\|^2 = \mu_i$  and  $\mathbf{v}_i = \mu_i \cdot \mathbf{v}_0 + \sqrt{\mu_i - \mu_i^2} \cdot \mathbf{w}_i$ . Therefore, the contribution of the + edges to the objective of the relaxation can be re-written as follows:

$$1 - \|\mathbf{v}_i - \mathbf{v}_j\|^2 = 1 - (\mu_i + \mu_j - 2 \cdot \mu_i \cdot \mu_j + 2 \cos(\theta_{ij}) \cdot \sqrt{(\mu_i - \mu_i^2) \cdot (\mu_j - \mu_j^2)}).$$

For simplicity we denote  $x = \mu_i, y = \mu_j, \theta = \theta_{ij}$  and the expression above as  $SDP(x, y, \theta)$ . Observe that the probability that the edge  $(i, j)$  is uncut equals the probability that the Brownian motion  $\bar{\mathbf{X}}_t$  is absorbed in  $(0, 0)$  or  $(1, 1)$ . Denote  $u_\theta(x, y)$  as the probability of ending in  $(0, 0)$  or  $(1, 1)$  conditioned on starting the walk at point  $(x, y)$ :

$$u_\theta(x, y) = \Pr[(\mathbf{X}_{\tau_{ni}} = \mathbf{X}_{\tau_{nj}}) | (\mathbf{X}_0)_i = x, (\mathbf{X}_0)_j = y].$$

<sup>5</sup> We mention that one can derive Lemma 19 via rotational symmetry of the boundary conditions of  $\partial[0, 1]^2$  for both cut and uncut edges, and similar rotational symmetry of the contribution to the SDP relaxation of both cut and uncut edges.

Observe that the boundary condition on  $u_\theta(x, y)$  is the following:  $u_\theta(x, y) = 1 - (x + y - 2xy) \quad \forall (x, y) \in \partial[0, 1]^2$ . Following Theorem 18  $u_\theta$  is the unique solution to the Dirichlet problem:

$$\frac{\partial^2 u_\theta}{\partial x^2} + \frac{\partial^2 u_\theta}{\partial y^2} + 2 \cos(\theta) \frac{\partial^2 u_\theta}{\partial x \partial y} = 0 \quad \forall (x, y) \in \text{Int}[0, 1]^2$$

$$u_\theta(x, y) = 1 - (x + y - 2xy) \quad \forall (x, y) \in \partial[0, 1]^2$$

The problem above can be numerically solved for any configuration  $(x, y, \theta)$ . Therefore, the approximation ratio for uncut edges is  $\min_{(x, y, \theta) \in F} \frac{u_\theta(x, y)}{\text{SDP}(x, y, \theta)}$  where  $F$  is the collection of all feasible configurations. Specifically,  $(x, y, \theta) \in F$  if it satisfies the triangle inequalities which are derived from the  $\ell$ -round SDP relaxation (see Appendix D Lemma 11 [43]). The numerical calculation via adaptation of the code used in [43] results in an approximation ratio of 0.843 for the uncut edges. ◀

Lemmas 19 and 17 are sufficient to extend the proof of Theorem 3 of [43] to **Fair-Max-Agreement** [2], this is summarized in the following theorem.

► **Theorem 20.** *There exists a  $O(n^{\text{poly}(\log(k)/\varepsilon)})$ -time algorithm for **Fair-Max-Agreement** [2], which for an instance  $G = (V_1 \cup \dots \cup V_k, E)$  outputs a  $(0.843 - \varepsilon, \varepsilon)$ -approximation with high probability.*

**Proof of Theorem 4.** Follows from Theorem 20 and Lemma 15. ◀

### 3 Hardness of Fair-Min-Disagreement

In this section we present the hardness results for **Fair-Min-Disagreement**. First we prove Theorem 1.

**Proof of Theorem 1.** We present a reduction from the **3-Partition** problem, as defined in [8, 26]. In **3-Partition** we are given  $n = 3\ell$  integer numbers  $a_1, a_2, \dots, a_n$  and a threshold  $A$  such that  $\frac{A}{4} < a_i < \frac{A}{2}$  and  $\sum_{i=1}^n a_i = \ell A$  (where  $a_1, \dots, a_n$  and  $A$  are polynomial in  $n$ ). The goal is to decide if the numbers can be partitioned into triplets such that each triplet sums up to exactly  $A$ . This problem is known to be strongly NP-complete [26].

Given an instance of the **3-Partition** problem we construct a graph for the **Fair-Min-Disagreement** problem as follows (we denote the two colors by red and blue). For each number  $a_i$  construct a clique with  $a_i$  red nodes, the edges in this clique are all labeled with  $+$ . Additionally, construct  $\ell$  cliques where each of them contains  $A$  blue nodes and the edges within such a clique are all labeled with  $+$ . For every pair of blue nodes which are not in the same clique, place an edge between them which is labeled with  $-$ . This finishes the definition of our instance for **Fair-Min-Disagreement**.

We claim that there is a solution to the given **3-Partition** instance if and only if there is a clustering of the **Fair-Min-Disagreement** instance whose cost is zero. Given a solution to the **3-Partition** instance we can construct a clustering of zero cost as follows. For each triplet  $a_{i_1}, a_{i_2}, a_{i_3}$  in the solution for **3-Partition** (recall that  $a_{i_1} + a_{i_2} + a_{i_3} = A$ ), define a cluster which contains the three red cliques corresponding to the numbers  $a_{i_1}, a_{i_2}, a_{i_3}$  and a single blue clique of size  $A$ . One can note that this is a valid, i.e., fair, clustering since the number of red and blue nodes is equal in all clusters. Furthermore, there are no unclustered nodes since  $\sum_{i=1}^n a_i = \ell A$ . The cost of this clustering is zero since: (1) all cliques, either red or blue, are contained as a whole in a single cluster, and thus all  $+$  edges are in agreement; and (2) every cluster contains exactly a single blue clique, and thus all  $-$  edges are also in agreement.



Given a clustering of cost zero we prove that one can partition the numbers to triplets such that the sum of each triplet is exactly  $A$ . Note that each clique, either red or blue, in the graph is contained as a whole in a single cluster, otherwise there is a  $+$  edges that is in disagreement which stands in contradiction to the fact that the clustering has zero cost. Moreover, each cluster contains exactly a single blue clique as a whole. The reason is that there cannot be no blue cliques in the cluster (if this occurs then the cluster has no blue nodes at all and this contradicts the fact the clustering is fair) and there cannot be two or more blue cliques in the cluster (if this occurs the cluster contains a  $-$  edge and this contradicts the fact the clustering has zero cost). Thus, the number of blue nodes in the cluster is  $A$ . Since the clustering is fair the number of red nodes in the cluster is also  $A$ . Recall that every number  $a_i$  satisfies that  $\frac{A}{4} < a_i < \frac{A}{2}$ . Hence, the cluster must contain exactly three red cliques that correspond to three numbers that sum up exactly to  $A$ . Therefore, the triplets we define as a solution to **3-Partition** are those that correspond to the three red cliques in each cluster. ◀

Let us now prove that a bi-criteria approximation is also not possible unless  $P = NP$ .

**Proof of Theorem 2.** We present a reduction from **Triangle-Partition**, which is known to be NP-hard [22]. In this problem the goal is to decide whether there is a set of node-disjoint triangles in a tripartite graph which covers all the nodes of the given tripartite graph. Note that without loss of generality one can assume that each of the three parts of the tripartite graph contains the same number of nodes. Otherwise, it is clear that the input graph cannot have all its nodes covered by node-disjoint triangles.

Given an instance  $G = (A \cup B \cup C, E)$  to **Triangle-Partition** we construct a graph  $G' = (A \cup B \cup C, E')$  for **Fair-Min-Disagreement** as follows. Each part of the three parts of  $G$  is given a unique color, i.e.,  $V_1 = A$ ,  $V_2 = B$ , and  $V_3 = C$ . Define the edges in  $G'$  as follows:  $E'^- \triangleq \{(u, v) | (u, v) \notin E\}$  and  $E'^+ \triangleq \emptyset$ . This finishes the definition of our instance for **Fair-Min-Disagreement**.

We claim that there is a solution to **Triangle-Partition** if and only if there is a solution  $\mathcal{C} = \{C_1, \dots, C_l\}$  to **Fair-Min-Disagreement** whose cost is zero and it satisfies that for every  $1 \leq r \leq l$ :  $|C_r \cap V_i|/|C_r \cap V_j| \leq (1 + \varepsilon)$  for every (ordered) pair of colors  $i$  and  $j$ .

Given a solution to **Triangle-Partition** we can construct a solution to **Fair-Min-Disagreement** by setting every triangle to be a different cluster. The nodes in each triangle are connected by edges in  $E$ . Therefore, there are no  $-$  edges between these nodes in  $E'$ . Since there are no  $+$  edges in  $E'$ , we can conclude that the cost of this solution for **Fair-Min-Disagreement** equals zero. Moreover, each cluster in the solution for **Fair-Min-Disagreement** contains exactly one node from each of the three colors. Hence, we proved the existence of the desired solution for **Fair-Min-Disagreement**.

Let  $\mathcal{C} = \{C_1, C_2, \dots, C_l\}$  be a solution to **Fair-Min-Disagreement** that has zero cost and satisfies that for every  $1 \leq r \leq l$ :  $|C_r \cap V_i|/|C_r \cap V_j| \leq (1 + \varepsilon)$  for every (ordered) pair of colors  $i$  and  $j$ . Note that for every  $1 \leq r \leq l$  and  $i = 1, 2, 3$ :  $|C_r \cap V_i| \leq 1$ . The reason for that is that two nodes of the same color are connected with a  $-$  edge in  $E'$ . Since  $\mathcal{C}$  has zero cost, any nodes of the same color cannot be in the same cluster. Moreover, for every  $1 \leq r \leq l$  and  $i = 1, 2, 3$ :  $|C_r \cap V_i| > 0$ . The reason for that is that if there is a (non-empty) cluster  $C_r$  and a color  $i$  for which  $|C_r \cap V_i| = 0$ , then  $C_r$  contains at least one node of color  $j$ ,  $j \neq i$ . For this ordered pair of colors ( $j$  and  $i$ ) the condition on  $\mathcal{C}$  is violated. Therefore, every cluster  $C_r$  contains exactly one node from every color, i.e., one node from every part of  $G$ . Because the cost of the clustering is zero, there are no  $-$  edges from  $E'$  inside each cluster which means that it forms a triangle in  $G$ . Clearly, these triangles are node-disjoint and contain all nodes in  $A \cup B \cup C$  since  $\mathcal{C}$  is a partition of  $A \cup B \cup C$ . This finishes the proof. ◀

## References

- 1 Saba Ahmadi, Sainyam Galhotra, Barna Saha, and Roy Schwartz. Fair correlation clustering, 2020. doi:10.48550/ARXIV.2002.03508.
- 2 Sara Ahmadian, Alessandro Epasto, Marina Knittel, Ravi Kumar, Mohammad Mahdian, Benjamin Moseley, Philip Pham, Sergei Vassilvitskii, and Yuyan Wang. Fair hierarchical clustering. *Advances in Neural Information Processing Systems*, 33:21050–21060, 2020.
- 3 Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, pages 267–275, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3292500.3330987.
- 4 Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Fair correlation clustering. *ArXiv*, abs/2002.02274, 2020.
- 5 Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke van Zuylen. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5):1110–1121, 2012. doi:10.1137/110848712.
- 6 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), November 2008. doi:10.1145/1411509.1411513.
- 7 Noga Amit. *The bichuster graph editing problem*. PhD thesis, Tel Aviv University, 2004.
- 8 Konstantin Andreev and Harald Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39:929–939, November 2006. doi:10.1007/s00224-006-1350-7.
- 9 Megasthenis Asteris, Anastasios Kyrillidis, Dimitris Papailiopoulos, and Alexandros Dimakis. Bipartite correlation clustering: Maximizing agreements. In *Artificial Intelligence and Statistics*, pages 121–129, 2016.
- 10 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. *ACM Transactions on Algorithms*, 13:1–27, October 2016. doi:10.1145/2907052.
- 11 Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413. PMLR, 2019.
- 12 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Machine Learning*, pages 238–247, 2002.
- 13 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, April 2011. doi:10.1109/FOCS.2011.95.
- 14 Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32, 2019.
- 15 Ioana O. Bercea, Samir Khuller, Clemens Rösner, Melanie Schmidt, Martin Groß, Aounon Kumar, and Daniel R. Schmidt. On the cost of essentially fair clusterings. In Dimitris Achlioptas and Laszlo A. Vegh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*, Leibniz International Proceedings in Informatics, LIPIcs. Schloss Dagstuhl- Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, September 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.18.
- 16 Simon Caton and Christian Haas. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020.
- 17 M. Charikar and A. Wirth. Maximizing quadratic programs: extending grothendieck's inequality. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60, 2004. doi:10.1109/FOCS.2004.39.
- 18 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005. Learning Theory 2003. doi:10.1016/j.jcss.2004.10.012.

- 19 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal lp rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 219–228, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746604.
- 20 Anshuman Chhabra, Karina Masalkovaitė, and Prasant Mohapatra. An overview of fairness in clustering. *IEEE Access*, 9:130698–130720, 2021. doi:10.1109/ACCESS.2021.3114099.
- 21 Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5036–5044, Red Hook, NY, USA, 2017. Curran Associates Inc.
- 22 Charles J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984. doi:10.1016/0166-218X(84)90075-1.
- 23 Tom Coleman, James Saunderson, and Anthony Wirth. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*, ESA '08, pages 308–319, 2008.
- 24 Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2):172–187, 2006. Approximation and Online Algorithms. doi:10.1016/j.tcs.2006.05.008.
- 25 Dotan Emanuel and Amos Fiat. Correlation clustering – minimizing disagreements on arbitrary weighted graphs. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, pages 208–220, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- 26 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., USA, 1990.
- 27 Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 438–448, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3442188.3445906.
- 28 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2, May 2005. doi:10.1145/1109557.1109686.
- 29 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995. doi:10.1145/227683.227684.
- 30 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for quadratic integer programming with PSD objectives. *CoRR*, abs/1104.4746, 2011. arXiv:1104.4746.
- 31 Eran Halperin and Uri Zwick. A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 210–225, Berlin, Heidelberg, 2001. Springer-Verlag.
- 32 Qiaoming Han, Yinyu Ye, and Jiawei Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. *Mathematical Programming*, 92:509–535, 2002.
- 33 Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning*, pages 3458–3467. PMLR, 2019.
- 34 Jean B. Lasserre. An explicit exact sdp relaxation for nonlinear 0-1 programs. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, pages 293–303, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- 35 Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, August 2003.

- 36 Bernt Øksendal. *Stochastic Differential Equations*. Springer Berlin Heidelberg, 2003. doi: 10.1007/978-3-642-14394-6.
- 37 Prasad Raghavendra and Ning Tan. Approximating cps with global cardinality constraints using sdp hierarchies. In *SODA*, 2012.
- 38 Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means. In *Approximation and Online Algorithms: 17th International Workshop, WAOA 2019, Munich, Germany, September 12–13, 2019, Revised Selected Papers*, pages 232–251, Berlin, Heidelberg, 2019. Springer-Verlag. doi:10.1007/978-3-030-39479-0\_16.
- 39 Chaitanya Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '04*, pages 526–527, USA, 2004. Society for Industrial and Applied Mathematics.
- 40 Jurgen Van Gael and Xiaojin Zhu. Correlation clustering for crosslingual link detection. In *IJCAI*, pages 1744–1749, 2007.
- 41 Anthony Wirth. Correlation clustering. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 227–231. Springer US, Boston, MA, 2010. doi: 10.1007/978-0-387-30164-8\_176.
- 42 Chenchen Wu, Donglei Du, and Dachuan Xu. An improved semidefinite programming hierarchies rounding approximation algorithm for maximum graph bisection problems. In Ding-Zhu Du and Guochuan Zhang, editors, *Computing and Combinatorics*, pages 304–315, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 43 Sepehr Abbasi Zadeh, Nikhil Bansal, Guru Guruganesh, Aleksandar Nikolov, Roy Schwartz, and Mohit Singh. Sticky brownian rounding and its applications to constraint satisfaction problems. *ArXiv*, abs/1812.07769, 2020.

## A Proof of Theorem 7

**Proof.** The algorithm chooses the best from the following two solutions: a single cluster containing all nodes and a solution with all clusters of size two that correspond to  $M^-$  from Observation 6 (we note  $M^-$  can be computed efficiently by finding a minimum cost perfect matching in a bipartite graph). The former solution has value of at least  $w(E^+)$ , whereas the latter solution has value of at least  $w(E^-) - w(M^- \cap E^-)$ . Following observation 6,  $OPT \leq w(E^+) + w(E^-) - w(M^- \cap E^-)$ . If  $w(E^+) > w(E^-) - w(M^- \cap E^-)$  we note that the single cluster solution has value of at least  $\frac{1}{2} \cdot OPT$ . Otherwise the solution with all clusters of size two, that correspond to  $M^-$ , has value of at least  $\frac{1}{2} \cdot OPT$ . Hence, the above algorithm achieves an approximation of  $\frac{1}{2}$ . ◀

## B Proof of Lemma 8

**Proof.** Given a clustering  $\mathcal{C} = \{C_1, \dots, C_l\}$  we can construct a clustering that has only two clusters  $\mathcal{S} = \{S, \bar{S}\}$  as follows. For every  $C_i$ , with a uniform probability (and independently over the clusters) we place all the nodes of  $C_i$  either in  $S$  or in  $\bar{S}$ . Note that all  $+$  edges that are in agreement in  $\mathcal{C}$  always remain in agreement in  $\mathcal{S}$ , thus  $v^+(\mathcal{S}) \geq v^+(\mathcal{C})$ . Moreover, the probability of every  $-$  edge that is in agreement in  $\mathcal{C}$  to still be in agreement in  $\mathcal{S}$  is exactly  $\frac{1}{2}$ . Therefore,  $\mathbb{E}[v^-(\mathcal{S})] \geq \frac{1}{2}v^-(\mathcal{C})$ . Hence, we can conclude that  $\mathbb{E}[v(\mathcal{S})] \geq v^+(\mathcal{C}) + \frac{1}{2}v^-(\mathcal{C})$  (so there exists a cluster  $\mathcal{S}$  with a value of at least  $v^+(\mathcal{C}) + \frac{1}{2}v^-(\mathcal{C})$ ). This finishes the proof. ◀

## C

 Proof of Lemma 9

**Proof.** We are given an instance of **Fair-Max-Agreement** with two colors and a ratio of 1 : 1. Let  $\mathcal{C}^* = \{C_1^*, \dots, C_l^*\}$  be an optimal clustering for this instance. Following observation 6 we can output the clustering  $M^-$  induces, and we note that the value of this clustering is at least  $v^-(\mathcal{C}^*)$ . Following Lemma 8 when applied for  $\mathcal{C}^*$  the given  $\alpha$ -approximation algorithm can be used to obtain a clustering  $\mathcal{S} = \{S, \bar{S}\}$  with value  $v(\mathcal{S}) \geq \alpha \cdot (v^+(\mathcal{C}^*) + \frac{1}{2}v^-(\mathcal{C}^*))$ . Therefore, choosing the best of the above two clusterings, we can output a solution whose value is at least  $\max\{\alpha \cdot (v^+(\mathcal{C}^*) + \frac{1}{2}v^-(\mathcal{C}^*)), v^-(\mathcal{C}^*)\}$  (we denote this value by  $y$ ). Now we show that for  $0 < \alpha < 1$  it holds that  $y \geq \frac{2\alpha}{2+\alpha} \cdot v(\mathcal{C}^*)$ .

The first case is when  $y = v^-(\mathcal{C}^*)$  and the second case is when  $y = \alpha \cdot v^+(\mathcal{C}^*) + \frac{1}{2}\alpha \cdot v^-(\mathcal{C}^*)$ . Let us focus on the first case, and note that assuming  $y = v^-(\mathcal{C}^*)$ , the definition of  $y$  implies  $v^+(\mathcal{C}^*) \leq (1/\alpha - 1/2)v^-(\mathcal{C}^*)$ . This in turn implies that:

$$v(\mathcal{C}^*) = v^+(\mathcal{C}^*) + v^-(\mathcal{C}^*) \leq v^-(\mathcal{C}^*) (1 + (1/\alpha - 1/2)) = (2+\alpha)/(2\alpha) \cdot y.$$

This concludes the proof for the first case. Let us now focus on the second case, and note that assuming  $y = \alpha \cdot v^+(\mathcal{C}^*) + \frac{1}{2}\alpha \cdot v^-(\mathcal{C}^*)$ , the definition of  $y$  implies  $v^-(\mathcal{C}^*) \leq (2\alpha)/(2-\alpha) \cdot v^+(\mathcal{C}^*)$ . This in turn implies that:

$$\begin{aligned} v(\mathcal{C}^*) &= v^+(\mathcal{C}^*) + v^-(\mathcal{C}^*) = v^+(\mathcal{C}^*) + (2+\alpha)/4 \cdot v^-(\mathcal{C}^*) + (2-\alpha)/4 \cdot v^-(\mathcal{C}^*) \\ &\leq v^+(\mathcal{C}^*) + (2+\alpha)/4 \cdot v^-(\mathcal{C}^*) + (2-\alpha)/4 \cdot (2\alpha)/(2-\alpha) \cdot v^+(\mathcal{C}^*) \\ &= \frac{2+\alpha}{2\alpha} \left( \alpha \cdot v^+(\mathcal{C}^*) + \frac{\alpha}{2} \cdot v^-(\mathcal{C}^*) \right) = \frac{2+\alpha}{2\alpha} \cdot y. \end{aligned}$$

This concludes the proof for the second case. ◀

## D

 Approximating Max-Agreement-Bisection

We claim that one can use the algorithm of Wu, Du and Xu [42], who built upon the work of Austrin, Benabbas and Georgiou [10] for **Max-Bisection**, to obtain a good approximation for **Max-Agreement-Bisection**. The algorithms of [10, 42] both perform the following three phases ([10] for the **Max-Bisection** problem and [42] for the general graph bisection problem). In the first phase the following  $\ell$ -round Lasserre SDP relaxation is solved:

$$\begin{aligned} \max \quad & \sum_{e=(i,j) \in E^+} w(e)(1/2 + 1/2\langle \mathbf{v}_i, \mathbf{v}_j \rangle) + \sum_{e=(i,j) \in E^-} w(e)(1/2 - 1/2\langle \mathbf{v}_i, \mathbf{v}_j \rangle) \\ \text{s.t.} \quad & \langle \mathbf{v}_\emptyset, \sum_{i \in V} \mathbf{v}_{S \triangle \{i\}} \rangle = 0 & \forall S \subseteq V, |S| < \ell \\ & \langle \mathbf{v}_{S_1}, \mathbf{v}_{S_2} \rangle = \langle \mathbf{v}_{S_3}, \mathbf{v}_{S_4} \rangle & \forall S_1, S_2, S_3, S_4 \subseteq V, \\ & & |S_1|, |S_2|, |S_3|, |S_4| \leq \ell, \\ & & S_1 \triangle S_2 = S_3 \triangle S_4 \\ & \langle \mathbf{v}_\emptyset, \mathbf{v}_\emptyset \rangle = 1 \end{aligned}$$

Let us denote by  $\{\mathbf{v}_S^*\}_{S \subseteq V, |S| < \ell}$  an optimal solution to the above relaxation. The following theorem shows how one can extract vectors  $\{\mathbf{v}_i\}_{i=0}^n$ , from  $\{\mathbf{v}_S^*\}_{S \subseteq V, |S| < \ell}$ , such that the value of the objective does not deteriorate much and the vectors  $\{\mathbf{v}_i\}_{i=0}^n$  have low correlation. Before formally stating the above, we introduce the following notation:

$$SDPVal(\{\mathbf{v}_i\}) \triangleq \sum_{e=(i,j) \in E^+} w(e)(1/2 + 1/2\langle \mathbf{v}_i, \mathbf{v}_j \rangle) + \sum_{e=(i,j) \in E^-} w(e)(1/2 - 1/2\langle \mathbf{v}_i, \mathbf{v}_j \rangle).$$

When reading the following theorem, the reader should recall the definitions of  $\mu_i$ ,  $\rho_{i,j}$ , and  $\mathbf{w}_i$ , given in Section 1.3.

► **Theorem 21** (Theorem 3.1 in [10], Theorem 2 in [42]). *There is an algorithm which given a graph  $G = (V, E)$  and  $t \in \mathbb{N}^{\geq 1}$  outputs a set of vectors  $\{\mathbf{v}_i\}_{i=0}^n$  in time  $n^{O(t)}$  such that:*

1.  $SDPVal(\{\mathbf{v}_i\}) \geq SDPVal(\{\mathbf{v}_i^*\}) - 10t^{-\frac{1}{2}}$ .
2.  $\sum_{i=1}^n \langle \mathbf{v}_0, \mathbf{v}_i \rangle = 0$ .
3. *The following triangle inequalities are satisfied for every  $1 \leq i, j \leq n$ :*

$$\mu_i + \mu_j + \rho_{ij} \geq -1, \mu_i - \mu_j - \rho_{ij} \geq -1$$

$$-\mu_i + \mu_j - \rho_{ij} \geq -1, -\mu_i - \mu_j + \rho_{ij} \geq -1$$

4.  $\mathbb{E}_{i,j \in V} [|\langle \mathbf{w}_i, \mathbf{w}_j \rangle|] \leq t^{-\frac{1}{4}}$ .

We note that the above theorem was proved in [10] for the objective of **Max-Bisection** and in [42] for the objective of **Max- $\frac{n}{2}$ -Uncut** (both heavily rely on Raghavendra and Tan [37]). However, one can note that the same proof holds for our definition of  $SDPVal(\{\mathbf{v}_i\})$  for **Max-Agreement-Bisection**.

In the second phase the rounding algorithm of [10] uses  $\{\mathbf{v}_i\}_{i=0}^n$  to extract a cut  $\tilde{S} = \{\tilde{S}, V \setminus \tilde{S}\}$ . This rounding algorithm has the following properties: (1) the rounding does not depend on the coefficients  $c_0, c_1, c_2, c_3$ ; and (2) the analysis is performed edge-wise, i.e., the ratio of the probability of an edge being satisfied by the rounding algorithm to the contribution of the same edge to the value of the relaxation is lower bounded. We note that this cut might not be a bisection, i.e.,  $|\tilde{S}|$  might not equal  $n/2$ , thus corrections must be made. The following lemma is immediate from Lemma 4 in [42] and Lemma 3.2 in [10], where the former is for the **Max- $\frac{n}{2}$ -Uncut** objective and the latter for the **Max-Bisection** objective.

► **Lemma 22.** *(following Lemma 4 in [42] and Lemma 3.2 in [10])*  
 $\mathbb{E}[v(\tilde{S})] \geq \alpha_0 \cdot SDPVal(\{\mathbf{v}_i\}_{i=0}^n)$ , where  $\alpha_0 \geq 0.8776$ .

The last phase is a size adjusting phase in which a subset of vertices from the larger side of the cut  $\tilde{S}$  is moved to the smaller side of the cut in order to create a bisection. This is performed either by choosing a random subset (as is done in [10]), or equivalently, greedily (as is done [42]). This phase incurs an additive loss of  $o(1)$  in the approximation guarantee. We can choose any of the above two options. The following lemma summarizes the approximation guarantee for **Max-Agreement-Bisection**, its proof follows from Theorem 21 and Lemma 22 similarly to [10, 42].

► **Lemma 23.** *Max-Agreement-Bisection is approximable in polynomial time to within a factor 0.8776.*






# On Sketching Approximations for Symmetric Boolean CSPs

Joanna Boyland ✉

Harvard College, Harvard University, Cambridge, MA, USA

Michael Hwang ✉

Harvard College, Harvard University, Cambridge, MA, USA

Tarun Prasad ✉ 

Harvard College, Harvard University, Cambridge, MA, USA

Noah Singer ✉ 

Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Harvard College, Harvard University, Cambridge, MA, USA

Santhoshini Velusamy ✉ 

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

---

## Abstract

A Boolean maximum constraint satisfaction problem,  $\text{Max-CSP}(f)$ , is specified by a predicate  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ . An  $n$ -variable instance of  $\text{Max-CSP}(f)$  consists of a list of constraints, each of which applies  $f$  to  $k$  distinct literals drawn from the  $n$  variables. For  $k = 2$ , Chou, Golovnev, and Velusamy [8] obtained explicit ratios characterizing the  $\sqrt{n}$ -space *streaming* approximability of every predicate. For  $k \geq 3$ , Chou, Golovnev, Sudan, and Velusamy [7] proved a general dichotomy theorem for  $\sqrt{n}$ -space *sketching* algorithms: For every  $f$ , there exists  $\alpha(f) \in (0, 1]$  such that for every  $\epsilon > 0$ ,  $\text{Max-CSP}(f)$  is  $(\alpha(f) - \epsilon)$ -approximable by an  $O(\log n)$ -space *linear* sketching algorithm, but  $(\alpha(f) + \epsilon)$ -approximation sketching algorithms require  $\Omega(\sqrt{n})$  space.

In this work, we give closed-form expressions for the sketching approximation ratios of multiple families of *symmetric* Boolean functions. Letting  $\alpha'_k = 2^{-(k-1)}(1 - k^{-2})^{(k-1)/2}$ , we show that for odd  $k \geq 3$ ,  $\alpha(k\text{AND}) = \alpha'_k$ , and for even  $k \geq 2$ ,  $\alpha(k\text{AND}) = 2\alpha'_{k+1}$ . Thus, for every  $k$ ,  $k\text{AND}$  can be  $(2 - o(1))2^{-k}$ -approximated by  $O(\log n)$ -space *sketching* algorithms; we contrast this with a lower bound of Chou, Golovnev, Sudan, Velingker, and Velusamy [5] implying that *streaming*  $(2 + \epsilon) \cdot 2^{-k}$ -approximations require  $\Omega(n)$  space! We also resolve the ratio for the “at-least- $(k-1)$ -1’s” function for all even  $k$ ; the “exactly- $\frac{k+1}{2}$ -1’s” function for odd  $k \in \{3, \dots, 51\}$ ; and fifteen other functions. We stress here that for general  $f$ , the dichotomy theorem in [7] only implies that  $\alpha(f)$  can be computed to arbitrary precision in  $\text{PSPACE}$ , and thus closed-form expressions need not have existed *a priori*. Our analyses involve identifying and exploiting structural “saddle-point” properties of this dichotomy.

Separately, for all *threshold* functions, we give optimal “bias-based” approximation algorithms generalizing [8] while simplifying [7]. Finally, we investigate the  $\sqrt{n}$ -space *streaming* lower bounds in [7], and show that they are *incomplete* for 3AND, i.e., they fail to rule out  $(\alpha(3\text{AND}) - \epsilon)$ -approximations in  $o(\sqrt{n})$  space.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Theory of computation  $\rightarrow$  Discrete optimization

**Keywords and phrases** Streaming algorithms, constraint satisfaction problems, approximability

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.38

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2112.06319> [3]

*Included in fourth author’s thesis*: <https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37371750> [21]



© Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 38; pp. 38:1–38:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Supplementary Material** *Interactive Resource* (*Interactive Mathematica notebook*):  
<https://notebookarchive.org/2022-03-a5vpzhg/>

**Funding** *Noah Singer*: Supported by the Harvard College Research Program, and by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

*Santhoshini Velusamy*: Supported in part by a Google Ph.D. Fellowship, a Simons Investigator Award to Madhu Sudan, and NSF Awards CCF 1715187 and CCF 2152413.

**Acknowledgements** All authors of this paper are or were supervised by Madhu Sudan. We would like to thank him for advice and guidance on both technical and organizational aspects of this project, as well as for many helpful comments on earlier drafts of this paper.

## 1 Introduction

In this work, we consider the *streaming approximability* of various *Boolean constraint satisfaction problems*, and we begin by defining these terms. See [7, §1.1-2] for more details on the definitions.

### 1.1 Setup: The streaming approximability of Boolean CSPs

#### 1.1.1 Boolean CSPs

Let  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  be a Boolean function. In an  $n$ -variable instance of the problem  $\text{Max-CSP}(f)$ , a *constraint* is a pair  $C = (\mathbf{b}, \mathbf{j})$ , where  $\mathbf{j} = (j_1, \dots, j_k) \in [n]^k$  is a  $k$ -tuple of distinct *indices*, and  $\mathbf{b} = (b_1, \dots, b_k) \in \{-1, 1\}^k$  is a *negation pattern*.

For Boolean vectors  $\mathbf{a} = (a_1, \dots, a_n), \mathbf{b} = (b_1, \dots, b_n) \in \{-1, 1\}^n$ , let  $\mathbf{a} \odot \mathbf{b}$  denote their *coordinate-wise product*  $(a_1 b_1, \dots, a_n b_n)$ . An *assignment*  $\sigma = (\sigma_1, \dots, \sigma_n) \in \{-1, 1\}^n$  satisfies  $C$  iff  $f(\mathbf{b} \odot \sigma|_{\mathbf{j}}) = 1$ , where  $\sigma|_{\mathbf{j}}$  is the  $k$ -tuple  $(\sigma_{j_1}, \dots, \sigma_{j_k})$  (i.e.,  $\sigma$  satisfies  $C$  iff  $f(b_1 \sigma_{j_1}, \dots, b_k \sigma_{j_k}) = 1$ ). An *instance*  $\Psi$  of  $\text{Max-CSP}(f)$  consists of constraints  $C_1, \dots, C_m$  with non-negative weights  $w_1, \dots, w_m$  where  $C_i = (\mathbf{j}(i), \mathbf{b}(i))$  and  $w_i \in \mathbb{R}$  for each  $i \in [m]$ ; the *value*  $\text{val}_{\Psi}(\sigma)$  of an assignment  $\sigma$  to  $\Psi$  is the (weighted) fraction of constraints in  $\Psi$  satisfied by  $\sigma$ , i.e.,  $\text{val}_{\Psi}(\sigma) \stackrel{\text{def}}{=} \frac{1}{W} \sum_{i \in [m]} w_i \cdot f(\mathbf{b}(i) \odot \sigma|_{\mathbf{j}(i)})$ , where  $W = \sum_{i=1}^m w_i$ . The *value*  $\text{val}_{\Psi}$  of an instance  $\Psi$  is the maximum value of any assignment  $\sigma \in \{-1, 1\}^n$ , i.e.,  $\text{val}_{\Psi} \stackrel{\text{def}}{=} \max_{\sigma \in \{-1, 1\}^n} \text{val}_{\Psi}(\sigma)$ .

#### 1.1.2 Approximations to CSPs

For  $\alpha \in [0, 1]$ , we consider the problem of  $\alpha$ -*approximating*  $\text{Max-CSP}(f)$ . In this problem, the goal of an algorithm  $\mathcal{A}$  is to, on input an instance  $\Psi$ , output an estimate  $\mathcal{A}(\Psi)$  such that with probability at least  $\frac{2}{3}$ ,  $\alpha \cdot \text{val}_{\Psi} \leq \mathcal{A}(\Psi) \leq \text{val}_{\Psi}$ . For  $\beta < \gamma \in [0, 1]$ , we also consider the closely related  $(\beta, \gamma)$ - $\text{Max-CSP}(f)$ . In this problem, the input instance  $\Psi$  is promised to either satisfy  $\text{val}_{\Psi} \leq \beta$  or  $\text{val}_{\Psi} \geq \gamma$ , and the goal is to decide which is the case with probability at least  $\frac{2}{3}$ .

#### 1.1.3 Streaming and sketching algorithms for CSPs

For various Boolean functions  $f$ , we consider algorithms which attempt to approximate  $\text{Max-CSP}(f)$  instances in the (*single-pass, insertion-only*) *space- $s$  streaming setting*. Such algorithms can only use space  $s$  (which is ideally small, such as  $O(\log n)$ , where  $n$  is the number of variables in an input instance), and, when given as input a CSP instance  $\Psi$ , can only read the list of constraints in a single, left-to-right pass.

We also consider a (seemingly) weak class of streaming algorithms called *sketching algorithms*, where the algorithm's output is determined by an length- $s$  string called a “sketch” produced from the input stream, and the sketch itself has the property that the sketch of the concatenation of two streams can be computed from the sketches of the two component streams. (See [7, §3.3] for a formal definition.) A special case of sketching algorithms are *linear sketches*, where each sketch (i.e., element of  $\{0, 1\}^s$ ) encodes an element of a vector space and we perform vector addition to combine two sketches.

## 1.2 Prior work and motivations

### 1.2.1 Prior results on streaming and sketching Max-CSP( $f$ )

We first give a brief review of what is already known about the streaming and sketching approximability of Max-CSP( $f$ ). For  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ , let  $\rho(f) \stackrel{\text{def}}{=} \Pr_{\mathbf{b} \sim \text{Unif}(\{-1, 1\}^k)}[f(\mathbf{b}) = 1]$ , where  $\text{Unif}(\{-1, 1\}^k)$  denotes the uniform distribution on  $\{-1, 1\}^k$ . For every  $f$ , the Max-CSP( $f$ ) problem has a trivial  $\rho(f)$ -approximation algorithm given by simply outputting  $\rho(f)$  since  $\mathbb{E}_{\mathbf{a} \sim \text{Unif}(\{-1, 1\}^n)}[\text{val}_\Psi(\mathbf{a})] = \Pr_{\mathbf{b} \sim \text{Unif}(\{-1, 1\}^k)}[f(\mathbf{b}) = 1] = \rho(f)$ . We refer to a function  $f$  as *approximation-resistant* for some class of algorithms (e.g., streaming or sketching algorithms with some space bound) if it cannot be  $(\rho(f) + \epsilon)$ -approximated for any constant  $\epsilon > 0$ . Otherwise, we refer to  $f$  as *approximable* for the class of algorithms.

The first two CSPs whose  $o(\sqrt{n})$ -space streaming approximabilities were resolved were Max-2XOR and Max-2AND. Kapralov, Khanna, and Sudan [18] showed that Max-2XOR is approximation-resistant to  $o(\sqrt{n})$ -space streaming algorithms. Later, Chou, Golovnev, and Velusamy [8], building on earlier work of Guruswami, Velusamy, and Velingker [12], gave an  $O(\log n)$ -space linear sketching algorithm which  $(\frac{4}{9} - \epsilon)$ -approximates Max-2AND for every  $\epsilon > 0$  and showed that  $(\frac{4}{9} + \epsilon)$ -approximations require  $\Omega(\sqrt{n})$  space, even for streaming algorithms.

In two recent works [7, 6], Chou, Golovnev, Sudan, and Velusamy proved so-called *dichotomy theorems* for sketching CSPs. In [7], they prove the dichotomy for CSPs over the Boolean alphabet with negations of variables (i.e., the setup we described in Section 1.1.1). In [6], they extend it to the more general case of CSPs over finite alphabets.<sup>1</sup> See [6, §1] and [21] for more general background on CSPs in the streaming setting.

[7] is most relevant for our purposes, as it concerns Boolean CSPs. For a fixed constraint function  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$ , the main result in [7] is the following *dichotomy theorem*: For any  $0 \leq \gamma < \beta \leq 1$ , either

1.  $(\beta, \gamma)$ -Max-CSP( $f$ ) has an  $O(\log n)$ -space linear sketching algorithm, *or*
2. For all  $\epsilon > 0$ , sketching algorithms for  $(\beta + \epsilon, \gamma - \epsilon)$ -Max-CSP( $f$ ) require  $\Omega(\sqrt{n})$  space.

Distinguishing whether (1) or (2) applies is equivalent to deciding whether two convex polytopes (which depend on  $f, \gamma, \beta$ ) intersect. We omit a technical statement of this criterion, and instead focus on the following corollary: there exists an  $\alpha(f) \in [0, 1]$  such that Max-CSP( $f$ ) can be  $(\alpha(f) - \epsilon)$ -approximated by  $O(\log n)$ -space linear sketches, but not  $(\alpha(f) + \epsilon)$ -approximated by  $o(\sqrt{n})$ -space sketches, for all  $\epsilon > 0$ ; furthermore,  $\alpha(f)$  equals the solution to an explicit minimization problem, which we describe in Section 2.1 (in the special case where  $f$  is symmetric).

<sup>1</sup> More precisely, [7] and [6] both consider the more general case of CSPs defined by *families* of functions of a specific arity. We do not need this generality for the purposes of our paper, and therefore omit it.

*A priori*, it may be possible to achieve an  $(\alpha(f) + \epsilon)$ -approximation with a  $o(\sqrt{n})$ -space *streaming* algorithm. But [7] also extends the lower bound (case 2 of the dichotomy) to cover streaming algorithms when special objects called *padded one-wise pairs* exist. See Section 2.4 below for a definition (again, specialized for symmetric functions). The padded one-wise pair criterion is sufficient to recover all previous streaming approximability results for Boolean functions (i.e., [18, 8]), and prove several new ones. In particular, [7] proves that if  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  has the property that there exists  $\mathcal{D} \in \Delta(f^{-1}(1))$  such that  $\mathbb{E}_{\mathbf{b} \sim \mathcal{D}}[b_i] = 0$  for all  $i \in [k]$  (where  $[k] \stackrel{\text{def}}{=} \{1, \dots, k\}$ ), then  $\text{Max-CSP}(f)$  is streaming approximation-resistant. For symmetric Boolean CSPs, they also prove the converse, and thus give a complete characterization for approximation resistance [7, Lemma 2.14]. However, besides  $\text{Max-2AND}$ , [7] does not explicitly analyze the approximation ratio of any CSP that is “approximable”, i.e., not approximation resistant.

### 1.2.2 Questions from previous work

In this work, we address several major questions about streaming approximations for Boolean CSPs which Chou, Golovnev, Sudan, and Velusamy [7] leave unanswered:

1. Can the framework in [7] be used to find closed-form sketching approximability ratios  $\alpha(f)$  for approximable problems  $\text{Max-CSP}(f)$  beyond  $\text{Max-2AND}$ ?
2. As observed in [5, §1.3], [7] implies the following “trivial upper bound” on streaming approximability: for all  $f$ ,  $\alpha(f) \leq 2\rho(f)$ . How tight is this upper bound?
3. Does the streaming lower bound (the “padded one-wise pair” criterion) in [7] suffice to resolve the streaming approximability of every function?
4. The optimal  $(\alpha(f) - \epsilon)$ -approximation algorithm for  $\text{Max-CSP}(f)$  in [7] requires running a “grid” of  $O(1/\epsilon^2)$  distinguishers for  $(\beta, \gamma)$ - $\text{Max-CSP}(f)$  distinguishing problems in parallel. Can we obtain simpler optimal sketching approximations?

### 1.3 Our results

We study the questions in Section 1.2.2 for *symmetric* Boolean CSPs. Symmetric Boolean functions are those functions that depend only on the Hamming weight of the input, i.e., number of 1’s in the input.<sup>2</sup> For a set  $S \subseteq [k]$ , we define  $f_{S,k} : \{-1, 1\}^k \rightarrow \{0, 1\}$  as the indicator function for the set  $\{\mathbf{b} \in \{-1, 1\}^k : \text{wt}(\mathbf{b}) \in S\}$  (where  $\text{wt}(\mathbf{b})$  denotes the Hamming weight of  $\mathbf{b}$ ). That is,  $f_{S,k}(\mathbf{x}) = 1$  if and only if  $\text{wt}(\mathbf{x}) \in S$ . Some well-studied examples of functions in this class include  $k\text{AND} = f_{\{k\},k}$ , the *threshold functions*  $\text{Th}_k^i = f_{\{i, i+1, \dots, k\},k}$ , and “exact weight” functions  $\text{Ex}_k^i = f_{\{i\},k}$ .<sup>3</sup>

<sup>2</sup> Note that the inputs are in  $\{-1, 1\}^k$ ; we define the Hamming weight as the number of 1’s, and not  $-1$ ’s (which is arguably more “natural” under the mapping  $b \in \{0, 1\} \mapsto (-1)^b \in \{-1, 1\}$ ), for consistency with [7].

<sup>3</sup> By [7, Lemma 2.14], if  $S$  contains elements  $s \leq \frac{k}{2}$  and  $t \geq \frac{k}{2}$ , not necessarily distinct, then  $f_{S,k}$  supports one-wise independence and is therefore approximation-resistant (even to streaming algorithms). Thus, we focus on the case where all elements of  $S$  are either larger than or smaller than  $\frac{k}{2}$ . Moreover, note that if  $S' = \{k - s : s \in S\}$ , every instance of  $\text{Max-CSP}(f_{S,k})$  can be viewed as an instance of  $\text{Max-CSP}(f_{S',k})$  with the same value, since for any constraint  $C = (\mathbf{b}, \mathbf{j})$  and assignment  $\sigma \in \{-1, 1\}^n$ , we have  $f_{S,k}(\mathbf{b} \odot \sigma|_{\mathbf{j}}) = f_{S',k}(\mathbf{b} \odot (-\sigma)|_{\mathbf{j}})$ . Thus, we further narrow our focus to the case where every element of  $S$  is larger than  $\frac{k}{2}$ .

### 1.3.1 The sketching approximability of Max- $k$ AND

Chou, Golovnev, and Velusamy [8] showed that  $\alpha(2\text{AND}) = \frac{4}{9}$  (and  $(\frac{4}{9} + \epsilon)$ -approximation can be ruled out even for  $o(\sqrt{n})$ -space streaming algorithms). For  $k \geq 3$ , while Chou, Golovnev, Velusamy, and Sudan [7] give optimal sketching approximation algorithms for Max- $k$ AND, they do not explicitly analyze the approximation ratio  $\alpha(k\text{AND})$ , and show only that it lies between  $2^{-k}$  and  $2^{-(k-1)}$ .

In this paper, we analyze the dichotomy theorem in [7], and obtain a closed-form expression for the sketching approximability of Max- $k$ AND for every  $k$ . For odd  $k \geq 3$ , define the constant

$$\alpha'_k \stackrel{\text{def}}{=} \left( \frac{(k-1)(k+1)}{4k^2} \right)^{(k-1)/2} = 2^{-(k-1)} \cdot \left( 1 - \frac{1}{k^2} \right)^{(k-1)/2}. \quad (1)$$

In Section 4, we prove the following:

► **Theorem 1.** *For odd  $k \geq 3$ ,  $\alpha(k\text{AND}) = \alpha'_k$ , and for even  $k \geq 2$ ,  $\alpha(k\text{AND}) = 2\alpha'_{k+1}$ .*

Since  $\rho(k\text{AND}) = 2^{-k}$ , Theorem 1 also has the following important corollary:

► **Corollary 2.**  $\lim_{k \rightarrow \infty} \frac{\alpha(k\text{AND})}{2\rho(k\text{AND})} = 1$ .

Recall that [7] implies that  $\alpha(f) \leq 2\rho(f)$  for all functions  $f$ . Indeed, Chou, Golovnev, Sudan, Velusamy, and Velingker [5] show that any function  $f$  cannot be  $(2\rho(f) + \epsilon)$ -approximated even by  $o(n)$ -space streaming algorithms. On the other hand, in Section 1.3.3 below, we describe simple  $O(\log n)$ -space sketching algorithms for Max- $k$ AND achieving the optimal ratio from [7]. Thus, as  $k \rightarrow \infty$ , these algorithms achieve an asymptotically optimal approximation ratio even among  $o(n)$ -space streaming algorithms!

### 1.3.2 The sketching approximability of other symmetric functions

We also analyze the sketching approximability of a number of other symmetric Boolean functions. Specifically, for the threshold functions  $\text{Th}_k^{k-1}$  for even  $k$ , we show that:

► **Theorem 3.** *For even  $k \geq 2$ ,  $\alpha(\text{Th}_k^{k-1}) = \frac{k}{2} \alpha'_{k-1}$ .*

We prove Theorem 3 in Section 5.1 using techniques similar to our proof of Theorem 1. We also provide partial results for  $\text{Ex}_k^{(k+1)/2}$ , including closed forms for small  $k$  and an asymptotic analysis of  $\alpha(\text{Ex}_k^{(k+1)/2})$ :

► **Theorem 4** (Informal version of Theorem 25). *For odd  $k \in \{3, \dots, 51\}$ , there is an explicit expression for  $\alpha(\text{Ex}_k^{(k+1)/2})$  as a function of  $k$ .*

► **Theorem 5.**  $\lim_{\text{odd } k \rightarrow \infty} \frac{\alpha(\text{Ex}_k^{(k+1)/2})}{\rho(\text{Ex}_k^{(k+1)/2})} = 1$ .

We prove Theorems 4 and 5 in Section 5.2. Finally, in Section 5.3, we explicitly resolve fifteen other cases (e.g.,  $f_{\{2,3\},3}$  and  $f_{\{4\},5}$ ) not covered by Theorems 1, 3, and 4.

### 1.3.3 Simple approximation algorithms for threshold functions

Chou, Golovnev, and Velusamy's optimal  $(\frac{4}{9} - \epsilon)$ -approximation for 2AND [8], like Guruswami, Velingker, and Velusamy's earlier  $(\frac{2}{5} - \epsilon)$ -approximation [12], is based on measuring a quantity called the *bias* of an instance  $\Psi$ , denoted  $\text{bias}(\Psi)$ , which is defined as follows: For each  $i \in [n]$ ,

$\text{diff}_i(\Psi)$  is the difference in total weight between constraints where  $x_i$  occurs positively and negatively, and  $\text{bias}(\Psi) \stackrel{\text{def}}{=} \frac{1}{km} \sum_{i=1}^n |\text{diff}_i(\Psi)| \in [0, 1]$ .<sup>4</sup> In the sketching setting,  $\text{bias}(\Psi)$  can be estimated using standard  $\ell_1$ -norm sketching algorithms [16, 17].

In Section 7, we give simple optimal bias-based approximation algorithms for threshold functions:

► **Theorem 6.** *Let  $f_{S,k} = \text{Th}_k^i$  be a threshold function. Then for every  $\epsilon > 0$ , there exists a piecewise linear function  $\gamma : [-1, 1] \rightarrow [0, 1]$  and a constant  $\epsilon' > 0$  such that the following is a sketching  $(\alpha(f_{S,k}) - \epsilon)$ -approximation for  $\text{Max-CSP}(f_{S,k})$ : On input  $\Psi$ , compute an estimate  $\hat{b}$  for  $\text{bias}(\Psi)$  up to a multiplicative  $(1 \pm \epsilon')$  error and output  $\gamma(\hat{b})$ .*

Our construction generalizes the algorithm in [8] for 2AND to all threshold functions, and is also a simplification, since the [8] algorithm computes a more complicated function of  $\hat{b}$ .

For all CSPs whose approximability we resolve in this paper, we apply an analytical technique which we term the “max-min method;” see the discussion in Section 2.3 below. For such CSPs, our algorithm can be extended to solve the problem of outputting an approximately optimal *assignment* (instead of just the value of such an assignment). Indeed, for this problem, we give a simple randomized streaming algorithm using  $O(n)$  space and time:

► **Theorem 7** (Informal version of Theorem 34). *Let  $f_{S,k}$  be a function for which the max-min method applies, such as  $k\text{AND}$ , or  $\text{Th}_k^{k-1}$  (for even  $k$ ). Then there exists a constant  $p^* \in [0, 1]$  such that following algorithm, on input  $\Psi$ , outputs an assignment with expected value at least  $\alpha(f_{S,k})\text{val}_\Psi$ : Assign variable  $i$  to 1 if  $\text{diff}_i(\Psi) \geq 0$  and  $-1$  otherwise, and then flip each variable’s assignment independently with probability  $p^*$ .*

Our algorithm can potentially be derandomized using universal hash families, as in Biswas and Raman’s recent derandomization [1] of the Max-2AND algorithm in [8].

### 1.3.4 Sketching vs. streaming approximability

Theorem 1 implies that  $\alpha(3\text{AND}) = \frac{2}{9}$ . We prove that the padded one-wise pair criterion of Chou, Golovnev, Sudan, and Velusamy [7] is not sufficient to completely resolve the *streaming* approximability of Max-3AND:

► **Theorem 8** (Informal version of Theorem 12 + Observation 13). *The padded one-wise pair criterion in [7] does not rule out a  $o(\sqrt{n})$ -space streaming  $(\frac{2}{9} + \epsilon)$ -approximation for 3AND for every  $\epsilon > 0$ ; however, it does rule out such an algorithm for  $\epsilon \gtrsim 0.0141$ .*

We state these results formally in Section 2.4 and prove them in Section 6. Separately, Theorem 3 implies that  $\alpha(\text{Th}_4^3) = \frac{4}{9}$ , and the padded one-wise pair criterion *can* be used to show that  $(\frac{4}{9} + \epsilon)$ -approximating  $\text{Max-CSP}(\text{Th}_4^3)$  requires  $\Omega(\sqrt{n})$  space in the streaming setting (see Observation 22 below).

## 1.4 Related work

The classical approximability of Max- $k\text{AND}$  has been the subject of intense study, both in terms of algorithms [11, 10, 26, 23, 25, 13, 14, 4] and hardness-of-approximation [15, 24, 22, 19, 9, 20], given its intimate connections to  $k$ -bit PCPs. Charikar, Makarychev, and

<sup>4</sup> [12, 8] did not normalize by  $\frac{1}{kW}$ .



Makarychev [4] constructed an  $\Omega(k2^{-k})$ -approximation to  $\text{Max-}k\text{AND}$ , while Samorodnitsky and Trevisan [20] showed that  $k2^{-(k-1)}$ -approximations and  $(k+1)2^{-k}$ -approximations are  $\text{NP-}$  and  $\text{UG-hard}$ , respectively.

Interestingly, recalling that  $\alpha(k\text{AND}) \rightarrow 2\rho(k\text{AND}) = 2^{-(k-1)}$  as  $k \rightarrow \infty$ , in the large- $k$  limit our simple randomized algorithm (given in Theorem 7) matches the performance of Trevisan’s [23] parallelizable LP-based algorithm for  $k\text{AND}$ , which (to the best of our knowledge) was the first work on the general  $k\text{AND}$  problem! The subsequent works [13, 14, 4] superseding [23] use more complex techniques involving semidefinite programming, but are structurally similar to our algorithm in Theorem 7: They all involve “guessing” an assignment  $\mathbf{x} \in \mathbb{Z}_2^n$  and then perturbing each bit with constant probability.

## 2 Our techniques

In this section, we give a more detailed background on the technical aspects of the dichotomy theorem in [7], and explain the novel aspects of our analysis.

### 2.1 The Chou, Golovnev, Sudan, and Velusamy [7] framework for symmetric functions

In this section, we describe the Chou, Golovnev, Sudan, and Velusamy [7] framework for finding the optimal sketching approximation ratio of a symmetric Boolean function  $f_{S,k}$ .

Let  $\Delta(\{-1, 1\}^k)$  denote the space of all distributions on  $\{-1, 1\}^k$ . For a distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  and  $\mathbf{x} \in \{-1, 1\}^k$ , we use  $\mathcal{D}(\mathbf{x})$  to denote the probability of sampling  $\mathbf{x}$  in  $\mathcal{D}$ . To a distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  we associate a *canonical instance*  $\Psi_{\mathcal{D}}$  of  $\text{Max-CSP}(f_{S,k})$  on  $k$  variables as follows. Let  $\mathbf{j} = (1, \dots, k)$ . For every negation pattern  $\mathbf{b} \in \{-1, 1\}^k$ ,  $\Psi_{\mathcal{D}}$  contains the constraint  $(\mathbf{b}, \mathbf{j})$  with weight  $\mathcal{D}(\mathbf{b})$ .

We say a distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  is *symmetric* if all vectors of equal Hamming weight are equiprobable, i.e., for every  $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^k$  such that  $\text{wt}(\mathbf{x}) = \text{wt}(\mathbf{y})$ ,  $\mathcal{D}(\mathbf{x}) = \mathcal{D}(\mathbf{y})$ . Let  $\Delta_k \subseteq \Delta(\{-1, 1\}^k)$  denote the set of all symmetric distributions on  $\{-1, 1\}^k$ . Given  $\mathcal{D} \in \Delta_k$ , let  $\mathcal{D}\langle i \rangle \stackrel{\text{def}}{=} \sum_{\mathbf{x} \in \{-1, 1\}^k: \text{wt}(\mathbf{x})=i} \mathcal{D}(\mathbf{x})$  denote the total probability mass on vectors of Hamming weight  $i$ . Note that any vector  $(\mathcal{D}\langle 0 \rangle, \dots, \mathcal{D}\langle k \rangle)$  of nonnegative values summing to 1 uniquely determines a distribution  $\mathcal{D} \in \Delta_k$ ; we write  $\mathcal{D} = (\mathcal{D}\langle 0 \rangle, \dots, \mathcal{D}\langle k \rangle)$  for notational convenience.

Let  $\text{Bern}(p)$  represent a random variable which is 1 with probability  $p$  and  $-1$  with probability  $1 - p$ . For  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  and  $p \in [0, 1]$ , let

$$\lambda_S(\mathcal{D}, p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{a} \sim \mathcal{D}, \mathbf{b} \sim \text{Bern}(p)^k} [f_{S,k}(\mathbf{a} \odot \mathbf{b})] = \mathbb{E}_{\mathbf{b} \sim \text{Bern}(p)^k} [\text{val}_{\Psi_{\mathcal{D}}}(\mathbf{b})] \quad (2)$$

denote the expected value of a “ $p$ -biased symmetric assignment” on  $\mathcal{D}$ ’s canonical instance. Also, for a symmetric distribution  $\mathcal{D} \in \Delta_k$ , we define its (scalar) *marginal*

$$\mu(\mathcal{D}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{b} \sim \mathcal{D}} [b_1] = \dots = \mathbb{E}_{\mathbf{b} \sim \mathcal{D}} [b_k]. \quad (3)$$

In general,  $\lambda_S$  is linear in  $\mathcal{D}$  and degree- $k$  in  $p$ , and  $\mu$  is linear in  $\mathcal{D}$ . For  $\mathcal{D} \in \Delta_k$ , we provide explicit formulas for  $\lambda_S$  and  $\mu$  in Section 3.

Roughly, [7] states that  $\text{Max-CSP}(f_{S,k})$  is hard to approximate in the sketching setting if there exist distributions  $\mathcal{D}_N, \mathcal{D}_Y \in \Delta_k$  such that (1)  $\mu(\mathcal{D}_N) = \mu(\mathcal{D}_Y)$  and (2)  $\mathcal{D}_Y$ ’s canonical instance is highly satisfied by the trivial (all-ones) assignment but (3)  $\mathcal{D}_N$ ’s canonical instance is not well-satisfied by any “biased symmetric assignment”. To be precise, for  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$ , let



$$\beta_S(\mathcal{D}) \stackrel{\text{def}}{=} \sup_{p \in [0,1]} \lambda_S(\mathcal{D}, p) \text{ and } \gamma_S(\mathcal{D}) \stackrel{\text{def}}{=} \lambda_S(\mathcal{D}, 1), \quad (4)$$

and define

$$\alpha(f_{S,k}) \stackrel{\text{def}}{=} \inf_{\mathcal{D}_N, \mathcal{D}_Y \in \Delta_k: \mu(\mathcal{D}_N) = \mu(\mathcal{D}_Y)} \left( \frac{\beta_S(\mathcal{D}_N)}{\gamma_S(\mathcal{D}_Y)} \right). \quad (5)$$

For every symmetric function  $f_{S,k}$ , [7] proves that  $\alpha(f_{S,k})$  is the optimal sketching approximation ratio for Max-CSP( $f_{S,k}$ ):

► **Theorem 9** (Combines [7, Theorem 2.10 and Lemma 2.14]). *Let  $f_{S,k} : \{-1, 1\}^k \rightarrow \{0, 1\}$  be a symmetric function. Then for every  $\epsilon > 0$ , there is an linear sketching  $(\alpha(f_{S,k}) - \epsilon)$ -approximation to Max-CSP( $f_{S,k}$ ) in  $O(\log n)$  space, but any sketching  $(\alpha(f_{S,k}) + \epsilon)$ -approximation to Max-CSP( $f_{S,k}$ ) requires  $\Omega(\sqrt{n})$  space.*

► **Remark.** In the general case where  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  is not symmetric, the approximability of  $f$  is no longer characterized by Equation (5). Instead, [7] requires taking an infimum over *all* (not necessarily symmetric) distributions  $\mathcal{D}_N, \mathcal{D}_Y \in \Delta(\{-1, 1\}^k)$ . Moreover, a general distribution  $\mathcal{D} \in \Delta(\{-1, 1\}^k)$  no longer has a single scalar marginal (as in Equation (3)). Instead, we must consider a *vector* marginal  $\mu(\mathcal{D}) = (\mu_1, \dots, \mu_k)$  with  $i$ -th component  $\mu_i = \mathbb{E}_{\mathbf{b} \sim \mathcal{D}}[b_i]$ ; correspondingly,  $\mathcal{D}_N$  and  $\mathcal{D}_Y$  are required to satisfy the constraint  $\mu(\mathcal{D}_N) = \mu(\mathcal{D}_Y)$ . These issues motivate our focus on *symmetric* functions in this paper. Since we need to consider only symmetric distributions in Equation (5),  $\mathcal{D}_Y$  and  $\mathcal{D}_N$  are each parameterized by  $k + 1$  variables (as opposed to  $2^k$  variables), and there is a single linear equality constraint (as opposed to  $k$  constraints).

## 2.2 Formulations of the optimization problem

In order to show that  $\alpha(2\text{AND}) = \frac{4}{9}$ , Chou, Golovnev, Sudan, and Velusamy [7, Example 1] use the following reformulation of the optimization problem on the right hand side of Equation (5). For a symmetric function  $f_{S,k}$  and  $\mu \in [-1, 1]$ , let

$$\beta_{S,k}(\mu) = \inf_{\mathcal{D}_N \in \Delta_k: \mu(\mathcal{D}_N) = \mu} \beta_S(\mathcal{D}_N) \text{ and } \gamma_{S,k}(\mu) = \sup_{\mathcal{D}_Y \in \Delta_k: \mu(\mathcal{D}_Y) = \mu} \gamma_S(\mathcal{D}_Y); \quad (6)$$

then

$$\alpha(f_{S,k}) = \inf_{\mu \in [-1, 1]} \left( \frac{\beta_{S,k}(\mu)}{\gamma_{S,k}(\mu)} \right). \quad (7)$$

The optimization problem on the right-hand side of Equation (7) appears simpler than that of Equation (5) because it is univariate, but there is a hidden difficulty: Finding an explicit solution requires giving explicit formulas for  $\beta_{S,k}(\mu)$  and  $\gamma_{S,k}(\mu)$ . In the case of  $2\text{AND} = f_{\{2\}, 2}$ , Chou, Golovnev, Sudan, and Velusamy [7] show that  $\gamma_{\{2\}, 2}(\mu)$  is an explicit linear function of  $\mu$ ; maximize the quadratic  $\lambda_{\{2\}}(\mathcal{D}_N, p)$  over  $p \in [0, 1]$  to find  $\beta_{\{2\}}(\mathcal{D}_N)$ ; and then minimize  $\beta_{\{2\}}(\mathcal{D}_N)$  given  $\mu(\mathcal{D}_N) = \mu$  to find  $\beta_{\{2\}, 2}(\mu)$ . However, while for general symmetric functions  $f_{S,k}$  we can describe  $\gamma_{S,k}(\mu)$  as an explicit piecewise linear function of  $\mu$  (see Lemma 16 below), we do not know how to find closed forms for  $\beta_{S,k}(\mu)$  even for  $3\text{AND}$ . Thus, in this work we introduce a different formulation of the optimization problem:

$$\alpha(f_{S,k}) = \inf_{\mathcal{D}_N \in \Delta_k} \left( \frac{\beta_S(\mathcal{D}_N)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right). \quad (8)$$

This reformulation is valid because

$$\alpha(f_{S,k}) = \inf_{\mu \in [-1,1], \mathcal{D}_N \in \Delta_k: \mu(\mathcal{D}_N) = \mu} \left( \frac{\beta_S(\mathcal{D}_N)}{\gamma_{S,k}(\mu)} \right) = \inf_{\mathcal{D}_N \in \Delta_k} \left( \frac{\beta_S(\mathcal{D}_N)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right).$$

We view optimizing directly over  $\mathcal{D}_N \in \Delta_k$  as an important conceptual switch. In particular, our formulation emphasizes the calculation of  $\beta_S(\mathcal{D}_N)$  as the centrally difficult feature, yet we can still take advantage of the relative simplicity of calculating  $\gamma_{S,k}(\mu)$ .

### 2.3 Our contribution: The max-min method

*A priori*, solving the optimization problem on the right-hand side of Equation (8) still requires calculating  $\beta_S(\mathcal{D}_N)$ , which involves maximizing a degree- $k$  polynomial. To get around this difficulty, we have made a key discovery, which was not noticed by Chou, Golovnev, Sudan, and Velusamy [7] even in the 2AND case. Let  $\mathcal{D}_N^*$  minimize the right-hand side of Equation (8), and  $p^*$  maximize  $\lambda_S(\mathcal{D}_N^*, \cdot)$ . After substituting  $\beta_S(\mathcal{D}) = \sup_{p \in [0,1]} \lambda_S(\mathcal{D}, p)$  in Equation (8), and applying the max-min inequality, we get

$$\begin{aligned} \alpha(f_{S,k}) &= \inf_{\mathcal{D}_N \in \Delta_k} \sup_{p \in [0,1]} \left( \frac{\lambda_S(\mathcal{D}_N, p)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right) \geq \sup_{p \in [0,1]} \inf_{\mathcal{D}_N \in \Delta_k} \left( \frac{\lambda_S(\mathcal{D}_N, p)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right) \\ &\geq \inf_{\mathcal{D}_N \in \Delta_k} \left( \frac{\lambda_S(\mathcal{D}_N, p^*)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right). \end{aligned} \quad (9)$$

Given  $p^*$ , the right-hand side of Equation (9) is relatively easy to calculate, being a ratio of a linear and piecewise linear function of  $\mathcal{D}_N$ . Our discovery is that, in a wide variety of cases, the quantity on the right-hand side of Equation (9) *equals*  $\alpha(f_{S,k})$ ; that is,  $(\mathcal{D}_N^*, p^*)$  is a *saddle point* of  $\frac{\lambda_S(\mathcal{D}_N, p)}{\gamma_{S,k}(\mu(\mathcal{D}_N))}$ .<sup>5</sup>

This yields a novel technique, which we call the “max-min method”, for finding a closed form for  $\alpha(f_{S,k})$ . First, we guess  $\mathcal{D}_N^*$  and  $p^*$ , and then, we show analytically that  $\frac{\lambda_S(\mathcal{D}_N, p)}{\gamma_{S,k}(\mu(\mathcal{D}_N))}$  has a saddle point at  $(\mathcal{D}_N^*, p^*)$  and that  $\lambda_S(\mathcal{D}_N, p)$  is maximized at  $p^*$ . These imply that  $\frac{\lambda_S(\mathcal{D}_N^*, p^*)}{\gamma_{S,k}(\mu(\mathcal{D}_N^*))}$  is a lower and upper bound on  $\alpha(f_{S,k})$ , respectively. For instance, in Section 4, in order to give a closed form for  $\alpha(k\text{AND})$  for odd  $k$  (i.e., the odd case of Theorem 1), we guess  $\mathcal{D}_N^* \langle \frac{k+1}{2} \rangle = 1$  and  $p^* = \frac{k+1}{2k}$  (by using Mathematica for small cases), and then check the saddle-point and maximization conditions in two separate lemmas (Lemmas 17 and 18, respectively). Then, we show that  $\alpha(k\text{AND}) = \alpha'_k$  by analyzing the right hand side of the appropriate instantiation of Equation (9). We use similar techniques for  $k\text{AND}$  for even  $k$  (also Theorem 1) and for various other cases in Sections 5.1–5.3.

In all of these cases, the  $\mathcal{D}_N^*$  we construct is supported on at most two distinct Hamming weights, which is the property which makes finding  $\mathcal{D}_N^*$  tractable (using computer assistance). However, this technique is not a “silver bullet”: it is not the case that the sketching approximability of every symmetric Boolean CSP can be exactly calculated by finding the optimal  $\mathcal{D}_N^*$  supported on two elements and using the max-min method. Indeed, (as mentioned in Section 5.3) we verify using computer assistance that this is not the case for  $f_{\{3\},4}$ .

<sup>5</sup> This term comes from the optimization literature; such points are also said to satisfy the “strong max-min property” (see, e.g., [2, pp. 115, 238]). The saddle-point property is guaranteed by von Neumann’s minimax theorem for functions which are concave and convex in the first and second arguments, respectively, but this theorem and the generalizations we are aware of do not apply even to 3AND.

Finally, we remark that the saddle-point property is precisely what defines the value  $p^*$  required for our simple classical algorithm for outputting approximately optimal assignments for  $\text{Max-CSP}(f_{S,k})$  where  $f_{S,k} = \text{Th}_k^i$  is a threshold function (see Theorem 34).

## 2.4 Streaming lower bounds

Chou, Golovnev, Sudan, and Velusamy [7] also define the following condition on pairs  $(\mathcal{D}_N, \mathcal{D}_Y)$ , stronger than  $\mu(\mathcal{D}_N) = \mu(\mathcal{D}_Y)$ , which implies hardness of  $(\gamma, \beta)\text{-Max-CSP}(f)$  for streaming algorithms:

► **Definition 10** (Padded one-wise pairs, [7, §2.3] (symmetric case)). *A pair of distributions  $(\mathcal{D}_Y, \mathcal{D}_N) \in \Delta_k$  forms a padded one-wise pair if there exists  $\tau \in [0, 1]$  and distributions  $\mathcal{D}_0, \mathcal{D}'_Y, \mathcal{D}'_N \in \Delta_k$  such that (1)  $\mu(\mathcal{D}'_Y) = \mu(\mathcal{D}'_N) = 0$  and (2)  $\mathcal{D}_Y = \tau\mathcal{D}_0 + (1 - \tau)\mathcal{D}'_Y$  and  $\mathcal{D}_N = \tau\mathcal{D}_0 + (1 - \tau)\mathcal{D}'_N$ .*

► **Theorem 11** (Streaming lower bound for padded one-wise pairs, [7, Theorem 2.11] (symmetric case)). *Let  $(\mathcal{D}_Y, \mathcal{D}_N)$  be a padded one-wise pair. Then for every  $\epsilon > 0$ ,  $(\beta_S(\mathcal{D}_Y) + \epsilon, \gamma_S(\mathcal{D}_N) - \epsilon)\text{-Max-CSP}(f)$  requires  $\Omega(\sqrt{n})$  space in the streaming setting.*

We prove that Theorem 11 fails to rule out streaming  $(\frac{2}{9} + \epsilon)$ -approximations to  $\text{Max-3AND}$  in the following sense:

► **Theorem 12.** *There is no infinite sequence  $(\mathcal{D}_Y^{(1)}, \mathcal{D}_N^{(1)}), (\mathcal{D}_Y^{(2)}, \mathcal{D}_N^{(2)}), \dots$  of padded one-wise pairs on  $\Delta_3$  such that*

$$\lim_{t \rightarrow \infty} \frac{\beta_{\{3\}}(\mathcal{D}_N^{(t)})}{\gamma_{\{3\}}(\mathcal{D}_Y^{(t)})} = \frac{2}{9}.$$

Theorem 12 is proven formally in Section 6; we give a proof outline in Appendix A.

Yet we still can achieve decent bounds using padded one-wise pairs:

► **Observation 13.** *The padded one-wise pair  $\mathcal{D}_N = (0, 0.45, 0.45, 0.1)$ ,  $\mathcal{D}_Y = (0.45, 0, 0, 0.55)$  (discovered by numerical search) does prove a streaming approximability upper bound of  $\approx .2362$  for 3AND, which is still quite close to  $\alpha(3\text{AND}) = \frac{2}{9}$ .*

## 3 Formulas for $\mu$ , $\lambda_S$ , and $\gamma_{S,k}$

In this section, we give explicit formulas for the quantities  $\mu(\mathcal{D})$ ,  $\lambda_S(\mathcal{D}, p)$ , and  $\gamma_{S,k}(\mu)$  (defined in Equations (2), (3), and (6), respectively) which will be used throughout the rest of the paper. For  $i \in [k]$ , let  $\epsilon_{i,k} \stackrel{\text{def}}{=} -1 + \frac{2i}{k}$ .

► **Lemma 14.** *For any  $\mathcal{D} \in \Delta_k$ ,*

$$\mu(\mathcal{D}) = \sum_{i=0}^k \epsilon_{i,k} \mathcal{D}\langle i \rangle.$$

**Proof of Lemma 14.** By definition (Equation (3)),  $\mu(\mathcal{D}) = \mathbb{E}_{\mathbf{b} \sim \mathcal{D}}[b_1]$ . We use linearity of expectation; the contribution of weight- $i$  vectors to  $\mu(\mathcal{D})$  is  $\mathcal{D}\langle i \rangle \cdot \frac{1}{k}(i \cdot 1 + (k - i) \cdot (-1)) = \epsilon_{i,k} \mathcal{D}\langle i \rangle$ . ◀

► **Lemma 15.** For any  $\mathcal{D} \in \Delta_k$  and  $p \in [0, 1]$ , we have

$$\lambda_S(\mathcal{D}, p) = \sum_{s \in S} \sum_{i=0}^k \left( \sum_{j=\max\{0, s-(k-i)\}}^{\min\{i, s\}} \binom{i}{j} \binom{k-i}{s-j} q^{s+i-2j} p^{k-s-i+2j} \right) \mathcal{D}\langle i \rangle$$

where  $q \stackrel{\text{def}}{=} 1 - p$ .

The proof of Lemma 15 is given in the full version [3].

► **Lemma 16.** Let  $S \subseteq [k]$ , and let  $s$  be its smallest element and  $t$  its largest element (they need not be distinct). Then for  $\mu \in [-1, 1]$ ,

$$\gamma_{S,k}(\mu) = \begin{cases} \frac{1+\mu}{1+\epsilon_{s,k}} & \mu \in [-1, \epsilon_{s,k}) \\ 1 & \mu \in [\epsilon_{s,k}, \epsilon_{t,k}] \\ \frac{1-\mu}{1-\epsilon_{t,k}} & \mu \in (\epsilon_{t,k}, 1] \end{cases}$$

(which also equals  $\min \left\{ \frac{1+\mu}{1+\epsilon_{s,k}}, 1, \frac{1-\mu}{1-\epsilon_{t,k}} \right\}$ ).

The proof of Lemma 16 is given in the full version [3].

## 4 Analysis of $\alpha(k\text{AND})$

In this section, we prove Theorem 1 (on the sketching approximability of  $\text{Max-}k\text{AND}$ ). Recall that in Equation (1), we defined

$$\alpha'_k = \left( \frac{(k-1)(k+1)}{4k^2} \right)^{(k-1)/2}.$$

Theorem 1 follows immediately from the following two lemmas:

► **Lemma 17.** For all odd  $k \geq 3$ ,  $\alpha(k\text{AND}) \leq \alpha'_k$ . For all even  $k \geq 2$ ,  $\alpha(k\text{AND}) \leq 2\alpha'_{k+1}$ .

► **Lemma 18.** For all odd  $k \geq 3$ ,  $\alpha(k\text{AND}) \geq \alpha'_k$ . For all even  $k \geq 2$ ,  $\alpha(k\text{AND}) \geq 2\alpha'_{k+1}$ .

To begin, we give explicit formulas for  $\gamma_{\{k\},k}(\mu(\mathcal{D}))$  and  $\lambda_{\{k\}}(\mathcal{D}, p)$ . Note that the smallest element of  $\{k\}$  is  $k$ , and  $\epsilon_{k,k} = 1$ . Thus, for  $\mathcal{D} \in \Delta_k$ , we have by Lemmas 14 and 16 that

$$\gamma_{\{k\},k}(\mu(\mathcal{D})) = \frac{1 + \sum_{i=0}^k (-1 + \frac{2i}{k}) \mathcal{D}\langle i \rangle}{2} = \sum_{i=0}^k \frac{i}{k} \mathcal{D}\langle i \rangle. \quad (10)$$

Similarly, we can apply Lemma 15 with  $s = k$ ; for each  $i \in \{0\} \cup [k]$ ,  $\max\{0, s - (k - i)\} = \min\{i, k\} = i$ , so we need only consider  $j = i$ , and then  $\binom{i}{j} = \binom{k-i}{s-j} = 1$ . Thus, for  $q = 1 - p$ , we have

$$\lambda_{\{k\}}(\mathcal{D}, p) = \sum_{i=0}^k q^{k-i} p^i \mathcal{D}\langle i \rangle \quad (11)$$

The proof of Lemma 17 is given in Appendix A. We also prove Lemma 18 in Appendix A using the max-min method. We rely on the following proposition which is a simple inequality for optimizing ratios of linear functions, which we prove in the full version [3]:

► **Proposition 19.** Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be defined by the equation  $f(\mathbf{x}) = \frac{\mathbf{a} \cdot \mathbf{x}}{\mathbf{b} \cdot \mathbf{x}}$  for some  $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$ . For every  $\mathbf{y}(1), \dots, \mathbf{y}(r) \in \mathbb{R}_{\geq 0}^n$ , and every  $\mathbf{x} = \sum_{i=1}^r \alpha_i \mathbf{y}(i)$  with each  $\alpha_i \geq 0$ , we have  $f(\mathbf{x}) \geq \min_i f(\mathbf{y}(i))$ . In particular, taking  $r = n$  and  $\mathbf{y}(1), \dots, \mathbf{y}(n)$  as the standard basis for  $\mathbb{R}^n$ , for every  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$ , we have  $f(\mathbf{x}) \geq \min_i \frac{a_i}{b_i}$ .

## 5 Further analyses of $\alpha(f)$ for symmetric Boolean functions $f$

### 5.1 $\text{Th}_k^{k-1}$ for even $k$

In this subsection, we prove Theorem 3 (on the sketching approximability of  $\text{Th}_k^{k-1}$  for even  $k \geq 2$ ). It is necessary and sufficient to prove the following two lemmas:

► **Lemma 20.** For all even  $k \geq 2$ ,  $\alpha(\text{Th}_k^{k-1}) \leq \frac{k}{2} \alpha'_{k-1}$ .

► **Lemma 21.** For all even  $k \geq 2$ ,  $\alpha(\text{Th}_k^{k-1}) \geq \frac{k}{2} \alpha'_{k-1}$ .

Firstly, we give explicit formulas for  $\gamma_{\{k-1,k\},k}$  and  $\lambda_{\{k-1,k\}}$ . We have  $\text{Th}_k^{k-1} = f_{\{k-1,k\},k}$ , and  $\epsilon_{k-1,k} = -1 + \frac{2(k-1)}{k} = 1 - \frac{2}{k}$ . Thus, Lemmas 14 and 16 give

$$\gamma_{\{k-1,k\},k}(\mu(\mathcal{D})) = \min \left\{ \frac{1 + \sum_{i=0}^k (-1 + \frac{2i}{k}) \mathcal{D}\langle i \rangle}{2 - \frac{2}{k}}, 1 \right\} = \min \left\{ \sum_{i=0}^k \frac{i}{k-1} \mathcal{D}\langle i \rangle, 1 \right\}. \quad (12)$$

Next, we calculate  $\lambda_{\{k-1,k\}}(\mathcal{D}, p)$  with Lemma 15. Let  $q = 1 - p$ , and let us examine the coefficient on  $\mathcal{D}\langle i \rangle$ .  $s = k$  contributes  $q^{k-i} p^k$ . In the case  $i \leq k-1$ ,  $s = k-1$  contributes  $(k-i)q^{k-i-1} p^{i+1}$  for  $j = i$ , and in the case  $i \geq 1$ ,  $s = k-1$  contributes  $iq^{k-i+1} p^{i-1}$  for  $j = i-1$ . Thus, altogether we can write

$$\lambda_{\{k-1,k\}}(\mathcal{D}, p) = \sum_{i=0}^k q^{k-i-1} p^{i-1} ((k-i)p^2 + pq + iq^2) \mathcal{D}\langle i \rangle. \quad (13)$$

The proofs of Lemmas 20 and 21 are given in Appendix A.

► **Observation 22.** For  $\text{Th}_4^3$  the optimal  $\mathcal{D}_N^* = (0, 0, \frac{4}{5}, \frac{1}{5}, 0)$  does participate in a padded one-wise pair with  $\mathcal{D}_Y^* = (\frac{4}{15}, 0, 0, \frac{11}{15}, 0)$  (given by  $\mathcal{D}_0 = (0, 0, 0, 1, 0)$ ,  $\tau = \frac{1}{5}$ ,  $\mathcal{D}'_N = (0, 0, 1, 0, 0)$ , and  $\mathcal{D}'_Y = (\frac{4}{15}, 0, 0, \frac{8}{15}, 0)$ ) so we can rule out streaming  $(\frac{4}{9} + \epsilon)$ -approximations to  $\text{Max-CSP}(\text{Th}_4^3)$  in  $o(\sqrt{n})$  space.

### 5.2 $\text{Ex}_k^{(k+1)/2}$ for (small) odd $k$

In this section, we prove bounds on the sketching approximability of  $\text{Ex}_k^{(k+1)/2}$  for odd  $k \in \{3, \dots, 51\}$ . Define  $\mathcal{D}_{0,k} \in \Delta_k$  by  $\mathcal{D}_{0,k}\langle 0 \rangle = \frac{k-1}{2k}$  and  $\mathcal{D}_{0,k}\langle k \rangle = \frac{k+1}{2k}$ . We prove the following two lemmas:

► **Lemma 23.** For all odd  $k \geq 3$ ,  $\alpha(\text{Ex}_k^{(k+1)/2}) \leq \lambda_{\{\frac{k+1}{2}\}}(\mathcal{D}_{0,k}, p'_k)$ , where  $p'_k \stackrel{\text{def}}{=} \frac{3k-k^2+\sqrt{4k+k^2-2k^3+k^4}}{4k}$ .

► **Lemma 24.** The following holds for all odd  $k \in \{3, \dots, 51\}$ . For all  $p \in [0, 1]$ , the expression  $\frac{\lambda_{\{\frac{k+1}{2}\}}(\cdot, p)}{\gamma_{\{\frac{k+1}{2}\},k}(\mu(\cdot))}$  is minimized at  $\mathcal{D}_{0,k}$ .

We begin by writing an explicit formula for  $\lambda_{\{\frac{k+1}{2}\}}$ . Lemma 15 gives

$$\lambda_{\{\frac{k+1}{2}\}}(\mathcal{D}, p) = \sum_{i=0}^k \left( \sum_{j=\max\{0, i-\frac{k-1}{2}\}}^{\min\{i, \frac{k+1}{2}\}} \binom{i}{j} \binom{k}{\frac{k+1}{2}-j} (1-p)^{(k+1)/2+i-2j} p^{(k-1)/2-i+2j} \right) \mathcal{D}\langle i \rangle.$$

For  $i \leq \frac{k-1}{2}$ , the sum over  $j$  goes from 0 to  $i$ , and for  $i \geq \frac{k+1}{2}$ , it goes from  $i - \frac{k-1}{2}$  to  $\frac{k+1}{2}$ . Thus, plugging in  $\mathcal{D}_{0,k}$ , we get:

$$\lambda_{\{\frac{k+1}{2}\}}(\mathcal{D}_{0,k}, p) = \binom{k}{\frac{k+1}{2}} \left( \frac{k-1}{2k} (1-p)^{(k+1)/2} p^{(k-1)/2} + \frac{k+1}{2k} (1-p)^{(k-1)/2} p^{(k+1)/2} \right). \quad (14)$$

By Lemmas 14 and 16,  $\gamma_{\{\frac{k+1}{2}\},k}(\mu(\mathcal{D}_{0,k})) = \gamma_{\{\frac{k+1}{2}\},k}(\frac{1}{k}) = 1$ . Thus, Lemmas 23 and 24 together imply the following theorem:

► **Theorem 25.** For odd  $k \in \{3, \dots, 51\}$ ,

$$\alpha(\text{Ex}_k^{(k+1)/2}) = \binom{k}{\frac{k+1}{2}} \left( \frac{k-1}{2k} (1-p'_k)^{(k+1)/2} (p'_k)^{(k-1)/2} + \frac{k+1}{2k} (1-p'_k)^{(k-1)/2} (p'_k)^{(k+1)/2} \right),$$

where  $p'_k = \frac{3k-k^2+\sqrt{4k+k^2-2k^3+k^4}}{4k}$  as in Lemma 23.

Recall that  $\rho(f_{(k+1)/2,k}) = \binom{k}{\frac{k+1}{2}} 2^{-k}$ . Although we currently lack a lower bound on  $\alpha(\text{Ex}_k^{(k+1)/2})$  for large odd  $k$ , the upper bound from Lemma 23 suffices to prove Theorem 5, i.e., it can be verified that

$$\lim_{k \text{ odd} \rightarrow \infty} \frac{\binom{k}{\frac{k+1}{2}} \left( \frac{k-1}{2k} (1-p'_k)^{(k+1)/2} (p'_k)^{(k-1)/2} + \frac{k+1}{2k} (1-p'_k)^{(k-1)/2} (p'_k)^{(k+1)/2} \right)}{\rho(\text{Ex}_k^{(k+1)/2})} = 1.$$

We remark that for  $\text{Ex}_k^{(k+1)/2}$ , our lower bound (Lemma 24) is *stronger* than what we were able to prove for  $k\text{AND}$  (Lemma 18) and  $\text{Th}_k^{k-1}$  (Lemma 21) because the inequality holds regardless of  $p$ . This is fortunate for us, as the optimal  $p^*$  from Lemma 23 is rather messy.<sup>6</sup> The proofs of Lemmas 23 and 24 are given in the full version [3].

### 5.3 More symmetric functions

In Table 1 below, we list four more symmetric Boolean functions (beyond  $k\text{AND}$ ,  $\text{Th}_k^{k-1}$ , and  $\text{Ex}_k^{(k+1)/2}$ ) whose sketching approximability we have analytically resolved using the “max-min method”. These values were calculated using two functions in the Mathematica code, `estimateAlpha` – which numerically or symbolically estimates the  $\mathcal{D}_N$ , with a given support, which minimizes  $\alpha$  – and `testMinMax` – which, given a particular  $\mathcal{D}_N$ , calculates  $p^*$  for that  $\mathcal{D}_N$  and checks analytically whether lower-bounding by evaluating  $\lambda_S$  at  $p^*$  proves that  $\mathcal{D}_N$  is minimal.

■ **Table 1** Symmetric functions for which we have analytically calculated exact  $\alpha$  values using the “max-min method”. For a polynomial  $P : \mathbb{R} \rightarrow \mathbb{R}$  with a *unique* positive real root, let  $\text{root}_{\mathbb{R}}(p)$  denote that root, and define the polynomials  $P_1(z) = -72 + 4890z - 108999z^2 + 800000z^3$ ,  $P_2(z) = -908 + 5021z - 9001z^2 + 5158z^3$ ,  $P_3(z) = -60 + 5745z - 183426z^2 + 1953125z^3$ ,  $P_4(z) = -344 + 1770z - 3102z^2 + 1811z^3$ . (We note that in the  $f_{\{4\},5}$  and  $f_{\{4,5\},5}$  calculations, we were required to check equality of roots numerically (to high precision) instead of analytically).

| $S$           | $k$ | $\alpha$                                             | $\mathcal{D}_N^*$                                                                |
|---------------|-----|------------------------------------------------------|----------------------------------------------------------------------------------|
| $\{2, 3\}$    | 3   | $\frac{1}{2} + \frac{\sqrt{3}}{18} \approx 0.5962$   | $(0, \frac{1}{2}, 0, \frac{1}{2})$                                               |
| $\{4, 5\}$    | 5   | $8 \text{root}_{\mathbb{R}}(P_1) \approx 0.2831$     | $(0, 0, 1 - \text{root}_{\mathbb{R}}(P_2), \text{root}_{\mathbb{R}}(P_2), 0, 0)$ |
| $\{4\}$       | 5   | $8 \text{root}_{\mathbb{R}}(P_3) \approx 0.2394$     | $(0, 0, 1 - \text{root}_{\mathbb{R}}(P_4), \text{root}_{\mathbb{R}}(P_4), 0, 0)$ |
| $\{3, 4, 5\}$ | 5   | $\frac{1}{2} + \frac{3\sqrt{5}}{125} \approx 0.5537$ | $(0, \frac{1}{2}, 0, 0, 0, \frac{1}{2})$                                         |

We remark that two of the cases in Table 1 (as well as  $k\text{AND}$ ), the optimal  $\mathcal{D}_N$  is rational and supported on two coordinates. However, in the other two cases in Table 1, the optimal  $\mathcal{D}_N$  involves roots of a cubic.

<sup>6</sup> The analogous statement is false for e.g.  $3\text{AND}$ , where we had  $\mathcal{D}_N^* = (0, 0, 1, 0)$ , but at  $p = \frac{3}{4}$ ,

$$\frac{\lambda_{\{3\}}((0, \frac{1}{2}, \frac{1}{2}, 0), \frac{3}{4})}{\gamma_{\{3\},3}(\mu(0, \frac{1}{2}, \frac{1}{2}, 0))} = \frac{3}{16} \leq \frac{27}{128} = \frac{\lambda_{\{3\}}((0, 0, 1, 0), \frac{3}{4})}{\gamma_{\{3\},3}(\mu(0, 0, 1, 0))}.$$

In Section 5.2, we showed that  $\mathcal{D}_N^*$  defined by  $\mathcal{D}_N^*(0) = \frac{k-1}{2k}$  and  $\mathcal{D}_N^*(k) = \frac{k+1}{2k}$  is optimal for  $\text{Ex}_k^{(k+1)/2}$  for odd  $k \in \{3, \dots, 51\}$ . Using the same  $\mathcal{D}_N^*$ , we are also able to resolve 11 other cases in which  $S$  is “close to”  $\{\frac{k+1}{2}\}$ ; for instance,  $S = \{5, 6\}, \{5, 6, 7\}, \{5, 7\}$  for  $k = 9$ . (We have omitted the values of  $\alpha$  and  $\mathcal{D}_N$  because they are defined using the roots of polynomials of degree up to 8.)

In all previously-mentioned cases, the condition “ $\mathcal{D}_N^*$  has support size 2” was helpful, as it makes the optimization problem over  $\mathcal{D}_N^*$  essentially univariate; however, we have confirmed analytically in two other cases ( $S = \{3\}, k = 4$  and  $S = \{3, 5\}, k = 5$ ) that “max-min method on distributions with support size two” does not suffice for tight bounds on  $\alpha$  (see `testDistsWithSupportSize2` in the Mathematica code). However, using the max-min method with  $\mathcal{D}_N$  supported on two levels still achieves decent (but not tight) bounds on  $\alpha$ . For  $S = \{3\}, k = 4$ , using  $\mathcal{D}_N = (\frac{1}{4}, 0, 0, 0, \frac{3}{4})$ , we get the bounds  $\alpha(f_{\{3\},4}) \in [0.3209, 0.3295]$  (the difference being 2.67%). For  $S = \{3, 5\}, k = 5$ , using  $\mathcal{D}_N = (\frac{1}{4}, 0, 0, 0, \frac{3}{4}, 0)$ , we get  $\alpha(f_{\{3,5\},5}) \in [0.3416, 0.3635]$  (the difference being 6.42%).

Finally, we have also analyzed cases where we get numerical solutions which are very close to tight, but we lack analytical solutions because they likely involve roots of high-degree polynomials. For instance, in the case  $S = \{4, 5, 6\}, k = 6$ , setting  $\mathcal{D}_N = (0, 0, 0, 0.930013, 0, 0, 0.069987)$  gives  $\alpha(f_{\{4,5,6\},6}) \in [0.44409972, 0.44409973]$ , differing only by 0.000003%. (We conjecture here that  $\alpha = \frac{4}{9}$ .) For  $S = \{6, 7, 8\}, k = 8$ , using  $\mathcal{D}_N = (0, 0, 0, 0, 0.699501, 0.300499)$ , we get the bounds  $\alpha(f_{\{6,7,8\},8}) \in [0.20848, 0.20854]$  (the difference being 0.02%).<sup>7</sup>

## 6 Incompleteness of streaming lower bounds: Proving Theorem 12

In this section, we prove Theorem 12, showing that the streaming lower bounds from [7] (Theorem 11) cannot characterize the *streaming* approximability of 3AND.

► **Lemma 26.** *For  $\mathcal{D} \in \Delta_3$ , the expression*

$$\frac{\lambda_{\{3\}}(\mathcal{D}, \frac{1}{3}\mathcal{D}\langle 1 \rangle + \frac{2}{3}\mathcal{D}\langle 2 \rangle + \mathcal{D}\langle 3 \rangle)}{\gamma_{\{3\},3}(\mu(\mathcal{D}))}$$

*is minimized uniquely at  $\mathcal{D} = (0, 0, 1, 0)$ , with value  $\frac{2}{9}$ .*

**Proof.** Letting  $p = \frac{1}{3}\mathcal{D}\langle 1 \rangle + \frac{2}{3}\mathcal{D}\langle 2 \rangle + \mathcal{D}\langle 3 \rangle$  and  $q = 1 - p$ , by Lemmas 14–16 the expression expands to

$$\frac{\mathcal{D}\langle 0 \rangle p^3 + \mathcal{D}\langle 1 \rangle p^2(1-p) + \mathcal{D}\langle 2 \rangle p(1-p)^2 + \mathcal{D}\langle 3 \rangle (1-p)^3}{\frac{1}{2}(1 - \mathcal{D}\langle 0 \rangle - \frac{1}{3}\mathcal{D}\langle 1 \rangle + \frac{1}{3}\mathcal{D}\langle 2 \rangle + \mathcal{D}\langle 3 \rangle)}.$$

The expression’s minimum, and its uniqueness, are confirmed analytically in the Mathematica code. ◀

► **Lemma 27.** *Let  $X$  be a compact topological space,  $Y \subseteq X$  a closed subspace,  $Z$  a topological space, and  $f : X \rightarrow Z$  a continuous map. Let  $x^* \in X, z^* \in Z$  be such that  $f^{-1}(z^*) = \{x^*\}$ . Let  $\{x_i\}_{i \in \mathbb{N}}$  be a sequence of points in  $Y$  such that  $\{f(x_i)\}_{i \in \mathbb{N}}$  converges to  $z^*$ . Then  $x^* \in Y$ .*

<sup>7</sup> Interestingly, in this latter case, we get bounds differing by 2.12% using  $\mathcal{D}_N = (0, 0, 0, 0, \frac{9}{13}, \frac{4}{13}, 0, 0, 0)$  in an attempt to continue the pattern from  $f_{\{7,8\},8}$  and  $f_{\{8\},8}$  (where we set  $\mathcal{D}_N^* = (0, 0, 0, 0, \frac{16}{25}, \frac{9}{25}, 0, 0, 0)$  and  $(0, 0, 0, 0, \frac{25}{41}, \frac{16}{41}, 0, 0, 0)$  in Section 5.1 and Section 4, respectively).



**Proof.** By compactness of  $X$ , there is a subsequence  $\{x_{j_i}\}_{i \in \mathbb{N}}$  which converges to a limit  $\tilde{x}$ . By closure,  $\tilde{x} \in Y$ . By continuity,  $f(\tilde{x}) = z^*$ , so  $\tilde{x} = x^*$ .  $\blacktriangleleft$

Finally, the proof of Theorem 12 is given in Appendix A.

## 7 Simple sketching algorithms for threshold functions

The main goal of this section is to prove Theorem 6, giving a simple “bias-based” sketching algorithm for threshold functions  $\text{Th}_k^i$ . Given an instance  $\Psi$  of  $\text{Max-CSP}(\text{Th}_k^i)$ , for  $i \in [n]$ , let  $\text{diff}_i(\Psi)$  denote the total weight of clauses in which  $x_i$  appears positively minus the weight of those in which it appears negatively; that is, if  $\Psi$  consists of clauses  $(\mathbf{b}(1), \mathbf{j}(1)), \dots, (\mathbf{b}(m), \mathbf{j}(m))$  with weights  $w_1, \dots, w_m$ , then

$$\text{diff}_i(\Psi) \stackrel{\text{def}}{=} \sum_{\ell \in [m] \text{ s.t. } j(\ell)_t = i \text{ for some } t \in [k]} b(\ell)_t w_\ell.$$

Let  $\text{bias}(\Psi) \stackrel{\text{def}}{=} \frac{1}{kW} \sum_{i=1}^n |\text{diff}_i(\Psi)|$ , where  $W = \sum_{\ell=1}^m w_\ell$  is the total weight in  $\Psi$ .

Let  $S = \{i, \dots, k\}$  so that  $\text{Th}_k^i = f_{S,k}$ . Recall the definitions of  $\beta_{S,k}(\mu)$  and  $\gamma_{S,k}(\mu)$  from Equation (7). Our simple algorithm for  $\text{Max-CSP}(\text{Th}_k^i)$  relies on the following two lemmas, which we prove below:

► **Lemma 28.**  $\text{val}_\Psi \leq \gamma_{S,k}(\text{bias}(\Psi))$ .

► **Lemma 29.**  $\text{val}_\Psi \geq \beta_{S,k}(\text{bias}(\Psi))$ .

Together, these two lemmas imply that outputting  $\alpha(\text{Th}_k^i) \cdot \gamma_{S,k}(\text{bias}(\Psi))$  gives an  $\alpha(\text{Th}_k^i)$ -approximation to  $\text{Max-CSP}(\text{Th}_k^i)$ , since  $\alpha(\text{Th}_k^i) = \inf_{\mu \in [-1,1]} \frac{\beta_{S,k}(\mu)}{\gamma_{S,k}(\mu)}$  (Equation (7)). We can implement this as a small-space sketching algorithm (up to an arbitrarily small constant  $\epsilon > 0$  in the approximation ratio) because  $\text{bias}(\Psi)$  is measurable using  $\ell_1$ -sketching algorithms (as used also in [12, 8, 7]) and  $\gamma_{S,k}(\cdot)$  is piecewise linear:

► **Theorem 30** ([16, 17]). *For every  $\epsilon > 0$ , there exists an  $O(\log n/\epsilon^2)$ -space randomized sketching algorithm for the following problem: The input is a stream  $S$  of updates of the form  $(i, v) \in [n] \times \{-\text{poly}(n), \dots, \text{poly}(n)\}$ , and the goal is to estimate the  $\ell_1$ -norm of the vector  $x \in [n]^n$  defined by  $x_i = \sum_{(i,v) \in S} v$ , up to a multiplicative factor of  $1 \pm \epsilon$ .*

► **Corollary 31.** *For  $f : \{-1, 1\}^k \rightarrow \{0, 1\}$  and every  $\epsilon > 0$ , there exists an  $O(\log n/\epsilon^2)$ -space randomized sketching algorithm for the following problem: The input is an instance  $\Psi$  of  $\text{Max-CSP}(\text{Th}_k^i)$  (given as a stream of constraints), and the goal is to estimate  $\text{bias}(\Psi)$  up to a multiplicative factor of  $1 \pm \epsilon$ .*

**Proof.** Invoke the  $\ell_1$ -norm sketching algorithm from Theorem 30 as follows: On each input constraint  $(\mathbf{b} = (b_1, \dots, b_k), \mathbf{j} = (j_1, \dots, j_k))$  with weight  $w$ , insert the updates  $(j_1, wb_1), \dots, (j_k, wb_k)$  into the stream (and normalize appropriately).  $\blacktriangleleft$

Theorem 6 then follows from Lemmas 16, 28, and 29 and Corollary 31; we include a formal proof in Appendix A for completeness.

To prove Lemmas 28 and 29, we require a bit more setup. Adapting notation from [7, §4.2], given an instance  $\Psi$  of  $\text{Max-CSP}(\text{Th}_k^i)$  and a “negation pattern”  $\mathbf{a} = (a_1, \dots, a_n) \in \{-1, 1\}^n$  for the variables, let  $\Psi^{\mathbf{a}}$  be the instance which results from  $\Psi$  by “flipping” the variables according to  $\mathbf{a}$  (formally, each constraint  $(\mathbf{b}, \mathbf{j})$  is replaced with  $(\mathbf{b} \odot \mathbf{a}|_{\mathbf{j}}, \mathbf{j})$ ). We summarize the useful properties of this operation in the following claim:

► **Proposition 32.** *Let  $\Psi$  be an instance of  $\text{Max-CSP}(\text{Th}_k^i)$  and  $\mathbf{a} = (a_1, \dots, a_n) \in \{-1, 1\}^n$ . Then:*

- i For each  $i \in [n]$ ,  $\text{diff}_i(\Psi^{\mathbf{a}}) = a_i \text{diff}_i(\Psi)$ .
- ii  $\text{bias}(\Psi) = \text{bias}(\Psi^{\mathbf{a}})$ .
- iii For any  $\sigma \in \{-1, 1\}^n$ ,  $\text{val}_{\Psi^{\mathbf{a}}}(\sigma) = \text{val}_{\Psi}(\mathbf{a} \odot \sigma)$ .
- iv  $\text{val}_{\Psi^{\mathbf{a}}} = \text{val}_{\Psi}$ .

The proof of Proposition 32 is given in the full version [3].

Also, given an instance  $\Psi$ , we define its “symmetrized canonical distribution”  $\mathcal{D}_{\Psi}^{\text{sym}} \in \Delta_k$  to be the distribution obtained by sampling a constraint at random from  $\Psi$  and outputting its “randomly permuted negation pattern”. Formally, let  $\mathbf{S}_k$  denote the set of permutations  $[k] \rightarrow [k]$ . For a vector  $\mathbf{b} = (b_1, \dots, b_k) \in \{-1, 1\}^k$  and a permutation  $\pi \in \mathbf{S}_k$ , let  $\pi(\mathbf{b}) = (b_{\pi(1)}, \dots, b_{\pi(k)})$ . Let  $C(i) = (\mathbf{b}(i), \mathbf{j}(i))$  denote the  $i$ -th constraint of  $\Psi$ , with weight  $w_i$ , and let  $W = \sum_{i=1}^m w_i$  be the total weight. To sample a random vector from  $\mathcal{D}_{\Psi}^{\text{sym}}$ , we sample  $i \in [m]$  with probability  $w_i/W$ , sample a permutation  $\pi \sim \text{Unif}(\mathbf{S}_k)$ , and output  $\pi(\mathbf{b}(i))$ . The useful properties of  $\mathcal{D}_{\Psi}^{\text{sym}}$  are summarized in the following claim:

► **Proposition 33.** *Let  $\Psi$  be an instance of  $\text{Max-CSP}(\text{Th}_k^i)$ . Then:*

- i For any  $p \in [0, 1]$ ,  $\mathbb{E}_{\mathbf{a} \sim \text{Bern}(p)^n} [\text{val}_{\Psi}(\mathbf{a})] = \lambda_S(\mathcal{D}_{\Psi}^{\text{sym}}, p)$ .
- ii  $\mu(\mathcal{D}_{\Psi}^{\text{sym}}) = \frac{1}{kW} \sum_{i=1}^n \text{diff}_i(\Psi) \leq \text{bias}(\Psi)$ .
- iii If  $\text{diff}_i(\Psi) \geq 0$  for all  $i \in [n]$ , then  $\mu(\mathcal{D}_{\Psi}^{\text{sym}}) = \text{bias}(\Psi)$ .

The proof of Proposition 33 is given in the full version [3]. The proofs of Lemmas 28 and 29 are given in Appendix A.

Finally, we state another consequence of Lemma 28 – a simple randomized,  $O(n)$ -time-and-space streaming algorithm for *outputting* approximately-optimal assignments when the max-min method applies.

► **Theorem 34.** *Let  $\text{Th}_k^i$  be a threshold function and  $p^* \in [0, 1]$  be such that the max-min method applies, i.e.,*

$$\alpha(\text{Th}_k^i) = \inf_{\mathcal{D}_N \in \Delta_k} \left( \frac{\lambda_S(\mathcal{D}_N, p^*)}{\gamma_{S,k}(\mu(\mathcal{D}_N))} \right).$$

*Then the following algorithm, on input  $\Psi$ , outputs an assignment with expected value at least  $\alpha(\text{Th}_k^i) \cdot \text{val}_{\Psi}$ : Assign every variable to 1 if  $\text{diff}_i(\Psi) \geq 0$ , and  $-1$  otherwise, and then flip each variable’s assignment independently with probability  $p^*$ .*

The proof of Theorem 34 is given in the full version [3].

## Discussion

In this paper, we introduce the max-min method and use it to resolve the streaming approximability of a wide variety of symmetric Boolean CSPs (including infinite families such as  $\text{Max-}k\text{AND}$  for all  $k$ , and  $\text{Th}_k^{k-1}$  for all even  $k$ ). However, these techniques are in a sense “ad hoc” since we use computer assistance to guess the optimal solution for our optimization problem. We leave the question of whether the max-min method can be applied to determine the sketching approximability for all symmetric Boolean CSPs as an interesting open problem.

Separately, we also establish that the techniques developed in [7] are not sufficient to characterize the *streaming* approximability of all CSPs. Indeed, we show that their streaming lower bound based on “padded one-wise pairs” cannot match the approximation ratio of their

optimal sketching algorithm for Max-3AND. While we believe that no  $o(\sqrt{n})$ -space streaming algorithm can beat their sketching algorithm for Max-3AND, proving this will require new techniques.

## References

- 1 Arindam Biswas and Venkatesh Raman. Sublinear-Space Approximation Algorithms for Max  $r$ -SAT. In *Computing and Combinatorics (COCOON 2021, Tainan, Taiwan, October 24-26, 2021)*, volume 13025 of *LNCS*, pages 124–136. Springer, Cham, 2021.
- 2 Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, first edition, March 2004. doi:10.1017/CB09780511804441.
- 3 Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. Sketching approximations for (some) symmetric Boolean CSPs: Closed-form ratios and simple algorithms, February 2022. arXiv:2112.06319.
- 4 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms*, 5(3):1–14, July 2009. Conference version in SODA 2007. doi:10.1145/1541885.1541893.
- 5 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022, Rome, Italy, June 20-24, 2022)*, 2022. To appear.
- 6 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021, Denver, CO, USA, February 7-10, 2022)*. IEEE Computer Society, 2021. doi:10.1109/FOCS52979.2021.00117.
- 7 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs with linear sketches, February 2022. arXiv:2102.12351v7.
- 8 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max- $k$ SAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 330–341. IEEE Computer Society, November 2020. doi:10.1109/FOCS46700.2020.00039.
- 9 Lars Engebretsen and Jonas Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. *Random Structures and Algorithms*, 33(4):497–514, December 2008. Conference version in STACS 2005. doi:10.1002/rsa.20226.
- 10 Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS 2003, January 4-6, 1995)*, pages 182–189. IEEE Computer Society, 1995. doi:10.1109/ISTCS.1995.377033.
- 11 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995. Conference version in STOC 1994. doi:10.1145/227683.227684.
- 12 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2017, Berkeley, CA, USA, August 16-18, 2017)*, volume 81 of *LIPICs*, pages 8:1–8:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, August 2017. doi:10.4230/LIPICs.APPROX-RANDOM.2017.8.
- 13 Gustav Hast. Approximating Max  $k$ CSP Using Random Restrictions. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2004, Cambridge, MA, USA, August 22-24, 2004)*, volume 3122 of *LNCS*, pages 151–162. Springer, 2004. doi:10.1007/978-3-540-27821-4\_14.

- 14 Gustav Hast. Approximating Max  $k$ CSP – Outperforming a Random Assignment with Almost a Linear Factor. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming (ICALP 2005, July 11-15, 2005)*, volume 3580 of *LNCS*, pages 956–968. Springer, 2005. doi:10.1007/11523468\_77.
- 15 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.
- 16 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, May 2006. Conference version in FOCS 2000. doi:10.1145/1147954.1147955.
- 17 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the Exact Space Complexity of Sketching and Streaming Small Norms. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010, Austin, TX, USA, January 17-19, 2010)*, pages 1161–1178. Society for Industrial and Applied Mathematics, 2010. doi:10.1137/1.9781611973075.93.
- 18 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015, San Diego, California, USA, January 4-6, 2015)*, pages 1263–1282. Society for Industrial and Applied Mathematics, January 2015. doi:10.1137/1.9781611973730.84.
- 19 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000, Portland, OR, USA, May 21-23, 2000)*, pages 191–199, Portland, Oregon, United States, 2000. Association for Computing Machinery. doi:10.1145/335305.335329.
- 20 Alex Samorodnitsky and Luca Trevisan. Gowers Uniformity, Influence of Variables, and PCPs. *SIAM Journal on Computing*, 39(1):323–360, January 2009. Conference version in STOC 2006. doi:10.1137/070681612.
- 21 Noah Singer. *On Streaming Approximation Algorithms for Constraint Satisfaction Problems*. Bachelor’s thesis, Harvard University, Cambridge, MA, March 2022.
- 22 Madhu Sudan and Luca Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (SFCS 1998, Palo Alto, CA, USA, November 8-11, 1998)*, pages 18–27. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743425.
- 23 Luca Trevisan. Parallel Approximation Algorithms by Positive Linear Programming. *Algorithmica*, 21(1):72–88, May 1998. doi:10.1007/PL00009209.
- 24 Luca Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998, Dallas, Texas, USA, May 24-26, 1998)*, pages 299–308. Association for Computing Machinery, 1998. doi:10.1145/276698.276769.
- 25 Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, Approximation, and Linear Programming. *SIAM Journal on Computing*, 29(6):2074–2097, January 2000. Conference version in FOCS 1996. doi:10.1137/S0097539797328847.
- 26 Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998, San Francisco, CA, USA, January 25-27, 1998)*, pages 201–210. Association for Computing Machinery, 1998. doi:10.5555/314613.314701.

## A Miscellaneous technical proofs

**Proof outline of Theorem 12.** As discussed in Section 2.3, since  $k = 3$  is odd, to prove Theorem 1 we show, using the max-min method, that  $\mathcal{D}_N^* = (0, 0, 1, 0)$  minimizes  $\frac{\beta_{\{3\}}(\cdot)}{\gamma_{\{3\},3}(\mu(\cdot))}$ . We can show that the corresponding  $\gamma_{\{3\},3}$  value is achieved by  $\mathcal{D}_Y^* = (\frac{1}{3}, 0, 0, \frac{2}{3})$ . In particular,  $(\mathcal{D}_N^*, \mathcal{D}_Y^*)$  are not a padded one-wise pair.

We can show that the minimizer of  $\gamma_{\{3\}}$  for a particular  $\mu$  is in general unique. Hence, it suffices to furthermore show that  $\mathcal{D}_N^*$  is the *unique* minimizer of  $\frac{\beta_{\{3\}}(\cdot)}{\gamma_{\{3\},3}(\mu(\cdot))}$ . For this purpose, the max-min method is not sufficient because  $\frac{\lambda_{\{3\}}(\cdot, p^*)}{\gamma_{\{3\},3}(\mu(\cdot))}$  is not uniquely minimized at  $\mathcal{D}_N^*$  (where we chose  $p^* = \frac{2}{3}$ ). Intuitively, this is because  $p^*$  is not a good enough estimate for the maximizer of  $\lambda_{\{3\}}(\mathcal{D}_N, \cdot)$ . To remedy this, we observe that  $\lambda_{\{3\}}((1, 0, 0, 0), \cdot)$ ,  $\lambda_{\{3\}}((0, 1, 0, 0), \cdot)$ ,  $\lambda_{\{3\}}((0, 0, 1, 0), \cdot)$  and  $\lambda_{\{3\}}((0, 0, 0, 1), \cdot)$  are minimized at  $0, \frac{1}{3}, \frac{2}{3}$ , and  $1$ , respectively. Hence, we instead lower-bound  $\lambda_{\{3\}}(\mathcal{D}_N, \cdot)$  by evaluating at  $\frac{1}{3}\mathcal{D}_N\langle 1 \rangle + \frac{2}{3}\mathcal{D}_N\langle 2 \rangle + \mathcal{D}_N\langle 3 \rangle$ , which does suffice to prove the uniqueness of  $\mathcal{D}_N^*$ . The theorem then follows from continuity arguments.  $\blacktriangleleft$

**Proof of Lemma 17.** Consider the case where  $k$  is odd. Define  $\mathcal{D}_N^*$  by  $\mathcal{D}_N^*\langle \frac{k+1}{2} \rangle = 1$  and let  $p^* = \frac{1}{2} + \frac{1}{2k}$ . Since

$$\alpha(k\text{AND}) \leq \frac{\beta_{\{k\}}(\mathcal{D}_N^*)}{\gamma_{\{k\},k}(\mu(\mathcal{D}_N^*))} \text{ and } \beta_{\{k\}}(\mathcal{D}_N) = \sup_{p \in [0,1]} \lambda_{\{k\}}(\mathcal{D}_N^*, p),$$

by Equations (4) and (8), respectively, it suffices to check that  $p^*$  maximizes  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$  and

$$\frac{\lambda_{\{k\}}(\mathcal{D}_N^*, p^*)}{\gamma_{\{k\},k}(\mu(\mathcal{D}_N^*))} = \alpha'_k.$$

Indeed, by Equation (11),

$$\lambda_{\{k\}}(\mathcal{D}_N^*, p) = (1-p)^{(k-1)/2} p^{(k+1)/2}.$$

To show  $p^*$  maximizes  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$ , we calculate its derivative:

$$\frac{d}{dp} \left[ (1-p)^{(k-1)/2} p^{(k+1)/2} \right] = -(1-p)^{(k-3)/2} p^{(k-1)/2} \left( kp - \frac{k+1}{2} \right),$$

which has zeros only at  $0, 1$ , and  $p^*$ . Thus,  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$  has critical points only at  $0, 1$ , and  $p^*$ , and it is maximized at  $p^*$  since it vanishes at  $0$  and  $1$ . Finally, by Equations (10) and (11) and the definition of  $\alpha'_k$ ,

$$\frac{\lambda_{\{k\}}(\mathcal{D}_N^*, p^*)}{\gamma_{\{k\},k}(\mu(\mathcal{D}_N^*))} = \frac{\left(\frac{1}{2} - \frac{1}{2k}\right)^{(k-1)/2} \left(\frac{1}{2} + \frac{1}{2k}\right)^{(k+1)/2}}{\frac{1}{2} \left(1 + \frac{1}{k}\right)} = \alpha'_k,$$

as desired.

Similarly, consider the case where  $k$  is even; here, we define  $\mathcal{D}_N^*$  by  $\mathcal{D}_N^*\langle \frac{k}{2} \rangle = \frac{(\frac{k}{2}+1)^2}{(\frac{k}{2})^2 + (\frac{k}{2}+1)^2}$  and  $\mathcal{D}_N^*\langle \frac{k}{2} + 1 \rangle = \frac{(\frac{k}{2})^2}{(\frac{k}{2})^2 + (\frac{k}{2}+1)^2}$ , and set  $p^* = \frac{1}{2} + \frac{1}{2(k+1)}$ . Using Equation (11) to calculate the derivative of  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$  yields

$$\begin{aligned} \frac{d}{dp} \left[ \frac{(\frac{k}{2}+1)^2}{(\frac{k}{2})^2 + (\frac{k}{2}+1)^2} (1-p)^{k/2} p^{k/2} + \frac{(\frac{k}{2})^2}{(\frac{k}{2})^2 + (\frac{k}{2}+1)^2} (1-p)^{k/2-1} p^{k/2+1} \right] \\ = -\frac{k}{2+2k+2k^2} (1-p)^{k/2-2} p^{k/2-1} \left( \frac{k}{2} + 1 - 2p \right) \left( (k+1)p - \left( \frac{k}{2} + 1 \right) \right), \end{aligned}$$

so  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$  has critical points at  $0, 1, \frac{1}{2} + \frac{k}{4}$ , and  $p^*$ ;  $p^*$  is the only critical point in the interval  $[0, 1]$  for which  $\lambda_{\{k\}}(\mathcal{D}_N^*, \cdot)$  is positive, and hence is its maximum. Finally, it can be verified algebraically using Equations (10) and (11) that  $\frac{\lambda_{\{k\}}(\mathcal{D}_N^*, p^*)}{\gamma_{\{k\},k}(\mu(\mathcal{D}_N^*))} = 2\alpha'_{k+1}$ , as desired.  $\blacktriangleleft$

**Proof of Lemma 18.** First, suppose  $k \geq 3$  is odd. Set  $p^* = \frac{1}{2} + \frac{1}{2k} = \frac{k+1}{2k}$ . We want to show that

$$\begin{aligned} \alpha'_k &\leq \inf_{\mathcal{D}_N \in \Delta_k} \frac{\lambda_{\{k\}}(\mathcal{D}_N, p^*)}{\gamma_{\{k\},k}(\mu(\mathcal{D}_N))} && \text{(max-min inequality, i.e., Equation (9))} \\ &= \inf_{\mathcal{D}_N \in \Delta_k} \frac{\sum_{i=0}^k (1-p^*)^{k-i} (p^*)^i \mathcal{D}_N \langle i \rangle}{\sum_{i=0}^k \frac{i}{k} \mathcal{D}_N \langle i \rangle}. && \text{(Equations (10) and (11))} \end{aligned}$$

By Proposition 19, it suffices to check that

$$\forall i \in \{0\} \cup [k], \quad (1-p^*)^{k-i} (p^*)^i \geq \alpha'_k \cdot \frac{i}{k}.$$

By definition of  $\alpha'_k$ , we have that  $\alpha'_k = (1-p^*)^{(k-1)/2} (p^*)^{(k-1)/2}$ . Defining  $r = \frac{p^*}{1-p^*} = \frac{k+1}{k-1}$  (so that  $p^* = r(1-p^*)$ ), factoring out  $(1-p^*)^k$ , and simplifying, we can rewrite our desired inequality as

$$\forall i \in \{0\} \cup [k], \quad \frac{1}{2}(k-1)r^{i-\frac{k-1}{2}} \geq i. \quad (15)$$

When  $i = \frac{k+1}{2}$  or  $\frac{k-1}{2}$ , we have equality in Equation (15). We extend to the other values of  $i$  by induction. Indeed, when  $i \geq \frac{k+1}{2}$ , then “ $i$  satisfies Equation (15)” implies “ $i+1$  satisfies Equation (15)” because  $ri \geq i+1$ , and when  $i \leq \frac{k-1}{2}$ , then “ $i$  satisfies Equation (15)” implies “ $i-1$  satisfies Equation (15)” because  $\frac{1}{r}i \geq i-1$ .

Similarly, in the case where  $k \geq 2$  is even, we set  $p^* = \frac{1}{2} + \frac{1}{2(k+1)}$  and  $r = \frac{p^*}{1-p^*} = \frac{k+2}{k}$ . In this case, for  $i \in \{0\} \cup [k]$  the following analogue of Equation (15) can be derived:

$$\forall i \in \{0\} \cup [k], \quad \frac{1}{2}kr^{i-\frac{k}{2}} \geq i,$$

and these inequalities follow from the same inductive argument.  $\blacktriangleleft$

**Proof of Lemma 20.** As in the proof of Lemma 17, it suffices to construct  $\mathcal{D}_N^*$  and  $p^*$  such that  $p^*$  maximizes  $\lambda_{\{k-1,k\}}(\mathcal{D}_N^*, \cdot)$  and  $\frac{\lambda_{\{k-1,k\}}(\mathcal{D}_N^*, p^*)}{\gamma_{\{k-1,k\},k}(\mu(\mathcal{D}_N^*))} = \frac{k}{2}\alpha'_{k-1}$ .

We again let  $p^* = \frac{1}{2} + \frac{1}{2(k-1)}$ , but define  $\mathcal{D}_N^*$  by  $\mathcal{D}_N^* \langle \frac{k}{2} \rangle = \frac{(\frac{k}{2})^2}{(\frac{k}{2})^2 + (\frac{k}{2}-1)^2}$  and  $\mathcal{D}_N^* \langle \frac{k}{2} + 1 \rangle = \frac{(\frac{k}{2}-1)^2}{(\frac{k}{2})^2 + (\frac{k}{2}-1)^2}$ . By Equation (13), the derivative of  $\lambda_{\{k-1,k\}}(\mathcal{D}_N^*, \cdot)$  is now

$$\begin{aligned} \frac{d}{dp} &\left[ \frac{(\frac{k}{2})^2}{(\frac{k}{2})^2 + (\frac{k}{2}-1)^2} (1-p)^{k/2-1} p^{k/2-1} \left( \frac{k}{2}p^2 + pq + \frac{k}{2}q^2 \right) + \right. \\ &\left. \frac{(\frac{k}{2}-1)^2}{(\frac{k}{2})^2 + (\frac{k}{2}-1)^2} (1-p)^{k/2-2} p^{k/2} \left( \left( \frac{k}{2}-1 \right) p^2 + pq + \left( \frac{k}{2}+1 \right) q^2 \right) \right] \\ &= -\frac{1}{8(k^2-2k+2)} (1-p)^{k/2-3} p^{k/2-2} (-k + (2(k-1)p)\xi(p), \end{aligned}$$

where  $\xi(p)$  is the cubic

$$\xi(p) = -8k(k-1)p^3 + 2(k^3 + k^2 + 6k - 12)p^2 - 2(k^3 - 4)p + k^2(k-2).$$

Thus,  $\lambda_{\{k-1,k\}}$ 's critical points on the interval  $[0, 1]$  are  $0, 1, p^*$  and any roots of  $\xi$  in this interval. We claim that  $\xi$  has no additional roots in the interval  $(0, 1)$ . This can be verified directly by calculating roots for  $k = 2, 4$ , so assume WLOG  $k \geq 6$ .



Suppose  $\xi(p) = 0$  for some  $p \in (0, 1)$ , and let  $x = \frac{1}{p} - 1 \in (0, \infty)$ . Then  $p = \frac{1}{1+x}$ ; plugging this in for  $p$  and multiplying through by  $(x+1)^3$  gives the new cubic

$$(k^3 - 2k^2)x^3 + (k^3 - 6k^2 + 8)x^2 + (k^3 - 4k^2 + 12k - 8)x + (k^3 - 8k^2 + 20k - 16) = 0 \quad (16)$$

whose coefficients are cubic in  $k$ . It can be verified by calculating the roots of each coefficient of  $x$  in Equation (16) that all coefficients are positive for  $k \geq 6$ . Thus, Equation (16) cannot have roots for positive  $x$ , a contradiction. Hence  $\lambda_{\{k-1, k\}}(\mathcal{D}_N^*, \cdot)$  is maximized at  $p^*$ . Finally, it can be verified that  $\frac{\lambda_{\{k-1, k\}}(\mathcal{D}_N^*, p^*)}{\gamma_{\{k-1, k\}, k}(\mu(\mathcal{D}_N^*))} = \frac{k}{2} \alpha'_{k-1}$ , as desired.  $\blacktriangleleft$

**Proof of Lemma 21.** Define  $p^* = \frac{1}{2} + \frac{1}{2(k-1)}$ . Following the proof of Lemma 18 and using the lower bound  $\gamma_{\{k-1, k\}, k}(\mu(\mathcal{D}_N)) \leq \sum_{i=0}^k \frac{i}{k-1} \mathcal{D}_N\langle i \rangle$ , it suffices to show that

$$\frac{k}{2} \alpha'_{k-1} \leq \inf_{\mathcal{D}_N \in \Delta_k} \frac{\sum_{i=0}^k (1-p^*)^{k-i-1} (p^*)^{i-1} ((k-i)(p^*)^2 + p^*(1-p^*) + i(1-p^*)^2) \mathcal{D}_N\langle i \rangle}{\sum_{i=0}^k \frac{i}{k-1} \mathcal{D}_N\langle i \rangle}$$

for which by Proposition 19, it in turn suffices to prove that for each  $i \in \{0\} \cup [k]$ ,

$$\frac{k}{2} \alpha'_{k-1} \frac{i}{k-1} \leq (1-p^*)^{k-i-1} (p^*)^{i-1} ((k-i)(p^*)^2 + p^*(1-p^*) + i(1-p^*)^2).$$

We again observe that  $\alpha'_{k-1} = (1-p^*)^{k/2-1} (p^*)^{k/2-1}$ , define  $r = \frac{p^*}{1-p^*} = \frac{k}{k-2}$ , and factor out  $(1-p^*)^{k-1}$ , which simplifies our desired inequality to

$$\frac{1}{2} r^{i-\frac{k}{2}-1} \cdot \frac{k-2}{k-1} (i+r+(k-i)r^2) \geq i. \quad (17)$$

for each  $i \in \{0\} \cup [k]$ . Again, we assume  $k \geq 6$  WLOG; the bases cases  $i = \frac{k}{2} - 1, \frac{k}{2}$  can be verified directly, and we proceed by induction. If Equation (17) holds for  $i$ , and we seek to prove it for  $i+1$ , it suffices to cross-multiply and instead prove the inequality

$$r(i+1+r+(k-(i+1))r^2)i \geq (i+1)(i+r+(k-i)r^2),$$

which simplifies to

$$(k-2i)(k-1)(k^2-4i-4) \leq 0,$$

which holds whenever  $\frac{k}{2} \leq i \leq \frac{k^2-4}{4}$  (and  $\frac{k^2-4}{4} \geq k$  for all  $k \geq 6$ ). The other direction (where  $i \leq \frac{k}{2} - 1$  and we induct downwards) is similar.  $\blacktriangleleft$

**Proof of Theorem 6.** To get an  $(\alpha - \epsilon)$ -approximation to  $\text{val}_\Psi$ , let  $\delta > 0$  be small enough such that  $\frac{1-\delta}{1+\delta} \alpha(\text{Th}_k^i) \geq \alpha(\text{Th}_k^i) - \epsilon$ . We claim that calculating an estimate  $\hat{b}$  for  $\text{bias}(\Psi)$  (using Corollary 31) up to a multiplicative  $\delta$  factor and outputting  $\hat{v} = \alpha(\text{Th}_k^i) \gamma_{S,k}(\frac{\hat{b}}{1+\delta})$  is sufficient.

Indeed, suppose  $\hat{b} \in [(1-\delta)\text{bias}(\Psi), (1+\delta)\text{bias}(\Psi)]$ ; then  $\frac{\hat{b}}{1+\delta} \in [\frac{1-\delta}{1+\delta}\text{bias}(\Psi), \text{bias}(\Psi)]$ . Now we observe

$$\begin{aligned} \gamma_{S,k} \left( \frac{\hat{b}}{1+\delta} \right) &\geq \gamma_{S,k} \left( \frac{1-\delta}{1+\delta} \text{bias}(\Psi) \right) && \text{(monotonicity of } \gamma_{S,k} \text{)} \\ &= \min \left\{ \frac{1 + \frac{1-\delta}{1+\delta} \text{bias}(\Psi)}{1 + \epsilon_{S,k}}, 1 \right\} && \text{(Lemma 16)} \\ &\geq \frac{1-\delta}{1+\delta} \min \left\{ \frac{1 + \text{bias}(\Psi)}{1 + \epsilon_{S,k}}, 1 \right\} && (\delta > 0) \\ &= \frac{1-\delta}{1+\delta} \gamma_{S,k}(\text{bias}(\Psi)). && \text{(Lemma 16)} \end{aligned}$$



## 38:22 On Sketching Approximations for Symmetric Boolean CSPs

Then we conclude

$$\begin{aligned}
(\alpha(\text{Th}_k^i) - \epsilon) \text{val}_\Psi &\leq (\alpha(\text{Th}_k^i) - \epsilon) \gamma_{S,k}(\text{bias}(\Psi)) && \text{(Lemma 28)} \\
&\leq \alpha(\text{Th}_k^i) \cdot \frac{1 - \delta}{1 + \delta} \gamma_{S,k}(\text{bias}(\Psi)) && \text{(assumption on } \delta) \\
&\leq \widehat{v} && \text{(our observation)} \\
&\leq \alpha(\text{Th}_k^i) \gamma_{S,k}(\text{bias}(\Psi)) && \text{(monotonicity of } \gamma_{S,k}) \\
&\leq \beta_{S,k}(\text{bias}(\Psi)) && \text{(Equation (7))} \\
&\leq \text{val}_\Psi, && \text{(Lemma 29)}
\end{aligned}$$

as desired.  $\blacktriangleleft$

**Proof of Theorem 12.** By Lemma 26,  $\frac{\beta_{\{3\}}(\mathcal{D}_N)}{\gamma_{\{3\},3}(\mu(\mathcal{D}_N))}$  is minimized *uniquely* at  $\mathcal{D}_N^* = (0, 0, 1, 0)$ . By Lemma 14 we have  $\mu(\mathcal{D}_N^*) = \frac{1}{3}$ , and by inspection from the proof of Lemma 16 below,  $\gamma_{\{3\}}(\mathcal{D}_Y)$  with  $\mu(\mathcal{D}_Y) = \frac{1}{3}$  is uniquely minimized by  $\mathcal{D}_Y^* = (\frac{1}{3}, 0, 0, \frac{2}{3})$ .

Finally, we rule out the possibility of an infinite sequence of padded one-wise pairs which achieve ratios arbitrarily close to  $\frac{2}{9}$  using topological properties. View a distribution  $\mathcal{D} \in \Delta_3$  as the vector  $(\mathcal{D}\langle 0 \rangle, \mathcal{D}\langle 1 \rangle, \mathcal{D}\langle 2 \rangle, \mathcal{D}\langle 3 \rangle) \in \mathbb{R}^4$ . Let  $D \subset \mathbb{R}^4$  denote the set of such distributions. Let  $M \subset D \times D \subset \mathbb{R}^8$  denote the subset of pairs of distributions with matching marginals, and let  $M' \subset M$  denote the subset of pairs with uniform marginals and  $P \subset M$  the subset of padded one-wise pairs.  $D$ ,  $M$ ,  $M'$ , and  $P$  are compact (under the Euclidean topology); indeed,  $D$ ,  $M$ , and  $M'$  are bounded and defined by a finite collection of linear equalities and strict inequalities, and letting  $M' \subset M$  denote the subset of pairs of distributions with matching *uniform* marginals,  $P$  is the image of the compact set  $[0, 1] \times D \times M' \subset \mathbb{R}^{13}$  under the continuous map  $\tau \times \mathcal{D}_0 \times (\mathcal{D}'_Y, \mathcal{D}'_N) \mapsto (\tau \mathcal{D}_0 + (1 - \tau) \mathcal{D}'_Y, \tau \mathcal{D}_0 + (1 - \tau) \mathcal{D}'_N)$ . Hence,  $P$  is closed.

Now the function

$$\alpha : M \rightarrow \mathbb{R} \cup \{\infty\} : (\mathcal{D}_N, \mathcal{D}_Y) \mapsto \frac{\beta_{\{3\}}(\mathcal{D}_N)}{\gamma_{\{3\}}(\mathcal{D}_Y)}$$

is continuous, since a ratio of continuous functions is continuous, and  $\beta_{\{3\}}$  is a single-variable supremum of a continuous function (i.e.,  $\lambda_S$ ) over a compact interval, which is in general continuous in the remaining variables. Thus, if there were a sequence of padded one-wise pairs  $\{(\mathcal{D}_N^{(i)}, \mathcal{D}_Y^{(i)}) \in P\}_{i \in \mathbb{N}}$  such that  $\alpha(\mathcal{D}_N^{(i)}, \mathcal{D}_Y^{(i)})$  converges to  $\frac{2}{9}$  as  $i \rightarrow \infty$ , since  $M$  is compact and  $P$  is closed, Lemmas 26 and 27 imply that  $(\mathcal{D}_N^*, \mathcal{D}_Y^*) \in P$ , a contradiction.  $\blacktriangleleft$

**Proof of Lemma 28.** Let  $\mathbf{opt} \in \{-1, 1\}^n$  denote the optimal assignment for  $\Psi$ . Then

$$\begin{aligned}
\text{val}_\Psi &= \text{val}_\Psi(\mathbf{opt}) && \text{(definition of } \mathbf{opt}) \\
&= \text{val}_{\Psi^{\mathbf{opt}}}(\mathbf{1}^n) && \text{(Item iii of Proposition 32)} \\
&= \lambda_S(\mathcal{D}_{\Psi^{\mathbf{opt}}}^{\text{sym}}, 1) && \text{(Item i of Proposition 33 with } p = 1) \\
&= \gamma_S(\mathcal{D}_{\Psi^{\mathbf{opt}}}^{\text{sym}}) && \text{(definition of } \gamma_S, \text{ Equation (4))} \\
&\leq \gamma_{S,k}(\mu(\mathcal{D}_{\Psi^{\mathbf{opt}}}^{\text{sym}})) && \text{(definition of } \gamma_{S,k}, \text{ Equation (6))} \\
&\leq \gamma_{S,k}(\text{bias}(\Psi^{\mathbf{opt}})) && \text{(Item ii of Proposition 33 and monotonicity of } \gamma_{S,k}) \\
&= \gamma_{S,k}(\text{bias}(\Psi)), && \text{(Item ii of Proposition 32)}
\end{aligned}$$

as desired.  $\blacktriangleleft$

**Proof of Lemma 29.** Let  $\mathbf{maj} \in \{-1, 1\}^n$  denote the assignment assigning  $x_i$  to 1 if  $\text{diff}_i(\Psi) \geq 0$  and  $-1$  otherwise. Now

$$\begin{aligned}
\text{val}_\Psi &= \text{val}_{\Psi^{\mathbf{maj}}} && \text{(Item iv of Proposition 32)} \\
&\geq \sup_{p \in [0,1]} \left( \mathbb{E}_{\mathbf{a} \sim \text{Bern}(p)^n} [\text{val}_{\Psi^{\mathbf{maj}}}(\mathbf{a})] \right) && \text{(probabilistic method)} \\
&= \sup_{p \in [0,1]} (\lambda_S(\mathcal{D}_{\Psi^{\mathbf{maj}}}^{\text{sym}}, p)) && \text{(Item i of Proposition 33)} \\
&\geq \beta_S(\mathcal{D}_{\Psi^{\mathbf{maj}}}^{\text{sym}}) && \text{(definition of } \beta_S, \text{ Equation (4))} \\
&\geq \beta_{S,k}(\mu(\mathcal{D}_{\Psi^{\mathbf{maj}}}^{\text{sym}})) && \text{(definition of } \beta_{S,k}, \text{ Equation (6))} \\
&= \beta_{S,k}(\text{bias}(\Psi^{\mathbf{maj}})) && \text{(Item iii of Proposition 33)} \\
&= \beta_{S,k}(\text{bias}(\Psi)), && \text{(Item ii of Proposition 32)}
\end{aligned}$$

as desired.  $\blacktriangleleft$



# Massively Parallel Algorithms for Small Subgraph Counting

Amartya Shankha Biswas ✉

CSAIL, MIT, Cambridge, MA, USA

Talya Eden ✉

CSAIL, MIT, Cambridge MA, USA

Boston University, MA, USA

Quanquan C. Liu ✉

Northwestern University, Evanston, IL, USA

Ronitt Rubinfeld ✉

CSAIL, MIT, Cambridge, MA, USA

Slobodan Mitrović ✉

University of California Davis, CA, USA

---

## Abstract

Over the last two decades, frameworks for distributed-memory parallel computation, such as MapReduce, Hadoop, Spark and Dryad, have gained significant popularity with the growing prevalence of large network datasets. The Massively Parallel Computation (MPC) model is the de-facto standard for studying graph algorithms in these frameworks theoretically. Subgraph counting is one such fundamental problem in analyzing massive graphs, with the main algorithmic challenges centering on designing methods which are both scalable and accurate.

Given a graph  $G = (V, E)$  with  $n$  vertices,  $m$  edges and  $T$  triangles, our first result is an algorithm that outputs a  $(1 + \varepsilon)$ -approximation to  $T$ , with asymptotically *optimal round and total space complexity* provided any  $S \geq \max(\sqrt{m}, n^2/m)$  space per machine and assuming  $T = \Omega(\sqrt{m/n})$ . Our result gives a quadratic improvement on the bound on  $T$  over previous works. We also provide a simple extension of our result to counting *any* subgraph of  $k$  size for constant  $k \geq 1$ . Our second result is an  $O_\delta(\log \log n)$ -round algorithm for exactly counting the number of triangles, whose total space usage is parametrized by the *arboricity*  $\alpha$  of the input graph. We extend this result to exactly counting  $k$ -cliques for any constant  $k$ . Finally, we prove that a recent result of Bera, Pashanasangi and Seshadhri (ITCS 2020) for exactly counting all subgraphs of size at most 5 can be implemented in the MPC model in  $\tilde{O}_\delta(\sqrt{\log n})$  rounds,  $O(n^\delta)$  space per machine and  $O(m\alpha^3)$  total space.

In addition to our theoretical results, we simulate our triangle counting algorithms in real-world graphs obtained from the Stanford Network Analysis Project (SNAP) database. Our results show that both our approximate and exact counting algorithms exhibit improvements in terms of round complexity and approximation ratio, respectively, compared to two previous widely used algorithms for these problems.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Graph algorithms analysis; Computing methodologies  $\rightarrow$  Massively parallel algorithms

**Keywords and phrases** triangle counting, massively parallel computation, clique counting, approximation algorithms, subgraph counting

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.39

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2002.08299>

**Funding** *Slobodan Mitrović*: S. Mitrović was supported by the Swiss NSF grant No. P400P2\_191122/1 and FinTech@CSAIL.



© Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Ronitt Rubinfeld, and Slobodan Mitrović; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 39; pp. 39:1–39:28



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Estimating the number of small subgraphs, cliques in particular, is a fundamental problem in computer science, and has been extensively studied both theoretically and from an applied perspective. Given its importance, the task of counting subgraphs has been explored in various computational settings, e.g., sequential [7, 91, 28], distributed and parallel [89, 78, 68, 80, 72], streaming [16, 66, 24, 76], and sublinear-time [44, 5, 13, 45]. There are usually two perspectives from which subgraph counting is studied: first, optimizing the running time (especially relevant in the sequential and sublinear-time settings) and, second, optimizing the space or query requirement (relevant in the streaming, parallel, and distributed settings). In each of these perspectives, there are two, somewhat orthogonal, directions that one can take. The first is *exact* counting. However, in most scenarios, algorithms that perform exact counting are prohibitive, e.g., they require too much space or too many parallel rounds to be implementable in practice.

Hence, the second direction of obtaining an *estimate/approximation* on the number of small subgraphs is both an interesting theoretical problem and of practical importance. If  $H_{\#}$  is the number of subgraphs isomorphic to  $H$ , the main question in approximate counting is whether we can design algorithms that, under given resource constraints, provide approximations that concentrate well. This concentration is usually parametrized by  $H_{\#}$  (and potentially some other parameters). In particular, most known results do not provide a strong approximation guarantee when  $H_{\#}$  is very small, e.g.,  $|H_{\#}| = O(1)$ . So, the main attempts in this line of work is to provide an estimation that concentrates well while imposing as small a lower bound on  $H_{\#}$  as possible.

Due to ever increasing sizes of data stores, there has been an increasing interest in designing scalable algorithms. The *Massively Parallel Computation* (MPC) model is a theoretical abstraction of popular frameworks for large-scale computation such as MapReduce [41], Hadoop [93], Spark [95] and Dryad [62]. MPC gained significant interest recently, most prominently in building algorithmic toolkits for graph processing [57, 74, 17, 8, 18, 59, 4, 83, 61, 38, 11, 12, 51, 58, 30, 14, 29, 21, 19, 23, 9, 15, 53, 50, 55, 71, 63, 34, 52, 54]. Efficiency of an algorithm in MPC is characterized by three parameters: round complexity, the space per machine in the system, and the number of machines/total memory used. Our work aims to design efficient algorithms with respect to all three parameters and is guided by the following question:

*How does one design efficient massively parallel algorithms for small subgraph counting?*

### 1.1 The MPC Model

In this paper, we are working in the Massively Parallel Computation (MPC) model introduced by [67, 57, 17]. The model operates as follows. There exist  $\mathcal{M}$  machines that communicate with each other in synchronous rounds. The graph input is initially distributed across the machines in some organized way such that machines know how to access the relevant information via communication with other machines. During each round, the machines first perform computation locally without communicating with other machines. The computation done locally can be unbounded (although the machines have limited space so any reasonable program will not do an absurdly large amount of computation). At the end of the round, the machines exchange messages to inform the computation for the next round. The total size of all messages that can be received by a machine is upper bounded by the size of its local memory, and each machine outputs messages of sufficiently small size that can fit into

its memory. If  $N$  is the total size of the data and each machine has  $S$  words of space, we are interested in the settings when  $S$  is sublinear in  $N$ . We use *total space* to refer to  $\mathcal{M} \cdot S$ , which is the total space that is available across all the machines.

## 1.2 Our Contributions

■ **Table 1** Summary of our main MPC triangle counting results compared to previous work. Our results are **bolded**. “ALB” refers to the approximation lower bound on the number of triangles required to obtain a  $(1 + \varepsilon)$ -approximation, with high probability.  $\alpha$  is the arboricity of the input graph and is generally small (logarithmic) in real-world networks. Parameter  $\delta > 0$  is *any* constant.

| Problem                       | Work        | MPC Rounds                                | Space Per Machine                | Total Space                      | ALB                                        |
|-------------------------------|-------------|-------------------------------------------|----------------------------------|----------------------------------|--------------------------------------------|
| Exact Triangle Counting       | [89]        | 2                                         | $O(\sqrt{m})$                    | $O(m^{3/2})$                     | -                                          |
|                               | [89]        | 1                                         | $o(m)$                           | $\omega(m)$                      | -                                          |
|                               | [36]        | $O(n)$                                    | $O(n)$                           | $O(m)$                           | -                                          |
|                               | folklore    | $O(\log n)$                               | $\Omega(\alpha^2)$               | $O(m\alpha)$                     | -                                          |
|                               | <b>Ours</b> | <b><math>O_\delta(\log \log n)</math></b> | <b><math>O(n^\delta)</math></b>  | <b><math>O(m\alpha)</math></b>   | -                                          |
| Approximate Triangle Counting | [78]        | $O(1)$                                    | $\Omega(m)$                      | $O(m)$                           | $\Omega(d_{avg})$                          |
|                               | [85]        | $O(1)$                                    | $O(n^\delta)$                    | $O(m)$                           | $\Omega(\sum_{v \in V} \deg(v)^2)$         |
|                               | <b>Ours</b> | <b><math>O(1)</math></b>                  | <b><math>\tilde{O}(n)</math></b> | <b><math>\tilde{O}(m)</math></b> | <b><math>\Omega(\sqrt{d_{avg}})</math></b> |

### 1.2.1 Triangle Counting

We provide a number of results for triangle counting in both the approximate and exact settings. Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges and  $T$  triangles. First we study the question of approximately counting the number of triangles under the restriction that the round and total space complexities are essentially *optimal*, i.e.,  $O(1)$  and  $\tilde{O}(m)$ , where  $\tilde{O}$  hides  $O(\text{poly log } n)$  factors, respectively. Here and throughout, we use  $O_\delta$  and  $O_\varepsilon$  to hide factors of  $\delta$  and  $\varepsilon$ , respectively, where we consider constant factors of  $\delta, \varepsilon > 0$  in this paper.

Our algorithm is surprisingly simple with a more complicated analysis, but improves on the previous best-known result by giving a  $(1 + \varepsilon)$ -approximation, with high probability, while achieving a *quadratic* improvement on the number of triangles required to ensure this approximation. The specific bounds are given in Table 1.

► **Theorem 1.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges, and let  $T$  be the number of triangles in  $G$ . Assuming*

$$(i) \ T = \tilde{\Omega}\left(\sqrt{\frac{m}{S}}\right), \quad (ii) \ S = \tilde{\Omega}\left(\max\left\{\frac{\sqrt{m}}{\varepsilon}, \frac{n^2}{m}\right\}\right),$$

*there exists an MPC algorithm, using  $\mathcal{M}$  machines, each with local space  $S$ , and total space  $\mathcal{M}S = \tilde{O}_\varepsilon(m)$ , that outputs a  $(1 \pm \varepsilon)$ -approximation of  $T$ , with high probability, in  $O(1)$  rounds.*

For  $S = \Theta(n \log n)$  (specifically,  $S > 100n \log n$ ) in Theorem 1, we derive the following corollary.

► **Corollary 2.** *Let  $G$  be a graph and  $T$  be the number of triangles it contains. If  $T \geq \sqrt{d_{avg}}$ , then there exists an MPC algorithm that in  $O(1)$  rounds with high probability outputs a  $(1 + \varepsilon)$ -approximation of  $T$ . This algorithm uses a total space of  $\tilde{O}(m)$  and space  $\tilde{O}(n)$  per machine.  $d_{avg}$  is the average degree of the vertices in the graph.*

There is a long line of work on computing approximate triangle counting in parallel computation [37, 90, 89, 94, 78, 69, 79, 85, 10, 80, 68, 64, 42] and references therein. Despite this progress, and to the best of our knowledge, on one hand, each MPC algorithm for exact triangle counting either requires strictly super-polynomial in  $m$  total space, or the number of rounds is super-constant (as seen in Table 1). On the other hand, the best-known, classic algorithm for approximate triangle counting by Pagh and Tsourakakis [78] requires  $T \geq d_{avg}$  even when the space per machine is  $\Theta(n)$ . We design an algorithm that has essentially optimal total space and round complexity, while at least quadratically improving the requirement on  $T$ .

Furthermore, since the amount of messages sent and received by each machine is bounded by  $O(n)$ , by [20], our algorithm directly implies an  $O(1)$ -rounds algorithm in the CONGESTED-CLIQUE model<sup>1</sup> under the same restriction  $T = \Omega(\sqrt{m/n})$ . The best known (to our knowledge) triangle approximation algorithm for general graphs in this model, is an  $O(n^{1/3}/T^{2/3})$ -rounds algorithm by [43]. The best-known previous bound only results in constant round complexity when  $T = \Omega(\sqrt{n})$ .

► **Corollary 3.** *Given a graph  $G = (V, E)$  with  $T$  triangles, if  $T = \Omega(\sqrt{m/n})$ , then there exists a  $O(1)$ -rounds algorithm in the CONGESTED-CLIQUE model that gives a  $(1 + \varepsilon)$ -approximation of  $T$  with high probability.*

The second question we consider is the question of exact counting, for which we present an algorithm whose total space depends on the arboricity of the input graph. The arboricity of a graph (roughly) equals the average degree of its densest subgraph. The class of graphs with bounded arboricity includes many important graph families such as planar graphs, bounded degree graphs and randomly generated preferential attachment graphs. In addition, many real-world graphs exhibit bounded arboricity [56, 48, 87], making this property important also in practical settings. For many problems, a bound on the arboricity of the graph allows for much more efficient algorithms and/or better approximation ratios [6, 48].

Specifically for the task of subgraph counting, in a seminal paper, Chiba and Nishizeki [35] prove that triangle enumeration can be performed in  $O(m\alpha)$  time, and assuming 3SUM-hardness this result is optimal up to dependencies in  $O(\text{poly log } n)$  [81, 70]. Many applied algorithms also rely on the property of having bounded arboricity in order to achieve better space and time bounds, e.g., [84, 36, 73]. Our main theorem with respect to this question is the following.

► **Theorem 4.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges and arboricity  $\alpha$ . COUNT-TRIANGLES( $G$ ) takes  $O_\delta(\log \log n)$  rounds,  $O(n^\delta)$  space per machine for any  $\delta > 0$ , and  $O(m\alpha)$  total space.*

It is interesting to note that our total space complexity matches the time complexity (both upper and conditional lower bounds) of combinatorial<sup>2</sup> triangle counting algorithms in the sequential model [35, 81, 70]. The best-known previous algorithm in this setting is the folklore algorithm of placing each vertex and its out-neighbors in the same machine and counting the incident triangles. Such an approach requires  $O(\log n)$  rounds and  $\Omega(\alpha^2)$  space per machine (summarized in Table 1). We prove the above theorem in Section 4.

<sup>1</sup> A distributed model where nodes communicate with each other over a complete network using  $O(\log n)$  bit messages [75].

<sup>2</sup> Combinatorial algorithms, usually, refer to algorithms that do not rely on fast matrix multiplication.



### 1.2.2 Clique Counting

All of our above triangle counting results can be extended to  $k$ -clique counting. In our full paper [27], we prove that our exact triangle counting result can be extended to exactly counting  $k$ -cliques for any constant  $k$ :

► **Theorem 5.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges and arboricity  $\alpha$ . COUNT-CLIQUEs( $G$ ) takes  $O_\delta(\log \log n)$  rounds,  $O(n^\delta)$  space per machine for any  $\delta > 0$ , and  $O(m\alpha^{k-2})$  total space.*

We can improve on the total space usage if we are given machines where the memory for each individual machine satisfies  $\alpha < n^{\delta'/2}$  where  $\delta' < \delta$ . In this case, we obtain an algorithm that counts the number of  $k$ -cliques in  $G$  using  $O(n\alpha^2)$  total space and  $O_\delta(\log \log n)$  communication rounds.

Furthermore, our approximate triangle counting results can be extended to counting *any* subgraph of size  $K$  where  $K$  is constant. Specifically, we obtain the following result:

► **Theorem 6.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges, and let  $B$  be the number of occurrences of a subgraph  $H$  with  $K$  vertices in  $G$ . If  $B \geq d_{\text{avg}}^{K/2-1}$ , then there exists an MPC algorithm that gives a  $(1 + \epsilon)$ -approximation of  $B$  in  $O(1)$  rounds, total space  $\tilde{O}(m)$ , and  $\tilde{O}(n)$  space per machine, with high probability. Here,  $d_{\text{avg}}$  is the average degree of the vertices in the graph.*

## 1.3 Other Small Subgraphs

Finally, we consider the problem of exactly counting subgraphs of size at most 5, and show that the recent result of Bera, Pashanasangi and Seshadhri [25] for this question in the sequential model, can be implemented in the MPC model. Ours is the first result for counting any arbitrary subgraph of size at most 5 in  $\text{poly}(\log n)$  rounds in the MPC model. Here too, our total space complexity matches the time complexity of the sequential model algorithm. It is an interesting open question whether our results can be extended to more general subgraphs following the results of [32, 26]. Section 6 summarizes the difficulties of implementing these algorithms in the MPC model and we present this question as interesting future work.

► **Theorem 7.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges, and arboricity  $\alpha$ . The algorithm of BPS for counting the number of occurrences of a subgraph  $H$  over  $k \leq 5$  vertices in  $G$  can be implemented in the MPC model in  $O_\delta(\sqrt{\log n} \log \log m)$  rounds, with high probability. The space requirement per machine is  $O(n^{2\delta})$  and the total space is  $O(m\alpha^3)$ .*

## 1.4 Related Work

There has been a long line of work on small subgraph counting in massive networks in the MapReduce model whose results translate to the MPC model. We first describe the works for *exact* triangle and  $k$ -clique counting. [89] first designed an algorithm for triangle counting, but their approach requires a super-linear total space of  $O(m^{3/2})$ . Another work, [2], shows how to count small subgraphs by using  $b^3$  machines, each requiring  $O(m/b^2)$  space per machine. Hence, it uses a total space of  $O(mb)$ . Therefore, this approach either requires super-linear total space or almost  $O(m)$  space per machine. [89] were the first to achieve constant number of rounds in MPC, where they design two algorithms. The first of those algorithms, that runs in 2 rounds, requires  $O(\sqrt{m})$  space per machine and total space  $O(m^{3/2})$ . Their second algorithm requires only one round for exact triangle counting,

total space  $O(\rho m)$  and space per machine  $O(m/\rho^2)$ . Therefore, for this algorithm to work with polynomially less than space  $m$  per machine, it has to allow for a total space that is polynomially larger than  $m$ . [36] focus on algorithms that require a total space of  $O(m)$ . In the worst case, their algorithm performs  $O(|E|/S)$  MPC rounds to output the exact count where  $S$  is the maximum space per machine. [49] extended and provided new algorithms for clique counting but they also require  $\Omega(m^{3/2})$  total space.

[90, 10] designed randomized algorithms for *approximate* triangle counting also in the MapReduce model (whose results, again, can be translated rather straightforwardly to the MPC model). Their approach first sparsifies the input graph by sampling a subset of edges, and executes some of the known algorithms for triangle counting on the sampled subgraph. Denoting their sampling probability by  $p$ , their approach outputs a  $(1 + \varepsilon)$ -approximate triangle count with probability at most  $1 - 1/(\varepsilon^2 p^3 T)$ .<sup>3</sup> To contrast this result with our approach, consider a graph  $G$  where  $m = \Theta(n^2)$ . Let  $G'$  be the edge-sparsified graph as explained above. To be able to execute the first algorithm of [89] on  $G'$  such that the total space requirement is  $O(m)$ , one can verify that it is needed to set  $p = \Theta(n^{-2/3})$ . This in turn implies that the result in [90, 10] outputs the correct approximation with constant probability only if  $T = \Omega(n^2)$ . An improved lower-bound can be obtained by using the second algorithm of [89]. By balancing out  $\rho$  and  $p$  and for  $S = O(n)$ , one can show that the sparsification results in a constant probability of success for  $T = \Omega(n)$ . On the other hand, for  $S = O(n)$ , our approach obtains the same guarantee even when  $T = \Theta(\sqrt{d_{avg}(G)}) = \Theta(\sqrt{n})$ .

The best-known algorithm of [78] is a randomized algorithm for approximate triangle counting based on graph partitioning. The graph is partitioned into  $1/p$  pieces, where  $p$  is at least the ratio of the maximum number of triangles sharing an edge and  $T$ . When all the triangles share one edge, then  $p \geq 1$ , and hence such an approach would require the space per machine to be  $\Omega(m)$ . Furthermore, this approach requires the number of triangles to be lower bounded by  $T = \Omega(d_{avg})$ . Another more recent work of [85] uses wedge sampling and provides a  $(1 + \varepsilon)$ -approximation of the triangle count in  $O(1)$  rounds when  $T$  is a constant fraction of the sum of squares of degrees. The comparison of our bounds with these previous results are summarized in Table 1.

**Other related work.** Subgraph counting (primarily triangles) was also extensively studied in the streaming model, see [16, 66, 31, 65, 76, 24, 13] and references therein. This culminated in a result that requires space  $\tilde{O}(m^{3/2}/(T\varepsilon^2))$  to estimate the number of triangles within a  $(1 + \varepsilon)$ -factor. In the semi-streaming setting it is assumed that one has  $\tilde{O}(n)$  space at their disposal. This result fits in this regime for  $T \geq m^{3/2}/n = d_{avg} \cdot m^{1/2}$ . As a reminder, our MPC result requires  $T \geq \sqrt{d_{avg}}$  when  $S = \tilde{O}(n)$ .

In a celebrated result, [7] designed an algorithm for triangle counting in the sequential settings that runs in  $O(m^{2\omega/(\omega+1)})$  time, where  $\omega$  is the best-known exponent of matrix multiplication. Since then, several important works have extended this result to  $k$ -clique counting [46, 91]. In the work-depth (shared-memory parallel processors) model, several results are known for this problem. There has been significant work on practical parallel algorithms for the case of triangle counting (e.g. [10, 89, 79, 80, 88] among others). There is even an annual competition for parallel triangle counting algorithms [1]. For counting  $k = 4$  and  $k = 5$  cliques, efficient practical solutions have also been developed [3, 40, 47, 60, 82]. [39] recently implemented the Chiba-Nishizeki algorithm [35] for  $k$ -cliques in the parallel setting; although, their work does not achieve polylogarithmic depth. Even more recently, [86]

<sup>3</sup> The actual probability is even smaller and also depends on pairs of triangles that share an edge.

enumerated  $k$ -cliques in the work-depth model in  $O(m\alpha^{k-2})$  expected work and  $O(\log^{k-2} n)$  depth with high probability, using  $O(m)$  space. Among other distinctions from our setting, the work-depth model assumes a shared, common memory.

In the CONGESTED-CLIQUE model, [33] present an  $\tilde{O}(n^{1-2/\omega}) = \tilde{O}(n^{0.158})$  rounds algorithm for matrix multiplication, implying the same complexity for exact triangle counting. [43] present an algorithm for approximate triangle counting in general graphs whose expected running time is  $O(n^{1/3}/T^{2/3})$ . They also present an  $O(\alpha^2/n)$ -rounds algorithm for bounded arboricity graphs.

## 2 Preliminaries

**Counting Duplicates.** We make use of interval trees for certain parts of our paper to count the number of repeating elements in a sorted list, given bounded space per machine. We use the interval tree implementation given by [57] to obtain our count duplicates algorithm in the MPC model. We prove the following theorem in the MPC model regarding our count duplicates tree implementation. The proofs of the following claims are given in our full paper [27].

► **Theorem 8.** *Given a sorted list of  $N$  elements implemented on processors where the space per processor is  $S$  and the total space among all processors is  $O(N)$ , for each unique element in the list, we can compute the number of times it repeats in  $O(\log_S N)$  communication rounds.*

We also use the following two new MPC primitives in proving our bounds. These primitives may be of use in other algorithms beyond the scope of our paper.

► **Lemma 9.** *Given two sets of tuples  $Q$  and  $C$  (both of which may contain duplicates), for each tuple  $q \in Q$ , we return whether  $q \in C$  in  $O(|Q \cup C|)$  total space and  $O_\delta(1)$  rounds given machines with space  $O(n^\delta)$  for any  $\delta > 0$ .*

► **Lemma 10.** *Given a machine  $M$  that has space  $O(n^{2\delta})$  for any  $\delta > 0$  and contains data of  $O(n^\delta)$  words, we can generate  $x$  copies of  $M$ , each holding the same data as  $M$ , using  $O(M \cdot x)$  machines with  $O(n^\delta)$  space each in  $O(\log_{n^\delta} x)$  rounds.*

## 3 Overview of Our Techniques

### 3.1 Exact Triangle Counting

Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges and arboricity at most  $\alpha$ . We tackle the task of exactly counting the number of triangles in  $G$  in  $O_\delta(\log \log n)$  rounds using the following ideas. In each round  $i$ , we partition the vertices into low-degree vertices  $A_i$  and high-degree vertices, according to a degree threshold  $\gamma_i$ , which grows doubly exponentially in the number of rounds. We then count the number of triangles incident to the set of low degree vertices  $A_i$ . Each low-degree vertex  $v \in A_i$  sends a list of its neighbors to all its neighbors. Then, any neighbor  $u$  of  $v$  that detects a common neighbor  $w$  to  $u$  and  $v$ , adds the triangle  $(u, v, w)$  to the list of discovered triangles.

Once all triangles incident to the vertices in  $A_i$  are processed, we remove this set from the graph and continue with the now smaller graph. This removal of the already processed vertices allows us to handle larger and larger degrees from step to step while using a total space of  $O(m\alpha)$ . This behavior also leads to the  $O_\delta(\log \log n)$  round complexity, as after

this many rounds all vertices are processed. The key insight in our proof that we maintain  $O(m\alpha)$  total space *even when we increase the degree threshold doubly exponentially*. Such insight allows us to obtain our improved number of rounds while maintaining the same total space as the previous folklore algorithm. Finally, we achieve improved space per machine to  $O(n^\delta)$  for any constant  $\delta > 0$  via a number of new MPC primitives. Our algorithm and its analysis are provided in Section 4.

### 3.2 Approximate Triangle Counting

Our work reduces approximate triangle counting to exact triangle counting in multiple (randomly chosen) induced subgraphs of the original graph. In our work, and in contrast to prior approaches (e.g., [78]), the induced subgraphs on different machines might *overlap* in both vertices and edges. This enables us to obtain better concentration bounds compared to prior work, but also brings many challenges. Surprisingly, our algorithm is very simple (with a more complicated analysis), but is able to achieve a better lower bound on the number of triangles required to achieve a  $(1 + \varepsilon)$ -approximation with high probability.

The high level idea is that each machine  $M_i$  samples a subset of vertices  $V_i$  by including each vertex in  $V_i$  with probability  $\hat{p}$ . Then, each machine computes the induced subgraph  $G[V_i]$  and the number of triangles in that subgraph. The total number of triangles seen across all the machines is used as an estimator. We repeat in parallel this sampling process  $O(\log n)$  times and return the median of the estimates. The main challenge this approach raises is: *How do we efficiently collect overlapping induced subgraphs?* (Indeed, approximate triangle counting, even when the number of triangles is  $O(1)$ , can be reduced to counting the number of edges in *sparse* induced subgraphs with the total size of subgraphs being  $\tilde{O}(m)$ .) We now describe how to handle this task in our case.

**Computing induced overlapping subgraphs.** It is unclear how to compute the induced subgraph on each machine in  $O(1)$  rounds without exceeding the total allowed space of  $\tilde{O}(m)$ . This task becomes easier if the subgraphs are disjoint. For example, such an issue is avoided when the graph is *partitioned* across machines as in the algorithm of Pagh and Tsourakakis [78] since there is one copy of each vertex among all the machines. This is not the case for our algorithm.

The trivial strategy of sampling vertices into the machines and querying for all possible edges between any pair of two vertices takes total space at least  $\sum_{i=1}^M X_i^2$  where  $X_i$  is the number of vertices sampled to each machine  $i$ . In general, this approach requires much larger than  $\tilde{O}(m)$  space. We tackle this challenge by using a *globally known* hash function  $h : V \times [\mathcal{M}] \rightarrow \{0, 1\}$ , to indicate whether vertex  $v$  is sampled in the  $i^{\text{th}}$  machine. By requiring that the hash function is known to all machines, we can efficiently compute which edges to send to each machine, i.e., which edges belong to the subgraph  $G[V_i]$ . However, in order for all machines to be able to compute the hash function, the hash function has to use limited space. Hence, we cannot hope for a fully independent function, rather we can only use an  $(S/\log n)$ -wise independent hash function. Still, we manage to show that we are able to handle the dependencies introduced by the hash function, even if we allow as little as  $O(\log n)$ -independence.

### 3.3 Counting $k$ -cliques and 5-subgraphs

We use similar techniques for both problems of exactly counting the number of  $k$ -cliques and of exactly counting subgraphs up to size 5. Our final result is the first MPC algorithm for counting any *arbitrary* subgraph  $H$  of size at most 5 in  $\text{poly}(\log n)$  MPC rounds.

Let  $H$  denote the subgraph of interest. We say that a subgraph that can be mapped to a subset of  $H$  of size  $i$  is a  $i$ -subcopy of  $H$ . Our main contribution in this section is a new MPC procedure that in each round, tries to extend  $i$ -subcopies of  $H$  to  $(i + 1)$ -subcopies of  $H$  by increasing the total space by a factor of at most  $\alpha$ . This is possible by ordering the vertices in  $H$  such that each vertex has at most  $O(\alpha)$  outgoing neighbors so that in each iteration only  $\alpha$  possible extensions should be considered per each previously discovered subcopy.

**Challenges.** The major challenge we face here is dealing with *finding* and *storing* copies of small (constant-sized) subgraphs in individual machines. This is a challenge due to the fact that an entire neighborhood of a vertex  $v$  may not fit on one machine (recall that we have no restrictions on how large the constant  $\delta$  in  $O(n^\delta)$  machine size can be). Thus, we cannot compute all such small subgraphs on one machine. However, if not done carefully, computing small subgraphs across many machines could potentially result in many rounds of computation (since we potentially have to try all combinations of vertices in a neighborhood). We solve this issue by formulating a new MPC procedure (Lemma 10) in which we carefully duplicate neighborhoods of vertices across machines. The detailed analysis of our algorithm is given in our full paper [27].

#### 4 Exact Triangle Counting in $O(m\alpha)$ Total Space

In this section we describe our algorithm for (exactly) counting the number of triangles in graphs  $G = (V, E)$  of arboricity  $\alpha$  and prove Theorem 4, restated here, in Appendix A.1. We first provide an overview of our algorithm and its challenges.

► **Theorem 11.** *Let  $G = (V, E)$  be a graph over  $n$  vertices,  $m$  edges and arboricity  $\alpha$ . COUNT-TRIANGLES( $G$ ) takes  $O_\delta(\log \log n)$  rounds,  $O(n^\delta)$  space per machine for some constant  $0 < \delta < 1$ , and  $O(m\alpha)$  total space.*

Importantly, unlike previous methods, we *do not* need to assume knowledge of the arboricity of the graph  $\alpha$  as input into our algorithm. The arboricity only shows up in our space bound as a property of the graph but we do not need to have knowledge of its value as we run the algorithm. The folklore algorithm shown in Table 1 requires an assumption of an upper bound on  $\alpha$  since in order to achieve  $O(\log n)$  rounds, we must count triangles incident to and remove all vertices with degree less than or equal to  $2\alpha$  in each round. The procedure gets stuck if we remove vertices with degree  $c$  where  $c < \alpha$  in each round because there exists an induced subgraph with degree at least  $\alpha$  in a graph with arboricity  $\alpha$ . One can estimate the arboricity of the graph using  $O(\log n)$  additional rounds or an  $O(\log n)$  additional factor in space. Our algorithm does not require this additional step.

In this section, we assume that individual machines have space  $\Theta(n^\delta)$  where  $\delta$  is some constant  $0 < \delta < 1$ . Given this setting, there are several challenges associated with this problem.

► **Challenge 12.** *The entire subgraph neighborhood of a vertex may not fit on a single machine. This means that all triangles incident to a particular vertex cannot be counted on one machine. Even if we are considering vertices with degree at most  $\alpha$ , it is possible that  $\alpha > n^\delta$ . Thus, we need to have a way to count triangles efficiently when the neighborhood of a vertex is spread across multiple machines.*

The second challenge is to avoid over-counting.

► **Challenge 13.** *When counting triangles across different machines, over-counting the triangles might occur, e.g., if two different machines count the same triangle. We need some way to deal with duplicate counting of the triangles to obtain the exact count of the triangles.*

We deal with the above challenges in our procedures below. We assume in our algorithm that each vertex can access its neighbors in  $O(1)$  rounds of communication; such can be ensured via standard MPC techniques. Let  $d_Q(v)$  be the degree of  $v$  in the subgraph induced by vertex set  $Q$ , i.e. in  $G[Q]$ . Our main algorithm consists of the following COUNT-TRIANGLES( $G$ ) procedure.

■ **Algorithm 1** Count-Triangles( $G = (V, E)$ ).

---

```

1: Let Q_i be the set of vertices not yet processed by iteration i . Initially set $Q_0 \leftarrow V$.
2: Let T be the current count of triangles. Set $T \leftarrow 0$.
3: for $i = 0$ to $i = \lceil \log_{3/2}(\log_2(n)) \rceil$ do
4: $\gamma_i \leftarrow 2^{(3/2)^i}$.
5: Let A_i be the list of vertices $v \in Q_i$ where $d_{Q_i}(v) \leq \gamma_i$. Set $Q_{i+1} \leftarrow Q_i \setminus A_i$.
6: parfor $v \in A_i$ do
7: Retrieve the list of neighbors of v and denote it by L_v .
8: Send each of v 's neighbors a copy of L_v .
9: end parfor
10: parfor $w \in Q_i$ do
11: Let $\mathcal{L}_w = \bigcup_{v \in (N(w) \cap A_i)} L_v$ be the union of neighbor lists received by w .
12: Set $T \leftarrow T + \text{FIND-TRIANGLES}(w, \mathcal{L}_w)$. ▷ Algorithm 2
13: end parfor
14: Return T .
```

---

*Round compression* is a technique formulated by [77, 38] that randomly partitions the vertices in a graph across machines where each machine then stores the induced subgraph induced by the partition. Then, a problem (e.g. maximum matching) is solved locally in each induced subgraph in each machine. The solutions in each machine allows one to remove certain vertices, reducing the degree of the remaining graph. In each round compression step, the maximum degree of the graph drops by a polynomial factor. This degree reduction then allows for more aggressive sampling in the next round compression step. This leads to  $O(\log \log \Delta)$  round compression steps until the maximum degree is  $\text{poly}(\log n)$ ; in this case, the remaining graph can be placed on a single machine.

Our algorithm, although similar, is simpler than the round compression technique. We do not require sampling since vertices are assigned to machines by degree, deterministically. The crux of our argument is showing that allowing for total space in terms of the arboricity  $\alpha$  leads to a simpler and deterministic argument. Furthermore, for this specific problem, we also do not need to place the induced subgraph on one machine. In the next section, we show an implementation that allows us to operate in the sublinear space per machine regime. We hope our algorithm and analysis will lead to other deterministic algorithms for bounded arboricity graphs in sublinear space per machine and  $O(\log \log n)$  rounds.

## 4.1 MPC Implementation Details

In order to implement COUNT-TRIANGLES( $G$ ) in the MPC model, we define our FIND-TRIANGLES( $w, \mathcal{L}$ ) procedure and provide additional details on sending and storing neighbor lists across different machines. We define *high-degree* vertices to be the set of

vertices whose degree is  $> \gamma$  and *low-degree* vertices to be ones whose degree is  $\leq \gamma$  (for some  $\gamma$  defined in our algorithm). We now define the function  $\text{FIND-TRIANGLES}(w, \mathcal{L})$  used in the above procedure:

■ **Algorithm 2**  $\text{Find-Triangles}(w, \mathcal{L}_w)$ .

- 
- 1: Sort all elements in  $(\mathcal{L}_w \cup (N(w) \cap Q_i))$  lexicographically, using the procedure given in Lemma 4.3 of [57]. Let this sorted list of all elements be  $S$ .
  - 2: Let  $T$  denote the corrected<sup>4</sup> number of duplicates in  $S$  using Theorem 8.
  - 3: Return  $T$ .
- 

**Allocating machines for sorting.** Since each  $v \in Q_i$  could have multiple neighbors whose degrees are  $\leq \gamma$ , the total size of all neighbor lists  $v$  receives could exceed their allowed space  $\Theta(n^\delta)$ . Thus, we allocate  $O\left(\frac{\gamma d_{Q_i}(v)}{n^\delta}\right)$  machines for each vertex  $v \in Q_i$  to store all neighbor lists that  $v$  receives.

The complete analysis for Theorem 11 is given in Appendix A.1.

We provide two additional extensions of our triangle counting algorithm to counting  $k$ -cliques:

► **Theorem 14.** *Given a graph  $G = (V, E)$  with arboricity  $\alpha$ , we can count all  $k$ -cliques in  $O(m\alpha^{k-2})$  total space,  $O_\delta(\log \log n)$  rounds, on machines with  $O(n^{2\delta})$  space for any  $0 < \delta < 1$ .*

We can prove a stronger result when we have some bound on the arboricity of our input graph. Namely, if  $\alpha = O(n^{\delta'/2})$  for any  $\delta' < \delta$ , then we obtain the following result:

► **Theorem 15.** *Given a graph  $G = (V, E)$  with arboricity  $\alpha$  where  $\alpha = O(n^{\delta'/2})$  for any  $\delta' < \delta$ , we can count all  $k$ -cliques in  $O(n\alpha^2)$  total space and  $O_\delta(\log \log n)$  rounds, on machines with  $O(n^\delta)$  space for any  $0 < \delta < 1$ .*

The proofs of these theorems are provided in our full paper [27].

## 5 Approximate Triangle Counting in General Graphs

In this section we provide our algorithm for estimating the number of triangles in general graphs (see Algorithms 3 and 6) and hence prove Theorem 1.

► **Theorem 1.** *Let  $G = (V, E)$  be a graph with  $n$  vertices,  $m$  edges, and let  $T$  be the number of triangles in  $G$ . Assuming*

$$(i) \ T = \tilde{\Omega}\left(\sqrt{\frac{m}{S}}\right), \quad (ii) \ S = \tilde{\Omega}\left(\max\left\{\frac{\sqrt{m}}{\varepsilon}, \frac{n^2}{m}\right\}\right),$$

*there exists an MPC algorithm, using  $\mathcal{M}$  machines, each with local space  $S$ , and total space  $\mathcal{M}S = \tilde{O}_\varepsilon(m)$ , that outputs a  $(1 \pm \varepsilon)$ -approximation of  $T$ , with high probability, in  $O(1)$  rounds.*

The rationale behind the lower bound constraints in Theorem 1 will become clear when we discuss the challenges and analysis (formally presented in the following sections).



## 5.1 Overview of the Algorithm and Challenges

Our approach is to use the collection of machines to repeat the following experiment multiple times in parallel. Each machine  $M_i$  samples a subset of vertices  $V_i$ , and then counts the number of triangles  $\hat{T}_i$  seen in each induced graph  $G[V_i]$ . We then use the sum  $\hat{T}$  of all  $\hat{T}_i$ 's as an unbiased estimator (after appropriate scaling) for the number of triangles  $T$  in the original graph.

---

■ **Algorithm 3** Approximate-Triangle-Counting( $G=(V,E)$ ).

---

```

1: $R \leftarrow 0$
2: parfor $i \leftarrow 1 \dots \mathcal{M}$ do
3: Let V_i be a random subset of V ▷ See Section 5.2 for details about the sampling
4: if size of $G[V_i]$ exceeds machine space S then
5: Ignore this sample and set $\hat{T}_i \leftarrow 0$
6: else
7: Let \hat{T}_i be the number of triangles in $G[V_i]$
8: $R \leftarrow R + 1$
9: end parfor
10: Let $\hat{T} = \sum_{i=1}^{\mathcal{M}} \hat{T}_i$
11: return $\frac{1}{\hat{p}^3 R} \hat{T}$

```

---

Moving forwards, for the most part, we will focus on a specific machine  $M_i$  containing  $V_i$  (a single experiment). We list the main challenges in the analysis of this algorithm, along with the sections that describe them.

1. **Section 5.2:** The induced subgraph  $G[V_i]$  fits into the memory  $S$  of  $M_i$  (thus allowing us to count the number of triangles in  $G[V_i]$  in one round).
2. **Section 5.3:** We can efficiently (in one round) collect all the edges in the induced subgraph  $G[V_i]$ . This involves presenting an MPC protocol such that the number of messages *sent and received* by any machine is at most the *space per machine*  $S$ .
3. **Section 5.4** With high constant probability, the number of messages sent and received by each machine  $M_i$  is at most  $S$ .
4. **Section 5.5:** With high *constant* probability (of at least 0.9), the sum of triangles across all machines,  $\hat{T}$ , is close to its expected value. Then, repeating the algorithm polylogarithmic number of times with only a polylogarithmic increase in total space, and by using the median trick, allows us to get a high probability bound. The specifics are discussed in Appendix A.1.7.

In each of the following sections, we first present a high level overview of the challenges that we need to solve and then follow these high-level descriptions with detailed proofs.

## 5.2 Challenge (1): Ensuring That $G[V_i]$ Fits on a Single Machine

### Ensuring that *edges* fit on a machine

Our algorithm constructs  $V_i$  by including each  $v \in V$  with probability  $\hat{p}$ , which implies that the expected number of edges in  $G[V_i]$  is  $\hat{p}^2 m$ . Since we have to ensure that each induced subgraph  $G[V_i]$  fits on a single machine, we obtain the constraint  $\hat{p}^2 m = O(S)$ . Concretely, we achieve this by defining:

$$\hat{p} \stackrel{\text{def}}{=} \frac{1}{10} \cdot \sqrt{\frac{S}{mk}}, \quad (1)$$

where the parameter  $k = O(\log n)$  will be exactly determined later (See Section 5.3).

### Ensuring that *vertices* fit on a machine

In certain regimes of values of  $n$  and  $m$ , the *expected number of vertices* ending up in an induced subgraph –  $\hat{p}n$ , may exceed the space limit  $S$ . Avoiding this scenario introduces an additional constraint  $\hat{p}n = O(S) \iff S = \Omega(kn^2/m)$ .

### Getting a high probability guarantee

As discussed above, the value of  $\hat{p} = \tilde{\Theta}_\varepsilon(\sqrt{S/m})$  is chosen specifically so that the *expected number of edges* in the induced subgraphs  $G[V_i]$  is  $\hat{p}^2 m \leq \Theta(S)$ , thus using all the available space (asymptotically). In order to guarantee that this bound holds *with high probability* (see Appendix A.1.4), we require additional constraints on the space per machine  $S = \tilde{\Omega}_\varepsilon(\sqrt{m})$ . We remark that this lower bound  $S = \tilde{\Omega}_\varepsilon(\sqrt{m})$  is essentially saying that  $\mathcal{M} = \tilde{O}_\varepsilon(\sqrt{m})$ , i.e. the *space per machine* is much larger than the *number of machines*. This is a realistic assumption as in practice we can have machines with  $10^{11}$  words of local random access memory, however, it is unlikely that we also have as many machines in our cluster.

### Lower Bound on *space per machine*

Combining the above two constraints, we get:

$$S > \max \left\{ 15 \frac{\sqrt{mk}}{\varepsilon}, \frac{100kn^2}{m} \right\} \implies S = \tilde{\Omega}_\varepsilon \left( \max \left\{ \sqrt{m}, \frac{n^2}{m} \right\} \right) \quad (2)$$

Note that Eq. (2) always allows *linear space per machine*, as long as  $m = \Omega(n)$ . The following sections, Appendices A.1.4 and A.1.5 present a detailed analysis, showing that the number of vertices and edges in each subgraph is at most  $S$  *with high probability*. In this high-level overview of the challenges, we defer a detailed analysis of these bounds to the later sections (Appendices A.1.4 and A.1.5) since the formal proof of these bounds also require a discussion of Section 5.3.

## 5.3 Challenge (2): Using $k$ -wise Independence to Compute the Induced Subgraph $G[V_i]$ in MPC

For each sub-sampled set of vertices  $V_i$ , we need to compute  $G[V_i]$ , i.e. we need to send all the edges in the induced subgraph  $G[V_i]$  to the machine  $M_i$ . Let  $Q_u$  denote the set of all machines containing  $u$ . Each edge  $(u, w)$  then needs to be sent to all machines that contain both  $u$  and  $w$ ,  $Q_u \cap Q_w$ . Naively, one could try to send the sets  $Q_u$  and  $Q_w$  to the edge  $e = (u, w)$ , for all  $e \in E$ . However, this strategy could result in  $Q_v$  being replicated  $d(v)$  times. Since the expected size of  $Q_v$  is  $|Q_v| = \hat{p}\mathcal{M}$  the total expected memory usage of this strategy would be  $\sum_{v \in V} |Q_v| \cdot d(v) = \tilde{\Theta}_\varepsilon(m \cdot \hat{p}\mathcal{M}) = \tilde{\omega}_\varepsilon(m)$ , since  $\hat{p} = \tilde{\Theta}(1/\sqrt{\mathcal{M}})$ . This defies our goal of optimal total memory.

Instead, we address this challenge by using globally known hash functions to sample the vertices on each machine. That is, we let  $h : V \times [\mathcal{M}] \rightarrow \{0, 1\}$  (formally presented in Definition 16) be a hash function known globally to all the machines. Then we can compute the induced subgraphs  $G[V_i]$  as follows.

► **Definition 16.** The hash function  $h(v, i)$  indicates whether vertex  $v$  is sampled in  $V_i$  or not. Specifically,  $h : V \times [\mathcal{M}] \rightarrow \{0, 1\}$  such that  $\mathbb{P}[h(v, i) = 1] = \hat{p}$  for all  $v \in V$  and  $i \in [\mathcal{M}]$ . Recall that  $\mathcal{M}$  is the number of machines, and  $\hat{p} = \frac{1}{10} \cdot \sqrt{\frac{S}{mk}}$  is the sampling probability set in Eq. (1).

---

**Algorithm 4** Compute-Induced-Subgraphs.

---

```

1: $Q_v \leftarrow \{i \in [\mathcal{M}] \mid h(v, i) = 1\}$.
2: $Q_w \leftarrow \{i \in [\mathcal{M}] \mid h(w, i) = 1\}$.
3: parfor $i \in Q_v \cap Q_w$ do
4: Send e to machine M_i , containing V_i .
5: end parfor

```

---

**Using limited independence.** Ideally, we would want a perfect hash function, which would allow us to sample the  $V_i$ 's i.i.d. from the uniform distribution on  $V$ . However, since the hash function needs to be known globally, it must fit into each of the machines. This implies that we *cannot* use a fully independent perfect hash function. Rather, we *can* use one that has a high level of independence. Specifically, given that the space per machine is  $S$ , we can have a globally known hash function  $h$  that is  $k$ -wise independent<sup>5</sup> for any  $k < \Theta(S/\log n)$ . In fact, we can get away with as little as  $(6 \log n)$ -wise independence (i.e.,  $k = 6 \log n$ ). Recalling Eq. (1), this also fixes the sampling probability to be  $\hat{p} = \sqrt{S/600m \log n}$ .

### 5.4 Challenge (3): Showing that, with high constant probability, the size of the sent/received messages is bounded

We need to show that the number of edges sent and received by any machine  $M_i$  is at most  $S$  with high constant probability. To this end, we partition the vertex set  $V$  into  $V_{light}$  and  $V_{heavy}$  by picking a threshold degree  $\tau$  for the vertices. Following this, we define *light edges* as ones that have both end-points in  $V_{light}$ , and conversely, any edge with at least one end-point in  $V_{heavy}$  is designated as *heavy*. In order for the protocol to succeed, the following must hold:

- (A) The number of *light edges* concentrates (see Appendix A.1.4).
- (B) The number of *heavy edges* concentrates (see Appendix A.1.5).
- (C) The number of sent messages is at most  $S$  (see Appendix A.1.6).

The first two items ensure that each machine  $M_i$  *receives* at most  $S$  messages, and the last item ensures that each machine *sends* at most  $S$  messages. Given the above, we proceed to address the last challenge.

### 5.5 Challenge (4): $\hat{T}$ is close to its expected value

In this section, we provide merely a brief discussion of this challenge for intuition, and we fully analyze the approximation guarantees of our algorithm in Appendix A.1.3. That analysis also makes clear the source of our advertised lower-bound on  $T$  for which an estimated count concentrates well.

**Lower Bound on Number of Triangles.** In order to output *any* approximation (note that we are ignoring all factors of  $\varepsilon$  and  $O(\text{poly } \log n)$  here) to the triangle count, we must see  $\Omega(1)$  triangles amongst all of the induced subgraphs on all the machines. The expected number of triangles in a specific induced  $G[V_i]$  is  $\hat{p}^3 T$ , and therefore, the expected number of triangles overall is  $\hat{p}^3 T \mathcal{M}$  which must be  $\Omega(1)$  for some setting of  $T$ . Since we set  $\hat{p}$  such that  $\hat{p}^2 m = \Theta(S)$ , this gives that  $\hat{p}^2 = O(S/m)$  which implies  $\hat{p}^2 \cdot \mathcal{M} = \hat{p}^2 \cdot (m/S) = \Theta(1)$ .

---

<sup>5</sup> A  $k$ -wise independent hash function is one where the hashes of *any*  $k$  distinct keys are guaranteed to be independent random variables (see [92]).

This then immediately implies that to show that  $\hat{p}^3 T$  is  $\Omega(1)$ , we need only show that  $\hat{p} \cdot T$  is  $\Omega(1)$ . Specifically, we show in Lemma 20 that when  $T > 1/\hat{p}$ , we can obtain a  $(1 \pm \varepsilon)$ -approximation. To get some intuition for this lower bound on  $T$ , note that, in the linear memory regime, when  $S = \Theta(n)$ , this translates to  $T > \sqrt{d_{avg}}$ , where  $d_{avg}$  is the average degree of  $G$ .

$$T > \frac{1}{\hat{p}} = \tilde{\Theta} \left( \sqrt{\frac{m}{S}} \right) \xrightarrow{\text{for } S=\tilde{\Theta}(n)} T > \tilde{\Theta} \left( \sqrt{d_{avg}} \right).$$

## 6 Open Questions

There are many interesting open questions that result from our study; among these open questions include improving the bounds presented in our algorithm: the round complexity and total space usage in our exact algorithms and the space per machine in our approximation algorithms. In addition to these questions, we also discuss two additional open questions with a larger research scope.

### Small subgraph counting for a broader class of small subgraphs

Two recent works of [32, 26] extend the result of [25] to a broader set of small subgraphs in the sequential model. However, their results depend crucially on a *DAG tree decomposition* which is non-trivial to implement in the MPC model. Furthermore, even given this DAG tree decomposition, their approach requires iterating through the tree from the leaf level by level up the tree. Such a procedure when implemented in the MPC model requires number of rounds that is  $O(\text{depth})$  where *depth* is the depth of the tree. The depth may not be  $\text{poly}(\log n)$ . In order to obtain efficient MPC implementation of these new algorithms, we must find novel solutions to the above two challenges.

### Counting in the AMPC model

A new (stronger) model of MPC, called the *adaptive* MPC model, was recently introduced by [22]. The AMPC model allows access to a *shared* distributed hash table at the end of every round; additionally, the algorithms are allowed *adaptive* access to this hash table. Such a model has shown to be very practical and have led to improvements in the number of rounds over previous MPC algorithms. Such a model seems to be quite relevant to our work since one of the main challenges in our approximation algorithms is to find the set of edges to give to each machine. (Such a challenge may no longer exist given a shared-memory distributed hash table.) We leave as an interesting open question to obtain better, more round efficient approximate triangle counting algorithms in the AMPC model.

### Triangle Counting in $O(1)$ Rounds in Sparse Graphs

For sparse graphs where  $m = \tilde{O}(n)$ , our approximation algorithm requires  $\tilde{\Omega}(n)$  space per machine which means that (almost) the entire graph can fit on one machine. This naturally leads to an interesting open question for whether we can obtain an approximate or exact triangle counting algorithm in  $O(1)$  rounds in sparse graphs while using *sublinear* space per machine ( $n^\delta$  space for any constant  $\delta > 0$ ).

---

References

---

- 1 GraphChallenge. URL: <http://graphchallenge.mit.edu/>.
- 2 Foto N Afrati, Dimitris Fotakis, and Jeffrey D Ullman. Enumerating subgraph instances using map-reduce. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 62–73. IEEE, 2013.
- 3 Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, Nick G. Duffield, and Theodore L. Willke. Graphlet decomposition: framework, algorithms, and applications. *Knowl. Inf. Syst.*, 50(3):689–722, 2017.
- 4 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *SPAA*, pages 202–211, 2015.
- 5 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.
- 6 Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54(4):544–556, 2009. doi:10.1007/s00453-008-9204-0.
- 7 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
- 8 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *STOC*, pages 574–583, 2014.
- 9 Alexandr Andoni, Clifford Stein, and Peilin Zhong. Log diameter rounds algorithms for 2-vertex and 2-edge connectivity. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 14:1–14:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.14.
- 10 Shaikh Arifuzzaman, Maleq Khan, and Madhav Marathe. Patric: A parallel algorithm for counting triangles in massive networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 529–538, 2013.
- 11 Sepehr Assadi. Simple round compression for parallel vertex cover. *arXiv preprint*, 2017. arXiv:1709.04599.
- 12 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019.
- 13 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *ITCS*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
- 14 Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. *arXiv preprint*, 2018. arXiv:1805.02974.
- 15 Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 461–470. ACM, 2019. doi:10.1145/3293611.3331596.
- 16 Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 623–632. Society for Industrial and Applied Mathematics, 2002.
- 17 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 273–284, 2013.

- 18 Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in parallel query processing. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 212–223, 2014.
- 19 Soheil Behnezhad, Sebastian Brandt, Mahsa Derakhshan, Manuela Fischer, MohammadTaghi Hajiaghayi, Richard M. Karp, and Jara Uitto. Massively parallel computation of matching and MIS in sparse graphs. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 481–490. ACM, 2019. doi:10.1145/3293611.3331609.
- 20 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Semi-mapreduce meets congested clique. *arXiv preprint*, 2018. arXiv:1802.10297.
- 21 Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Marina Knittel, and Hamed Saleh. Streaming and massively parallel algorithms for edge coloring. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.15.
- 22 Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Lacki, Vahab Mirrokni, and Warren Schudy. Massively parallel computation via remote memory access. *ACM Transactions on Parallel Computing*, 8(3):1–25, 2021.
- 23 Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially faster massively parallel maximal matching. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1637–1649. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00096.
- 24 Suman K Bera and Amit Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In *34th Symposium on Theoretical Aspects of Computer Science*, 2017.
- 25 Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear Time Subgraph Counting, Graph Degeneracy, and the Chasm at Size Six. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:20, Dagstuhl, Germany, 2020. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2020.38.
- 26 Suman K Bera, Noujan Pashanasangi, and C Seshadhri. Near-linear time homomorphism counting in bounded degeneracy graphs: the barrier of long induced cycles. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2315–2332. SIAM, 2021.
- 27 Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, and Ronitt Rubinfeld. Parallel algorithms for small subgraph counting. *CoRR*, abs/2002.08299, 2020. arXiv:2002.08299.
- 28 Andreas Bjöklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. *Algorithms - ESA 2009*, pages 578–586, 2009. doi:10.1007/978-3-642-04128-0\_52.
- 29 Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, MohammadTaghi HajiAghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: quantum and MapReduce. In *SODA*, pages 1170–1189, 2018.
- 30 Sebastian Brandt, Manuela Fischer, and Jara Uitto. Breaking the linear-memory barrier in MPC: Fast MIS on trees with  $n^\epsilon$  memory per machine. *arXiv preprint*, 2018. arXiv:1802.06748.
- 31 Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. How hard is counting triangles in the streaming model? In *International Colloquium on Automata, Languages, and Programming*, pages 244–254. Springer, 2013.



- 32 Marco Bressan. Faster subgraph counting in sparse graphs. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- 33 Keren Censor-Hillel, Petteri Kaski, Janne H Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. *Distributed Computing*, 32(6):461–478, 2019.
- 34 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of  $(\Delta+1)$  coloring in congested clique, massively parallel computation, and centralized local computation. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 471–480. ACM, 2019. doi:10.1145/3293611.3331607.
- 35 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985.
- 36 Shumo Chu and James Cheng. Triangle listing in massive networks and its applications. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 672–680, 2011.
- 37 Jonathan Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4):29–41, 2009.
- 38 Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1197655.
- 39 Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing  $k$ -cliques in sparse real-world graphs\*. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 589–598, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. doi:10.1145/3178876.3186125.
- 40 V. S. Dave, N. K. Ahmed, and M. Hasan. PE-CLoG: Counting edge-centric local graphlets. In *IEEE International Conference on Big Data*, pages 586–595, 2017.
- 41 Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- 42 Laxman Dhulipala, Quanquan C. Liu, Julian Shun, and Shangdi Yu. Parallel batch-dynamic  $k$ -clique counting. In Michael Schapira, editor, *2nd Symposium on Algorithmic Principles of Computer Systems, APOCS 2020, Virtual Conference, January 13, 2021*, pages 129–143. SIAM, 2021. doi:10.1137/1.9781611976489.10.
- 43 Danny Dolev, Christoph Lenzen, and Shir Peled. “Tri, Tri again”: Finding triangles and small subgraphs in a distributed setting. *Distributed Computing*, pages 195–209, 2012. doi:10.1007/978-3-642-33651-5\_14.
- 44 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.
- 45 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of  $k$ -cliques in low-arboricity graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1467–1478, 2020. doi:10.1137/1.9781611975994.89.
- 46 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1–3):57–67, October 2004. doi:10.1016/j.tcs.2004.05.009.
- 47 Ethan R. Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G. Dimakis. Distributed estimation of graph 4-profiles. In *International Conference on World Wide Web (WWW)*, pages 483–493, 2016.
- 48 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithms*, 18(3):364–375, 2013. doi:10.1145/2543629.



- 49 Irene Finocchi, Marco Finocchi, and Emanuele G Fusco. Clique counting in mapreduce: Algorithms and experiments. *Journal of Experimental Algorithmics (JEA)*, 20:1–20, 2015.
- 50 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.
- 51 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *PODC*. arXiv:1802.08237, 2018.
- 52 Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional hardness results for massively parallel computation from distributed lower bounds. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1650–1663. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00097.
- 53 Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrović. Improved parallel algorithms for density-based network clustering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2201–2210, Long Beach, California, USA, 09–15 June 2019. PMLR. URL: <http://proceedings.mlr.press/v97/ghaffari19a.html>.
- 54 Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1260–1279. SIAM, 2020. doi:10.1137/1.9781611975994.77.
- 55 Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1636–1653. SIAM, 2019. doi:10.1137/1.9781611975482.99.
- 56 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, pages 159–167, 2006. doi:10.1007/11917496\_15.
- 57 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Proceedings of the 22Nd International Conference on Algorithms and Computation, ISAAC'11*, pages 374–383, Berlin, Heidelberg, 2011. Springer-Verlag. doi:10.1007/978-3-642-25591-5\_39.
- 58 Nicholas J. A. Harvey, Christopher Liaw, and Paul Liu. Greedy and local ratio algorithms in the MapReduce model. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 43–52, New York, NY, USA, 2018. ACM. doi:10.1145/3210377.3210386.
- 59 James W Hegeman and Sriram V Pemmaraju. Lessons from the congested clique applied to MapReduce. *Theoretical Computer Science*, 608:268–281, 2015.
- 60 Tomaz Hocevar and Janez Demsar. A combinatorial approach to graphlet counting. *Bioinformatics*, pages 559–65, 2014.
- 61 Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *STOC*, pages 798–811, 2017.
- 62 Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS operating systems review*, volume 41(3), pages 59–72. ACM, 2007.
- 63 Giuseppe F. Italiano, Silvio Lattanzi, Vahab S. Mirrokni, and Nikos Parotsidis. Dynamic algorithms for the massively parallel computation model. In Christian Scheideler and Petra Berenbrink, editors, *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019*, pages 49–58. ACM, 2019. doi:10.1145/3323165.3323202.

- 64 Shweta Jain and C Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Proceedings of the 26th International Conference on World Wide Web*, pages 441–449, 2017.
- 65 Madhav Jha, Comandur Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 589–597, 2013.
- 66 Daniel M Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 598–609. Springer, 2012.
- 67 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *SODA*, pages 938–948, 2010.
- 68 Tamara G Kolda, Ali Pinar, Todd Plantenga, C Seshadhri, and Christine Task. Counting triangles in massive graphs with mapreduce. *SIAM Journal on Scientific Computing*, 36(5):S48–S77, 2014.
- 69 Mihail N Kolountzakis, Gary L Miller, Richard Peng, and Charalampos E Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.
- 70 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1272–1287. SIAM, 2016.
- 71 Jakub Lacki, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Walking randomly, massively, and efficiently. *CoRR*, abs/1907.05391, 2019. [arXiv:1907.05391](#).
- 72 Longbin Lai, Lu Qin, Xuemin Lin, and Lijun Chang. Scalable subgraph enumeration in mapreduce. *Proceedings of the VLDB Endowment*, 8(10):974–985, 2015.
- 73 Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical computer science*, 407(1-3):458–473, 2008.
- 74 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA*, pages 85–94, 2011.
- 75 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in  $o(\log \log n)$  communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.
- 76 Andrew McGregor, Sofya Vorotnikova, and Hoa T Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 401–411, 2016.
- 77 Krzysztof Onak. Round compression for parallel graph algorithms in strongly sublinear space. *CoRR*, abs/1807.08745, 2018. [arXiv:1807.08745](#).
- 78 Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.
- 79 Ha-Myung Park and Chin-Wan Chung. An efficient mapreduce algorithm for counting triangles in a very large graph. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 539–548, 2013.
- 80 Ha-Myung Park, Francesco Silvestri, U Kang, and Rasmus Pagh. Mapreduce triangle enumeration with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1739–1748, 2014.
- 81 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610, 2010. doi:10.1145/1806689.1806772.
- 82 Ali Pinar, C. Seshadhri, and Vaidyanathan Vishal. ESCAPE: Efficiently counting all 5-vertex subgraphs. In *International Conference on World Wide Web (WWW)*, pages 1431–1440, 2017.
- 83 Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and circuits: (on lower bounds for modern parallel computation). In *SPAA*, pages 1–12, 2016.

- 84 Thomas Schank and Dorothea Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *Experimental and Efficient Algorithms*, pages 606–609. Springer, 2005.
- 85 Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 10–18. SIAM, 2013.
- 86 Jessica Shi, Laxman Dhulipala, and Julian Shun. Parallel clique counting and peeling algorithms. *CoRR*, abs/2002.10047, 2020. [arXiv:2002.10047](#).
- 87 Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018.
- 88 J. Shun and K. Tangwongsan. Multicore triangle computations without tuning. In *2015 IEEE 31st International Conference on Data Engineering*, pages 149–160, April 2015. doi:10.1109/ICDE.2015.7113280.
- 89 Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th international conference on World wide web*, pages 607–614, 2011.
- 90 Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 837–846, 2009.
- 91 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009. doi:10.1016/j.ipl.2008.10.014.
- 92 Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.
- 93 Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
- 94 Jin-Hyun Yoon and Sung-Ryul Kim. Improved sampling for triangle counting with mapreduce. In *International Conference on Hybrid Information Technology*, pages 685–689. Springer, 2011.
- 95 Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

## **A** Exact Triangle Counting Analysis

### **A.1 Detailed Analysis**

In this section we give the full details and analysis of algorithm Algorithm 1 given in Section 4, for exactly counting the number of triangles in the graph.

We first provide a detailed version of Algorithm 2 that also takes into account over counting due to the fact that each triangle might be counted by several endpoints, and then continue to prove the main theorem of this section, Theorem 4.

#### **A.1.1 Details about finding duplicate elements using Theorem 8**

$\text{FIND-TRIANGLES}(w, \mathcal{L}_w)$  finds triangles by counting the number of duplicates that occur between elements in lists. Theorem 8 provides a MPC implementation for finding the count of all occurrences of every element in a sorted list. Provided a sorted list of neighbors of  $v \in Q_i$  and neighbor lists in  $\mathcal{L}_v$ , this function counts the number of intersections between a neighbor list sent to  $v$  and the neighbors of  $v$ . Every intersection indicates the existence of a triangle. As given,  $\text{FIND-TRIANGLES}(w, \mathcal{L}_w)$  (see v Algorithm 2) returns a 6-approximation of the number of triangles in any graph. We provide a detailed and somewhat more complicated algorithm  $\text{FIND-TRIANGLES-EXACT}(w, \mathcal{L}_w)$  that accounts for over-counting of triangles and returns the exact number of triangles.

Since Theorem 8 returns the total count of each element, we subtract the value returned by 1 to obtain the number of intersections. Finally, each triangle containing one low-degree vertex will be counted twice, each containing two low-degree vertices will be counted 4 times, and each containing three low-degree vertices will be counted 6 times. Thus, we need to divide the counts by 2, 4, and 6, respectively, to obtain the exact count of unique triangles.

■ **Algorithm 5 Find-Triangles-Exact**( $w, \mathcal{L}_w$ ).

---

```

1: Set the number of triangles $T_i \leftarrow 0$.
2: Sort all elements in $(\mathcal{L}_w \cup (N(w) \cap Q_i))$ lexicographically using the procedure given in
 Lemma 4.3 of [57]. Let this sorted list of all elements be S .
3: Count the duplicates in S using Theorem 8.
4: parfor all $v \in N(w)$ do
5: Let R be the number of duplicates of v returned by Theorem 8.
6: if $d_{Q_i}(v) > \gamma_i$ and $d_{Q_i}(w) > \gamma_i$ then
7: Increment $T_i \leftarrow T_i + \frac{R-1}{2}$.
8: else if $(d_{Q_i}(v) > \gamma_i \text{ and } d_{Q_i}(w) \leq \gamma_i) \text{ or } (d_{Q_i}(v) \leq \gamma_i \text{ and } d_{Q_i}(w) > \gamma_i)$ then
9: Increment $T_i \leftarrow T_i + \frac{R-1}{4}$.
10: else
11: Increment $T_i \leftarrow T_i + \frac{R-1}{6}$.
12: end parfor
13: Return T_i .

```

---

Substituting FIND-TRIANGLES-EXACT in COUNT-TRIANGLES finds the exact count of triangles in graphs with arboricity  $\alpha$  using  $O(m\alpha)$  total space.

### A.1.2 Proof of Theorem 4

First, all proofs below assume we start at a cutoff of  $\gamma = 4\alpha$ . Because we increase the cutoff bound doubly exponentially, we can reach such a bound in  $O(\log \log \alpha)$  rounds. Thus, in the following proofs, we ignore all rounds before we get to a round where  $\gamma \geq 4\alpha$ . Before proving the theorem, we provide several useful lemmas stating that the number of vertices and edges remaining at the beginning of each iteration is bounded.

► **Lemma 17.** *At the beginning of iteration  $i$  of COUNT-TRIANGLES, given  $\gamma_i = 2^{(3/2)^i} \cdot (2\alpha)$  as stated in Algorithm 1, the number of remaining vertices  $N_i = |Q_i|$  is at most  $\frac{n}{2^{2 \cdot ((3/2)^i - 1)}}$ .*

**Proof.** Let  $N_i$  be the number of vertices in  $Q_i$  at the beginning of iteration  $i$ . Since the subgraph induced by  $Q_i$  must have arboricity bounded by  $\alpha$ , we can bound the total degree of  $Q_i$ ,

$$\sum_{v \in Q_i} d_{Q_i}(v) < 2\alpha|Q_i| = 2N_i\alpha.$$

At the end of the iteration, we only keep the vertices in  $Q_{i+1} = \{v \in Q_i \mid d_{Q_i}(v) > \gamma_i\}$ . If we assume that  $|Q_{i+1}| > \frac{N_i}{\gamma_i/(2\alpha)}$ , then we obtain a contradiction since this implies that

$$\sum_{v \in Q_{i+1}} d_{Q_i}(v) > |Q_{i+1}| \cdot \gamma_i > 2N_i\alpha > \sum_{v \in Q_i} d_{Q_i}(v).$$

Then, the number of remaining vertices follows directly from the above by induction on  $i$  with base case  $N_1 = n$ ,

$$N_i \leq \frac{N_{i-1}}{\gamma_i/(2\alpha)} = \frac{N_{i-1}}{2^{(3/2)^{i-1}}} \leq \frac{n}{\prod_{j=0}^{i-1} 2^{(3/2)^j}} = \frac{n}{2^{2 \cdot ((3/2)^i - 1)}}. \quad \blacktriangleleft$$

We can show a similar statement for the number of edges that remain at the start of the  $i^{\text{th}}$  iteration.

► **Lemma 18.** *At the beginning of iteration  $i$  of COUNT-TRIANGLES, given  $\gamma_i$ , the number of remaining edges  $m_i$  is at most  $m_i \leq \frac{m}{2^{2 \cdot ((3/2)^i - 1)}}$ .*

**Proof.** The number of vertices remaining at the beginning of iteration  $i$  is given by  $|Q_i|$ . Thus, because the arboricity of our graph is  $\alpha$ , we can upper bound  $m_i$  by

$$m_i \leq |Q_i|\alpha.$$

Then, we can also lower bound the number of edges at the beginning of iteration  $i - 1$  since the vertices that remain at the beginning of round  $i$  are ones which have greater than  $\gamma_{i-1}$  degree,

$$m_{i-1} \geq \frac{1}{2} \sum_{v \in Q_{i-1}} d_{Q_{i-1}}(v) \geq \frac{1}{2} |Q_i| \gamma_{i-1}.$$

Thus, we conclude that  $m_i \leq \frac{2\alpha m_{i-1}}{\gamma_{i-1}}$ . By induction on  $i$  with base case  $m_0 = m$ , we obtain,

$$m_i \leq 2\alpha \left( \frac{m_{i-1}}{\gamma_{i-1}} \right) \leq \frac{m}{\prod_{j=0}^{i-1} 2^{(3/2)^j}} = \frac{m}{2^{2 \cdot ((3/2)^i - 1)}}. \quad \blacktriangleleft$$

The above lemmas allows us to bound the total space used by the algorithm.

► **Lemma 19.** *COUNT-TRIANGLES( $G$ ) uses  $O(m\alpha)$  total space when run on a graph  $G$  with arboricity  $\alpha$ .*

**Proof.** The total space the algorithm requires is the sum of the space necessary for storing the neighbor lists sent by all vertices with degree  $\leq \gamma_i$  and the space necessary for all vertices to store their own neighbor lists. The total space necessary for each vertex to store its own neighbor list is  $O(m)$ .

Now we compute the total space used by the algorithm during iteration  $i$ . The number of vertices in  $Q_i$  at the beginning of this iteration is at most  $N_i \leq \frac{n}{2^{2 \cdot ((3/2)^i - 1)}}$  by Lemma 17. Each vertex  $v$  with  $d_{Q_i}(v) \leq \gamma_i$ , makes  $d_{Q_i}(v)$  copies of its neighbor list  $(N(v) \cap Q_i)$  and sends each neighbor in  $N(v) \cap Q_i$  a copy of the list. Thus, the total space required by the messages sent by  $v$  is  $d_{Q_i}(v)^2 \leq \gamma_i^2$ .  $v$  sends at most one message of size  $d_{Q_i}(v) \leq \gamma_i$  along each edge  $(v, w)$  for  $w \in N(v) \cap Q_i$ . Then, by Lemma 18 the total space required by all the low-degree vertices in round  $i$  is at most (as at most two messages are sent along each edge):

$$2m_i \cdot \gamma_i \leq \frac{m}{2^{2 \cdot ((3/2)^i - 1)}} \cdot \left[ 2^{(3/2)^i} (2\alpha) \right] = 16m\alpha. \quad \blacktriangleleft$$

We are now ready to prove Theorem 4.

**Proof of Theorem 4.** By Lemma 17, the number of vertices remaining in  $Q_i$  at the beginning of iteration  $i$  is  $\frac{n}{2^{2 \cdot ((3/2)^i - 1)}}$ . This means that the procedure runs for  $O(\log \log n)$  iterations before there will be no vertices. For each of the  $O(\log \log n)$  iterations, COUNT-TRIANGLES( $G$ ) uses  $O_\delta(1)$  rounds of communication for the low-degree vertices to send their neighbor lists to their neighbors. The algorithm then calls FIND-TRIANGLES-EXACT( $w, \mathcal{L}_w$ ) on each vertex  $w \in Q_i$  (in parallel) to find the number of triangles incident to  $w$  and vertices in  $A_i \subseteq Q_i$ . FIND-TRIANGLES-EXACT( $w, \mathcal{L}_w$ ) requires  $O(\log_{n^\delta}(m\alpha)) = O(1/\delta)$  rounds by Lemma 4.3 of [57] and Theorem 8. Therefore, the total number of rounds required by COUNT-TRIANGLES( $G$ ) is  $O\left(\frac{\log \log n}{\delta}\right) = O_\delta(\log \log n)$ . ◀

### A.1.3 Showing Concentration for the Triangle Count

In the subsequent proofs, we will use the following assumptions from within Theorem 1 (note that we added specific constants).

$$T \geq 10\sqrt{\frac{mk}{S}} \quad S \geq \max\left\{15\frac{\sqrt{mk}}{\varepsilon}, \frac{100kn^2}{m}\right\} \quad \mathcal{M} = \frac{2000mk}{\varepsilon^2 S} \quad (3)$$

Note that we set the number of machines to a specific value, instead of lower bounding it. This is acceptable, because we can just ignore some of the machines.

Algorithm 3 outputs an estimate on the number of triangles in  $G$  (Line 11). It is not hard to show that in expectation this output equals  $T$  *even with limited independence* as discussed above. The main challenge is to show that this output also concentrates well around its expectation. Specifically, we show the following claim.

► **Lemma 20.** *Ignore Line 4 of Algorithm 3. Let  $\hat{T}$  be as defined on Line 10 and  $\mathcal{M} = \frac{20}{\varepsilon^2 \hat{p}^2}$  be as defined in Eq. (3), and assume that  $T \geq 1/\hat{p}$ . Then, the following hold:*

- (A)  $\mathbb{E}[\hat{T}] = \hat{p}^3 \cdot R \cdot T$ , and
- (B)  $\mathbb{P}\left[|\hat{T} - \mathbb{E}[\hat{T}]| > \varepsilon \mathbb{E}[\hat{T}]\right] < \frac{1}{10}$ .

We will prove Property (B) of the claim by applying Chebyshev's inequality, for which we need to compute  $\text{Var}[\hat{T}]$ . Let  $\Delta(G)$  be the set of all triangles in  $G$ . For a triangle  $t \in \Delta(G)$ , let  $\hat{T}_{i,t} = 1$  if  $t \in V[G_i]$ , and  $\hat{T}_{i,t} = 0$  otherwise. Hence,  $\hat{T}_i = \sum_{t \in \Delta(G)} \hat{T}_{i,t}$ . We begin by deriving  $\mathbb{E}[\hat{T}]$  and then proceed to showing that  $\text{Var}[\hat{T}] = \sum_{i=1}^R \text{Var}[\hat{T}_i]$ . After that we upper-bound  $\text{Var}[\hat{T}_i]$  and conclude the proof by applying Chebyshev's inequality.

#### A.1.3.1 Deriving $\mathbb{E}[\hat{T}]$

Let  $t$  be a triangle in  $G$ . Let  $\hat{T}_t$  be a random variable denoting the total number of times  $t$  appears in  $G[V_i]$ , for all  $i = 1 \dots R$ . Given that  $\mathbb{P}[u \in V_i] = \hat{p}$ , we have that  $\mathbb{P}[t \in G[V_i]] = \hat{p}^3$ . Therefore,  $\mathbb{E}[\hat{T}_t] = R \cdot \hat{p}^3$ .

Since  $\hat{T} = \sum_{t \in \Delta(G)} \hat{T}_t$ , we have

$$\mathbb{E}[\hat{T}] = \sum_{t \in \Delta(G)} \mathbb{E}[\hat{T}_t] = \hat{p}^3 \cdot R \cdot T. \quad (4)$$

This proves Property (A) of this claim.

### A.1.3.2 Decoupling $\text{Var} [\hat{T}]$

To compute variance, one considers the second moment of a given random variable. So, to compute  $\text{Var} [\hat{T}]$ , we will consider products  $\hat{T}_{i,t_1} \cdot \hat{T}_{j,t_2}$ . Each of those products depend on at most 6 vertices. Now, given that we used a 6-wise independent function (see Section 5.3) to sample vertices in each  $V_i$ , one could expect that  $\text{Var} [\hat{T}_i]$  and  $\text{Var} [\hat{T}_j]$  for  $i \neq j$  behave like they are independent, i.e., one could expect that it holds  $\text{Var} [\hat{T}] = \sum_{i=1}^R \text{Var} [\hat{T}_i]$ . As we show next, it is indeed the case. We have

$$\text{Var} [\hat{T}] = \mathbb{E} [\hat{T}^2] - \mathbb{E} [\hat{T}]^2 = \mathbb{E} \left[ \left( \sum_{i=1}^R \sum_{t \in \Delta(G)} \hat{T}_{i,t} \right)^2 \right] - \left( \sum_{i=1}^R \sum_{t \in \Delta(G)} \mathbb{E} [\hat{T}_{i,t}] \right)^2$$

Consider now  $\hat{T}_{i,t_1}$  and  $\hat{T}_{j,t_2}$  for  $i \neq j$  and some  $t_1, t_2 \in \Delta(G)$  not necessarily distinct. In the first summand of (5), we will have  $\mathbb{E} [2\hat{T}_{i,t_1} \cdot \hat{T}_{j,t_2}]$ . The vertices constituting  $t_1$  and  $t_2$  are 6 *distinct* copies of some (not necessarily all distinct) vertices of  $V$ . Since they are chosen by applying a 6-wise independent function, we have  $\mathbb{E} [2\hat{T}_{i,t_1} \cdot \hat{T}_{j,t_2}] = 2\mathbb{E} [\hat{T}_{i,t_1}] \cdot \mathbb{E} [\hat{T}_{j,t_2}]$ . On the other hand, the second summand of (5) also contains  $2\mathbb{E} [\hat{T}_{i,t_1}] \cdot \mathbb{E} [\hat{T}_{j,t_2}]$ , which follows by direct expansion of the sum. Therefore, all the terms  $\mathbb{E} [2\hat{T}_{i,t_1} \cdot \hat{T}_{j,t_2}]$  in  $\text{Var} [\hat{T}]$  for  $i \neq j$  cancel each other. So, we can also write  $\text{Var} [\hat{T}]$  as

$$\text{Var} [\hat{T}] = \sum_{i=1}^R \mathbb{E} \left[ \left( \sum_{t \in \Delta(G)} \hat{T}_{i,t} \right)^2 \right] - \sum_{i=1}^R \left( \sum_{t \in \Delta(G)} \mathbb{E} [\hat{T}_{i,t}] \right)^2 = \sum_{i=1}^R \text{Var} [\hat{T}_i].$$

Therefore, to upper-bound  $\text{Var} [\hat{T}]$  it suffices to upper-bound  $\text{Var} [\hat{T}_i]$ .

### A.1.3.3 Upper-bounding $\text{Var} [\hat{T}_i]$

We have

$$\begin{aligned} \text{Var} [\hat{T}_i] &= \mathbb{E} \left[ \left( \sum_{t \in \Delta(G)} \hat{T}_{i,t} \right)^2 \right] - \left( \sum_{t \in \Delta(G)} \mathbb{E} [\hat{T}_{i,t}] \right)^2 \leq \mathbb{E} \left[ \left( \sum_{t \in \Delta(G)} \hat{T}_{i,t} \right)^2 \right] \\ &= \mathbb{E} \left[ \sum_{t \in \Delta(G)} \hat{T}_{i,t}^2 \right] + \mathbb{E} \left[ \sum_{t_1, t_2 \in \Delta(G); t_1 \neq t_2} \hat{T}_{i,t_1} \cdot \hat{T}_{i,t_2} \right]. \end{aligned} \quad (5)$$

Since each  $\hat{T}_{i,t}$  is a 0/1 random variables,  $\hat{T}_{i,t}^2 = \hat{T}_{i,t}$ . Let  $t_1 \neq t_2$  be two triangles in  $\Delta(G)$ . Let  $k$  be the number of distinct vertices they are consisted of, which implies  $4 \leq k \leq 6$ . Then, observe that  $\mathbb{E} [\hat{T}_{i,t_1} \cdot \hat{T}_{i,t_2}] = \hat{p}^k \leq \hat{p}^4$ . We now have all ingredients to upper-bound  $\text{Var} [\hat{T}_i]$ . From (5) and our discussion it follows

$$\text{Var} [\hat{T}_i] \leq T\hat{p}^3 + T^2\hat{p}^4 \leq 2T^2\hat{p}^4, \quad (6)$$

where we used our assumption that  $T \geq 1/\hat{p}$ .



### A.1.3.4 Finalizing the proof

From (5) and (6) we have

$$\text{Var} [\hat{T}] \leq 2RT^2\hat{p}^4.$$

So, from Chebyshev's inequality and (4) we derive

$$\mathbb{P} \left[ |\hat{T} - \mathbb{E} [\hat{T}]| > \varepsilon \mathbb{E} [\hat{T}] \right] < \frac{\text{Var} [\hat{T}]}{\varepsilon^2 \mathbb{E} [\hat{T}]^2} \leq \frac{2RT^2\hat{p}^4}{\varepsilon^2 \hat{p}^6 R^2 T^2} = \frac{2}{\varepsilon^2 \hat{p}^2 R}.$$

Hence, for  $R \geq \frac{20}{\varepsilon^2 \hat{p}^2}$  we get the desired bound.

### A.1.4 Bounding the Number of Light Edges Received by a Machine

We will now bound the probability that any of the induced subgraphs *does not fit* on a machine. To that end, we set a degree threshold  $\tau = \frac{k}{\hat{p}}$ , and define the set of *light* vertices  $V_{\text{light}}$  to be the ones with degree less than  $\tau$ . All other vertices are *heavy*, and we let them comprise the set  $V_{\text{heavy}}$ .

Fix a machine  $M_i$ . We prove that, with probability at least 9/10, the number of edges in  $G[V_i]$  is upper bounded by  $S$ .

We start with analyzing the contribution of the light vertices to the induced subgraphs. We first consider the simpler case of bounding the number of edges in  $G[V_i]$  that have both end-points in  $V_{\text{light}}$ . We refer to such edges as *light edges* and denote them by  $E_{\text{light}}$ . For every edge  $e \in E_{\text{light}}$ , we define a random variable  $Z_e^{(i)}$  as follows.

$$Z_e^{(i)} = \begin{cases} 1 & \text{if } e \in G[V_i], \\ 0 & \text{otherwise.} \end{cases}$$

We let  $Z^{(i)}$  be the sum over all random variables  $Z_e^i$ ,  $Z^i = \sum_{e \in E_{\text{light}}} Z_e^i$ , and we let  $m_\ell$  denote the total number of edges with *light* endpoints in the original graph  $G$ , i.e.,  $m_\ell = |E_{\text{light}}|$ . Due to space constraints, the proof of the following lemma can be found in our full paper [27].

We prove the following lemma.

► **Lemma 21.** *With probability at least 9/10, for every  $i \in [\mathcal{M}]$ ,  $G[V_i]$  contains at most  $\frac{1}{4}S$  light edges.*

We can now use Chebyshev's inequality to conclude that

$$\begin{aligned} \mathbb{P} \left[ |Z^{(i)} - \mathbb{E}[Z^{(i)}]| > S/\sqrt{3} \right] &\leq \frac{\text{Var} [Z^{(i)}]}{S^2/3} \\ \implies \mathbb{P} \left[ Z^{(i)} > 3S/4 \right] &\leq \frac{3}{30S} = \frac{1}{10S} \end{aligned}$$

Finally, we can use union bound over all  $\mathcal{M}$  machines to upper bound the probability that, *any* of the  $Z^{(i)}$  values exceeds  $3S/4$  (using the constraints described in Eq. (3) to simplify).

$$\frac{\mathcal{M}}{10S} = \frac{2000mk}{\varepsilon^2 S} \cdot \frac{1}{10S} \leq \frac{200mk}{\varepsilon^2} \cdot \frac{1}{(15\sqrt{mk}/\varepsilon)^2} = \frac{200mk}{\varepsilon^2 S^2},$$

Therefore, with probability at least 9/10, none of the induced subgraphs  $G[V_i]$  will contain more than  $3S/4$  light edges.

### A.1.5 Bounding the Number of Heavy Edges Received by a Machine

Next, we turn our attention to the edges that have at least one endpoint in  $V_{heavy}$  (we call such edges *heavy*). We will show that for each  $v \in V_{heavy} \cap V_i$ , the number of edges contributed by  $v$  concentrates around its expectation.<sup>6</sup> In this section, we will use  $2m_h$  to denote the total degree of all the heavy vertices i.e.  $2m_h = \sum_{v \in V_{heavy}} d(v)$ . Due to space constraints, we present the proofs of the following theorems in our full paper [27].

► **Theorem 22** (Heavy edges). *With high probability, the number of edges in  $G[V_i]$  that have some endpoint with degree larger than  $\tau$  is at most  $S/8$ .*

Combining this result with Theorem 22, we conclude the following:

► **Theorem 23.** *With probability at least  $9/10$ , the maximum number of edges in any of the  $G[V_i]$ s (where  $i \in [R]$ ) does not exceed  $S$ , and hence Algorithm 3 does not terminate on Line 4.*

### A.1.6 Upper-Bounding the Number of Messages Sent by any Machine

Recalling Algorithm 4, we note that the number of messages *received* by the machine containing  $V_i$ , is equal to the number of edges in  $G[V_i]$ . Therefore, the last section essentially proved that the number of messages (edges) *received* by a particular machine is upper-bounded by  $S$ . Conversely, in this section, we will justify that the number of messages *sent* by any machine is  $O(S)$ . Since the number of edges stored in a machine is  $\leq S$ , it suffices to show that for each edge  $e$ , Algorithm 4 sends only  $O(1)$  messages (each message is a copy of the edge  $e$ ). Our full proofs are included in our full paper [27].

Let  $Z_i^{(e)}$  be the  $\{0, 1\}$  indicator random variable for  $e \in G[V_i]$ , and let  $Z^{(e)}$  be the sum of  $Z_i^{(e)}$  for all  $i \in [M]$ . Here,  $Z^{(e)}$  represents the number of messages that are created by edge  $e$ . Additionally we make  $r = SM/m = O_\varepsilon(\log n)$  copies of each edge  $e$ , and ensure that all replicates reside on the same machine. We distribute the  $Z^{(e)}$  messages evenly amongst the replicates, so that each replica is only responsible for  $Z^{(e)}/r$  messages.

Since all replicates are on the same machine, this last step is purely conceptual, but it will simplify our argument, by allowing us to charge the outgoing messages to each replicate (as opposed to each edge). Our goal will be show that each replicate is responsible for only  $O(1)$  messages, which is the same as showing that w.h.p.  $Z^{(e)}/r = O(1)$ .

Clearly  $\mu = \mathbb{E}[Z^{(e)}] = \hat{p}^2 \cdot \mathcal{M} = \frac{SM}{100mk}$ . With  $\delta = \frac{100e^{1/3}mk^2}{SM}$

$$\mathbb{P}\left[Z^{(e)} > \delta\mu\right] \leq e^{-\lfloor k/2 \rfloor} = \frac{1}{n^3} \implies \mathbb{P}\left[\frac{Z^{(e)}}{r} > \frac{e^{1/3}k}{r}\right] \leq \frac{1}{n^3}$$

Using the assumption (from Eq. (3)) that  $\mathcal{M} > 2000mk/S \implies r > 2000k$ , we see that with high probability, the number of messages sent by any replicate is bounded above by  $e^{1/3}/2000 \leq 1$ . So, the number of messages sent from any machine is bounded by  $S$  with high probability.

### A.1.7 Getting the High Probability Bound

By building on Lemma 20 and Algorithm 3, we design Algorithm 6 that outputs an approximate triangle counting with high probability, as opposed with only constant success probability. It is important to note that in the below algorithm, all  $O(\log n)$  *independent iterations* (Line 3) are done *in parallel*, simultaneously, not sequentially.

<sup>6</sup> Intuitively, this is because  $v$  has high degree, and therefore the number of its sampled neighbors ( $|N(v) \cap V_i|$ ) will concentrate.

■ **Algorithm 6** Approximate Triangle Counting.

---

```

1: function APPROX-TRIANGLES-MAIN($G = (V, E)$)
2: Let $I \leftarrow 100 \cdot \log n$.
3: parfor $i \leftarrow 1 \dots I$ do ▷ Perform all I iterations in parallel simultaneously in $O(1)$
 rounds.
4: Let Y_i be the output of Algorithm 3 invoked on G . We assume that each invocation
 of Algorithm 3 uses fresh randomness compared to previous runs.
5: end parfor
6: Let \mathcal{Y} be the list of all Y_i , for $i = 1 \dots I$.
7: Sort \mathcal{Y} in non-decreasing order.
8: return the median of \mathcal{Y}

```

---

We have the following guarantee for Algorithm 6.

► **Theorem 24.** *Let  $Y$  be the output of Algorithm 6. Then, with high probability it holds*

$$|Y - T| \leq \varepsilon T.$$



In the proof of this theorem we use the following concentration bound.

► **Theorem 25** (Chernoff bound). *Let  $X_1, \dots, X_k$  be independent random variables taking values in  $[0, 1]$ . Let  $X \stackrel{\text{def}}{=} \sum_{i=1}^k X_i$  and  $\mu \stackrel{\text{def}}{=} \mathbb{E}[X]$ . Then, for any  $\delta \in [0, 1]$  it holds  $\mathbb{P}[X \leq (1 - \delta)\mu] \leq \exp(-\delta^2\mu/2)$ .*

# Hardness Results for Weaver’s Discrepancy Problem

Daniel A. Spielman  

Yale University, New Haven, CT, USA

Peng Zhang  

Rutgers University, Piscataway, NJ, USA

---

## Abstract

Marcus, Spielman and Srivastava (Annals of Mathematics 2014) solved the Kadison–Singer Problem by proving a strong form of Weaver’s conjecture: they showed that for all  $\alpha > 0$  and all lists of vectors of norm at most  $\sqrt{\alpha}$  whose outer products sum to the identity, there exists a signed sum of those outer products with operator norm at most  $\sqrt{8\alpha} + 2\alpha$ . We prove that it is NP-hard to distinguish such a list of vectors for which there is a signed sum that equals the zero matrix from those in which every signed sum has operator norm at least  $\eta\sqrt{\alpha}$ , for some absolute constant  $\eta > 0$ . Thus, it is NP-hard to construct a signing that is a constant factor better than that guaranteed to exist.

For  $\alpha = 1/4$ , we prove that it is NP-hard to distinguish whether there is a signed sum that equals the zero matrix from the case in which every signed sum has operator norm at least  $1/4$ .

**2012 ACM Subject Classification** Theory of computation → Problems, reductions and completeness; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Discrepancy Problem, Kadison–Singer Problem, Hardness of Approximation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.40

**Category** APPROX

**Funding** This work was supported in part by NSF Grant CCF-1562041, ONR Award N00014-20-1-2335, and a Simons Investigator Award to Daniel Spielman.

## 1 Introduction

Implicit in Weaver’s [17] conjecture  $KS_2$  is the following discrepancy problem: given vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$ , find an  $\mathbf{x} \in \{\pm 1\}^n$  minimizing the operator norm of  $\sum_i \mathbf{x}(i) \mathbf{v}_i \mathbf{v}_i^*$ , where  $\mathbf{v}_i^*$  is the conjugate <sup>1</sup> transpose of  $\mathbf{v}_i$ . Weaver proved that conjecture  $KS_2$  implies a positive resolution of the Kadison–Singer Problem. It is equivalent<sup>2</sup> to the statement that there are constants  $\alpha > 0$  and  $\beta < 1$  such that for all vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$  that satisfy

$$\|\mathbf{v}_i\|^2 \leq \alpha, \text{ for all } i, \text{ and } \sum_{i=1}^n \mathbf{v}_i \mathbf{v}_i^* = I,$$

there exists a  $\mathbf{x} \in \{\pm 1\}^n$  so that

$$\left\| \sum_i \mathbf{x}(i) \mathbf{v}_i \mathbf{v}_i^* \right\| \leq \beta.$$

---

<sup>1</sup> While all vectors in this paper have Real entries, Weaver’s conjecture remains natural over the Complexes.

<sup>2</sup> Weaver’s statement is slightly different from this, but he proves it is equivalent to this in part b of his Theorem 2. See the Remarks section for some explanation.



© Daniel A. Spielman and Peng Zhang;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 40; pp. 40:1–40:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Here,  $\|v_i\|$  refers to the standard Euclidean norm of a vector and the norm around the signed sum of outer products refers to the operator norm induced by the Euclidean vector norm:

$$\|M\| = \max_{t: \|t\|=1} \|Mt\|.$$

Marcus, Spielman, and Srivastava [14] solved the Kadison–Singer Problem by proving that Weaver's conjecture is true with  $\beta = \sqrt{8\alpha} + 2\alpha$ . Their result was improved by Bownik, Casazza, Marcus, and Speegle [4], who reduced the bound on  $\beta$  to a little below  $\sqrt{8\alpha}$ .

Neither of these results are accompanied by efficient algorithms, and many have wondered if there are efficient algorithms for choosing vectors  $\mathbf{x}$  that satisfy the conditions of Weaver's Conjecture. The currently best known algorithm for constructing such an  $\mathbf{x}$  runs in time  $O(2^{\sqrt[3]{n}/\alpha})$ , and achieves  $\beta$  arbitrarily close to  $\sqrt{8\alpha} + 2\alpha$  [1]. Jourdan, Macgregor, and Sun [11] give a quasipolynomial time algorithm that approximates the best  $\beta$  when the number of vectors is exponential in the dimension.

In this paper, we prove that it is NP-hard to distinguish between the cases in which there exists an  $\mathbf{x}$  that makes the signed sum of outer products the all-0 matrix from the case in which all  $\mathbf{x}$  result in a signed sum with operator norm at least  $\eta\sqrt{\alpha}$ , for some constant  $\eta > 0$ .

Before stating our results in more detail, we introduce some notation that makes those statements compact. Given a list of vectors  $\mathcal{V} = \mathbf{v}_1, \dots, \mathbf{v}_n$  and a vector  $\mathbf{x} \in \{\pm 1\}^n$ , we let  $M(\mathcal{V}, \mathbf{x})$  denote the signed sum of outer products,

$$\sum_i \mathbf{x}(i) \mathbf{v}_i \mathbf{v}_i^*.$$

When just given the list of vectors  $\mathcal{V}$ , we define the minimum achievable operator norm of such a signed sum of outer products to be

$$W(\mathcal{V}) = \min_{\mathbf{x} \in \{\pm 1\}^n} \|M(\mathcal{V}, \mathbf{x})\|.$$

We say that a list of vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$  is  $\alpha$ -Weaver if  $\sum_i \mathbf{v}_i \mathbf{v}_i^* = I$  and  $\|\mathbf{v}_i\|^2 \leq \alpha$  for all  $i$ . In this notation, Weaver's conjecture  $KS_2$  says that for some  $\alpha > 0$  and  $\beta < 1$ , every  $\alpha$ -Weaver list of vectors  $\mathcal{V}$  satisfies  $W(\mathcal{V}) < \beta$ .

We prove that there is a constant  $\eta > 0$  such that for every  $\alpha > 0$  it is NP-hard to distinguish  $\alpha$ -Weaver lists of vectors with  $W(\mathcal{V}) = 0$  from those for which  $W(\mathcal{V}) \geq \eta\sqrt{\alpha}$ . As we know  $W(\mathcal{V}) \leq \sqrt{8\alpha}$ , this result is optimal up to the constant  $\eta$ . Our proof depends on the NP-hardness of approximating Max 2-2 Set Splitting [9, 6]. The factor  $\alpha$  can depend on the number of vectors: we only require  $\alpha \geq \Omega(n^{-1/2})$ . We begin by showing that for 1/4-Weaver vectors  $\mathcal{V}$ , it is NP-hard to distinguish whether  $W(\mathcal{V}) = 0$  or  $W(\mathcal{V}) \geq 1/4$ . Interestingly, this result only depends on the NP-hardness of 2-2 Set Splitting. Jourdan, Macgregor, and Sun [11] independently proved that it is hard to approximate  $W(\mathcal{V})$  for 1/4-Weaver vectors  $\mathcal{V}$ . Their result builds on the NP-hardness of NAE-3SAT, from which the hardness of 2-2 Set Splitting is naturally derived.

Our results are inspired by and analogous to the hardness results for Spencer's Discrepancy Problem established by Charikar, Newman, and Nikolov [7]. Spencer [16] proved that for vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$  in  $\{0, 1\}^n$ , there always exists a  $\mathbf{x} \in \{\pm 1\}^n$  such that  $\|\sum_i \mathbf{x}(i) \mathbf{v}_i\|_\infty \leq 6\sqrt{n}$ . Charikar, Newman, and Nikolov prove that there is a constant  $c$  for which it is NP-hard to distinguish vectors for which this sum can be made zero from those for which the sum always has infinity norm at least  $c\sqrt{n}$ . We follow their lead in deriving hardness from the hardness of approximating Max 2-2 Set Splitting. However, our reduction seems very different from theirs. For Spencer's discrepancy problem, the NP-hardness of approximating Max 2-2 Set

Splitting immediately implies that it is NP-hard to distinguish vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n \in \{0, 1\}^n$  for which there exists an  $\mathbf{x} \in \{\pm 1\}^n$  satisfying  $\sum_{i=1}^n \mathbf{x}(i) \mathbf{v}_i = \mathbf{0}$  from those for which every  $\mathbf{x} \in \{\pm 1\}^n$  must have  $\|\sum_{i=1}^n \mathbf{x}(i) \mathbf{v}_i\|_\infty \geq 2$ . The main challenge of [7] is amplifying this discrepancy gap from 0 vs 2 to 0 vs  $c\sqrt{n}$ . For Weaver's problem, the NP-hardness of 2-2 Set Splitting immediately implies that it is NP-hard to distinguish rank-3 matrices  $A_1, \dots, A_n$  of norm at most  $1/4$  that satisfy  $\sum_{i=1}^n A_i = I$  for which there exists an  $\mathbf{x} \in \{\pm 1\}^n$  satisfying  $\sum_{i=1}^n \mathbf{x}(i) A_i = \mathbf{0}$  from those for which every  $\mathbf{x} \in \{\pm 1\}^n$  must have  $\|\sum_{i=1}^n \mathbf{x}(i) A_i\| \geq 1/2$ . The first challenge in our work is that of turning these rank-3 matrices into rank-1 matrices that satisfy the conditions of Weaver's problem. Our analog of amplification appears when we produce vectors of smaller norm. Another difference between these problems is that we do not know a polynomial time algorithm that approximately solves Weaver's problem, while Bansal [3] showed that Spencer's problem could be approximately solved in polynomial time.

## 2 Notation

We write  $\mathbf{e}_i$  for the elementary unit vector with a 1 in coordinate  $i$ , and we let  $\mathbf{1}$  denote the vector with all entries 1.

As mentioned earlier, we write  $\mathbf{v}^*$  for the conjugate transpose of a vector  $\mathbf{v}$ . As this paper only constructs vectors with Real entries, one can just treat this as the transpose. We let  $\|\mathbf{v}\| = \sqrt{\mathbf{v}^* \mathbf{v}}$  denote the standard Euclidean norm of the vector  $\mathbf{v}$ . Unless otherwise specified, when we write the norm of a matrix we mean the operator norm. We recall that the operator norm of a matrix is at least as large as the operator norm of every one of its submatrices. For a symmetric matrix, the operator norm is the largest absolute value of its eigenvalues.

The other norm we consider of a matrix is its Frobenius norm, written  $\|M\|_F$ , which equals the square root of the sum of the squares of the entries of  $M$ . From the identity  $\|M\|_F^2 = \text{Tr}(MM^*)$ , one can see that the square of the Frobenius norm of  $M$  equals the sum of the squares of the singular values of  $M$ .

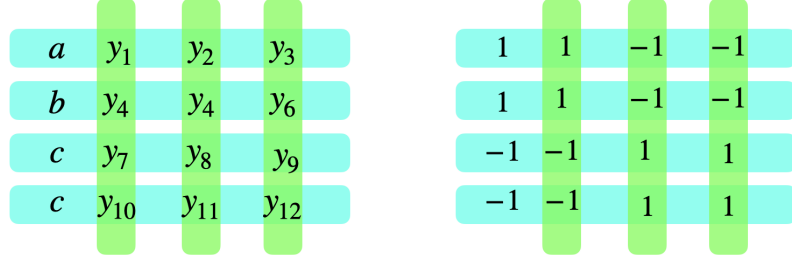
## 3 2-2 Set Splitting

The 2-2 Set Splitting Problem was defined and proved NP-complete by Guruswami [9]. An instance of the problem consists of a list of sets  $S_1, \dots, S_m$ , each of which contains exactly four elements of  $\{1, \dots, n\}$  which we identify with  $\pm 1$  valued variables  $\mathbf{x}(1), \dots, \mathbf{x}(n)$ . A vector  $\mathbf{x} \in \{\pm 1\}^n$  satisfies a set  $S_j$  if  $\sum_{i \in S_j} \mathbf{x}(i) = 0$ , and an  $\mathbf{x}$  satisfies the instance if it satisfies all the sets. We say that an instance is  $\gamma$ -unsatisfiable if for every  $\mathbf{x}$  at least a  $\gamma$  fraction of the sets are unsatisfied. Guruswami proves that for every  $\epsilon > 0$  it is NP-hard to distinguish satisfiable instances from those that are  $(1/12 - \epsilon)$ -unsatisfiable. Charikar, Guruswami, and Wirth [6] observe that Guruswami's construction has the property that there is a constant  $B$  so that no variable appears in more than  $B$  sets.

► **Theorem 1** (Guruswami). *For every  $\epsilon > 0$  there is a constant  $B$  so that for instances of the 2-2 Set Splitting Problem in which every variable appears in at most  $B$  sets, it is NP-hard to distinguish satisfiable instances from those  $1/12 - \epsilon$  unsatisfiable.*

As the fact that there is a constant upper bound on the number of occurrences of each variable is not explicitly stated in [9], we sketch a simple proof with a worse constant in the Appendix.

We define the (3,2-2) Set Splitting Problem to be the restriction of the 2-2 Set Splitting Problem to instances in which every variable appears in at most 3 sets.



■ **Figure 1** A depiction of the equality gadget, and a setting of the variables that satisfies all the clauses when  $a = b = 1$ .

► **Lemma 2.** *The  $(3,2-2)$  Set Splitting Problem is NP-hard. This remains true if we require that no pair of sets intersect in more than one variable. Moreover, there is a constant  $\gamma > 0$  such that it is NP hard to distinguish satisfiable instances of the  $(3,2-2)$  Set Splitting Problem from those that are  $\gamma$ -unsatisfiable.*

The key to proving this lemma is the introduction of an *equality gadget* that forces variables to have the same value. To force variables  $a$  and  $b$  to have the same value, we introduce variables  $c$  and  $y_1, \dots, y_{12}$  and the seven sets

$$\begin{aligned} &\{a, y_1, y_2, y_3\}, \{b, y_4, y_5, y_6\}, \{c, y_7, y_8, y_9\}, \{c, y_{10}, y_{11}, y_{12}\}, \\ &\{y_1, y_4, y_7, y_{10}\}, \{y_2, y_5, y_8, y_{11}\}, \{y_3, y_6, y_9, y_{12}\}. \end{aligned} \quad (1)$$

► **Lemma 3.** *No variable appears in more than 2 of the 7 sets listed in (1), and variables  $a$  and  $b$  each appear once. No pair of these sets intersects in more than 1 variable. If all 7 of the sets are satisfied, then  $a = b$ . And, if  $a = b$  then there is a setting of the remaining variables that satisfies all the sets.*

**Proof.** The Figure 1 shows a setting of the variables that satisfies all the sets in the case that  $a = b = 1$ . If  $a = b = -1$ , we need merely reverse all the signs.

To see that  $a = b$  when these sets are satisfied, note that the last three sets require half of the variables  $y_1, \dots, y_{12}$  to be 1 and half to be  $-1$ . If the first 4 sets are satisfied we can combine this the fact with the double-occurrence of  $c$  to conclude that there must be an even number of 1s among  $a, b, y_1, \dots, y_{12}$ , and so  $a$  can be 1 if and only if  $b$  is as well. ◀

**Proof of Lemma 2.** Let  $S_1, \dots, S_m$  be any instance of the 2-2 Set Splitting problem on variables  $x_1, \dots, x_n$ . We replace each variable with many copies of itself, and use equality gadgets to force all those copies to be equal. More formally, if variable  $x_i$  appears  $k$  times, then we create  $k$  new variables  $x_{i,1}, \dots, x_{i,k}$ , and replace each occurrence of  $x_i$  with one of these. Call the resulting sets on the new variables the *substituted sets*.

We then add  $k - 1$  equality gadgets with distinct extra variables to force  $x_{i,j}$  to equal  $x_{i,j+1}$  for  $1 \leq j < k$ . Each of the variables  $x_{i,j}$  appears in at most 3 sets: one substituted set and one set in each of up to two equality gadgets. The substituted sets are all mutually disjoint, as are the equality gadgets. The only sets that can intersect are inside equality gadgets, or a substituted set and a set in an equality gadget that both contain a variable  $x_{i,j}$ . This would be the only variable in which they intersect.

The derivation of the inapproximability result uses standard techniques, such as those from [9, Section 2.1]. Assume that the input instance is  $1/13$ -unsatisfiable. As each equality



gadget involves 7 sets and the number of equality gadgets is at most  $4m$ , the total number of sets in the new system is at most  $29m$ . For any setting of the variables  $x_{i,j}$ , let  $u_0$  be the number of unsatisfied substituted sets and  $u_1$  be the number of unsatisfied sets in equality gadgets. Call an index  $i$  inconsistent if there exist  $j$  and  $k$  for which  $x_{i,j} \neq x_{i,k}$ . The number of inconsistent indices is at most  $u_1$ . If all the indices were consistent, we would have  $u_0 \geq m/13$ . As each original variable appears in at most  $B$  sets, the number of unsatisfied substituted sets must be at least  $m/13 - Bu_1$ . Thus, the number of unsatisfied sets is at least

$$\max(u_0, u_1) \geq \frac{m}{13B+1},$$

and the new instance is  $\gamma$ -unsatisfiable for  $\gamma \geq 1/(13 \cdot 29 \cdot (B+1))$ .  $\blacktriangleleft$

#### 4 $\alpha = 1/4$

► **Theorem 4.** *Given a list of  $1/4$ -Weaver vectors  $\mathcal{V}$ , it is NP-hard to distinguish whether  $W(\mathcal{V}) = 0$  or  $W(\mathcal{V}) \geq 1/4$ .*

If we were considering sums of arbitrary matrices rather than sums of outer products, we could prove something like Theorem 4 by constructing  $m$ -by- $m$  diagonal matrices  $D_1, \dots, D_n$  such that

$$D_i(j, j) = \begin{cases} 1/4 & \text{if } i \in S_j \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding 2-2 Set Splitting instance is then satisfiable if and only if there exists an  $\mathbf{x} \in \{\pm 1\}^n$  so that  $\sum_i \mathbf{x}(i) D_i = \mathbf{0}$ . In the case where no such sum exists, some entry of the sum must have absolute value at least  $1/2$ . Note that  $\sum_i D_i = I$ . To turn this problem about sums of matrices into an instance of Weaver's problem, we express each  $D_i$  as a sum of orthogonal vectors.

Define

$$\mathbf{q}_1 = \begin{pmatrix} -1/3 \\ 2/3 \\ 2/3 \end{pmatrix}, \quad \mathbf{q}_2 = \begin{pmatrix} 2/3 \\ -1/3 \\ 2/3 \end{pmatrix}, \quad \text{and} \quad \mathbf{q}_3 = \begin{pmatrix} 2/3 \\ 2/3 \\ -1/3 \end{pmatrix}. \quad (2)$$

Observe that each  $\mathbf{q}_i$  is a unit vector, and that

$$\mathbf{q}_1 \mathbf{q}_1^* + \mathbf{q}_2 \mathbf{q}_2^* + \mathbf{q}_3 \mathbf{q}_3^* = I_3.$$

We will use the following special property of these vectors.

► **Lemma 5.** *For every  $\mathbf{z} \in \{\pm 1\}^3$  whose entries are not all equal and for every diagonal matrix  $X$ ,*

$$\left\| X + \sum_i \mathbf{z}(i) \mathbf{q}_i \mathbf{q}_i^* \right\| \geq 1.$$

**Proof.** If one of the entries of  $\mathbf{z}$  differs from the other two, then the matrix  $\sum_i \mathbf{z}(i) \mathbf{q}_i \mathbf{q}_i^*$  is equal to plus or minus a permutation of the matrix

$$R_1 \stackrel{\text{def}}{=} I - 2\mathbf{q}_1 \mathbf{q}_1^* = \frac{1}{9} \begin{pmatrix} 7 & 4 & 4 \\ 4 & 1 & -8 \\ 4 & -8 & 1 \end{pmatrix}.$$

We now show that for every diagonal matrix  $X$ , the operator norm of  $R_1 + X$  is at least 1. Let

$$Y \stackrel{\text{def}}{=} \frac{1}{16} \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & -7 \\ 2 & -7 & 0 \end{pmatrix}.$$

The matrix  $Y$  has inner product 1 with every matrix of the form  $R_1 + X$ . The eigenvalues of  $Y$  are  $\lambda_1 = -1/2$ ,  $\lambda_2 = 1/16$ , and  $\lambda_3 = 7/16$ . But, what really matters is that their absolute values sum to 1. Let corresponding unit-norm eigenvectors be  $\phi_1, \phi_2, \phi_3$ , and recall that  $Y = \sum_i \lambda_i \phi_i \phi_i^*$ . Then for every diagonal  $X$ ,

$$1 = \text{Tr}((R_1 + X)^T Y) = \sum_i \lambda_i \phi_i^* (R_1 + X) \phi_i \leq \sum_i |\lambda_i| \|R_1 + X\| = \|R_1 + X\|. \quad \blacktriangleleft$$

**Proof of Theorem 4.** Let  $S_1, \dots, S_m$  be an instance of the (3,2-2) Set Splitting Problem on variables  $x_1, \dots, x_n$  such that no two sets intersect in more than one variable. Lemma 2 tells us that deciding whether the instance is satisfiable is NP-hard.

As suggested by the discussion before Lemma 5, we would like to construct vectors for each variable in the (3,2-2) Set Splitting Problem whose coordinates correspond to the sets in the instance. To use Lemma 5, we need the vectors we construct to have exactly 3 non-zero coordinates. However, some of the variables in the Set Splitting Problem instance might appear in fewer than 3 sets. To accommodate these, we introduce extra coordinates. For each  $i$  let  $A_i$  be the indices of the sets in which variable  $x_i$  appears. The number of extra coordinates we need is  $b \stackrel{\text{def}}{=} \sum_i 3 - |A_i|$ . Let  $B_1, \dots, B_m$  be a partition of  $\{m+1, \dots, m+b\}$  with  $|B_i| = 3 - |A_i|$ . The set  $B_i$  supplies the extra coordinates we need for variable  $x_i$ , and will be empty if  $x_i$  appears in exactly 3 sets. Let  $A = \{1, \dots, m\}$  and  $B = \cup_i B_i$ .

We now let  $T_i \stackrel{\text{def}}{=} A_i \cup B_i$  be the 3 coordinates associated with variable  $i$ . Define three vectors  $\mathbf{q}_{i,h} \in \mathbb{R}^{m+b}$  to be zero everywhere but on coordinates in  $T_i$ , on which they equal  $(1/2)\mathbf{q}_h$ . Let  $D_i$  be the diagonal matrix that is  $1/4$  on rows and columns indexed by  $T_i$  and 0 elsewhere, so that

$$\mathbf{q}_{i,1} \mathbf{q}_{i,1}^* + \mathbf{q}_{i,2} \mathbf{q}_{i,2}^* + \mathbf{q}_{i,3} \mathbf{q}_{i,3}^* = D_i. \quad (3)$$

For the  $i$  for which  $B_i$  is non-empty, we introduce vectors  $\mathbf{r}_{j,h} = (1/2)\mathbf{e}_j$  for  $j \in B_i$  and  $1 \leq h \leq 3$ .

We now consider Weaver's problem on the list of vectors  $\mathcal{V}$  consisting of  $\{\mathbf{q}_{i,h}\}$  and  $\{\mathbf{r}_{i,h}\}$ . To see that this collection of vectors is  $1/4$ -Weaver, first observe that each vector of form  $\mathbf{q}_{i,h}$  or  $\mathbf{r}_{i,h}$  has norm  $1/2$ . The sum of their outer products is

$$M(\mathcal{V}, \mathbf{1}) = \sum_{1 \leq i \leq n, 1 \leq h \leq 3} \mathbf{q}_{i,h} \mathbf{q}_{i,h}^* + \sum_{j \in B, 1 \leq h \leq 3} \mathbf{r}_{j,h} \mathbf{r}_{j,h}^*.$$

To see that  $M(\mathcal{V}, \mathbf{1})$  is the identity, first observe that all of its off-diagonal entries are zero. For  $j \in A$ , the  $(j, j)$  entry is the sum of  $1/4$  for every variable in set  $S_j$ , and is thus 1. For  $j \in B$ , the  $(j, j)$  entry receives a contribution of  $1/4$  from the sum  $\sum_{1 \leq h \leq 3} \mathbf{q}_{i,h} \mathbf{q}_{i,h}^*$  for the  $i$  such that  $j \in B_i$ , and another  $1/4$  from each  $\mathbf{r}_{j,h} \mathbf{r}_{j,h}^*$  where  $1 \leq h \leq 3$ .

Let  $\mathbf{z}(i, h)$  be the sign for the outer product  $\mathbf{q}_{i,h} \mathbf{q}_{i,h}^*$  and let  $\mathbf{w}(j, h)$  be the sign for the outer product  $\mathbf{r}_{j,h} \mathbf{r}_{j,h}^*$ , and extend the definition of  $M$  so that we can write the signed sum of outer products as  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$ .

If the (3,2-2) Set Splitting instance is satisfied by  $\mathbf{x}$ , then set  $\mathbf{z}(i, h) = \mathbf{x}(i)$  for each  $i$ , and for each  $j$  in a non-empty  $B_i$ , set  $\mathbf{w}(j, 1) = \mathbf{x}(i)$  and  $\mathbf{w}(j, 2) = \mathbf{w}(j, 3) = -\mathbf{x}(i)$ . This causes the signed sum of the outer products of the vectors to be the zero matrix:

$$\begin{aligned} M(\mathcal{V}, \mathbf{z}, \mathbf{w}) &= \sum_{1 \leq i \leq n, 1 \leq h \leq 3} \mathbf{z}(i, h) \mathbf{q}_{i,h} \mathbf{q}_{i,h}^* + \sum_{1 \leq i \leq n} \sum_{j \in B_i, 1 \leq h \leq 3} \mathbf{w}(j, h) \mathbf{r}_{j,h} \mathbf{r}_{j,h}^* \\ &= \sum_{1 \leq i \leq n, 1 \leq h \leq 3} \mathbf{x}(i) \mathbf{q}_{i,h} \mathbf{q}_{i,h}^* - \frac{1}{4} \sum_{1 \leq i \leq n, j \in B_i} \mathbf{x}(i) \mathbf{e}_j \mathbf{e}_j^* \\ &= \sum_{1 \leq i \leq n} \mathbf{x}(i) D_i - \frac{1}{4} \sum_{1 \leq i \leq n, j \in B_i} \mathbf{x}(i) \mathbf{e}_j \mathbf{e}_j^* \\ &= 0 \end{aligned}$$

where the last equation holds since the (3,2-2) Set Splitting instance is satisfied by  $\mathbf{x}$ .

If the (3,2-2) Set Splitting instance is unsatisfiable, we will show that for every  $\mathbf{z}$  and  $\mathbf{w}$ , the norm of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  is at least  $1/4$ . We break our analysis into two cases. In the first, we examine what happens if there exists a vector  $\mathbf{x}$  so that for all  $i$  and  $h$ ,  $\mathbf{z}(i, h) = \mathbf{x}(i)$ . In this case,

$$\sum_{1 \leq i \leq n, 1 \leq h \leq 3} \mathbf{z}(i, h) \mathbf{q}_{i,h} \mathbf{q}_{i,h}^* = \sum_{1 \leq i \leq n} \mathbf{x}(i) D_i.$$

As the (3,2-2) Set Splitting instance is not satisfied by  $\mathbf{x}$ , there must be some set  $j$  for which the absolute value of sum of  $\mathbf{x}(i)$  for  $i \in S_j$  is at least 2, and thus the  $(j, j)$  entry of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  has absolute value at least  $1/2$ . As the operator norm of a matrix is at least the absolute value of its largest diagonal, in this case the norm of the signed sum must be at least  $1/2$ .

In the other case there is some  $i$  for which not all of the  $\mathbf{z}(i, h)$  are equal. Now, consider the entries of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  that appear in rows and columns indexed by  $T_i$ . As each pair of sets  $T_i$  and  $T_k$  can intersect in at most one element for  $i \neq k$ , the off-diagonal entries of this submatrix are equal to the off-diagonals of  $\sum_{1 \leq h \leq 3} \mathbf{z}(i, h) \mathbf{q}_h \mathbf{q}_h^*$ . Regardless of the diagonals, Lemma 5 tells us that this submatrix has operator norm at least  $1/4$ , and thus  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  does as well.  $\blacktriangleleft$

## 5 General $\alpha$

► **Theorem 6.** *There exists a constant  $\eta > 0$  so that for every integer  $k \geq 2$  it is NP-hard to distinguish a list of  $1/2k$ -Weaver vectors  $\mathcal{W}$  for which  $W(\mathcal{W}) = 0$  from those for which  $W(\mathcal{W}) \geq \eta/\sqrt{k}$ .*

The proof employs two reductions, the first of which is a variation of the one used in the previous section. When this reduction is applied to a  $\gamma$ -unsatisfiable (3,2-2) Set Splitting instance, it produces a set of vectors  $\mathcal{V}$  so that for all  $\mathbf{x}$ , a constant fraction of the diagonals of  $M(\mathcal{V}, \mathbf{x})$  have absolute value at least  $1/50$ . The second reduction converts these into instances of  $1/2k$ -Weaver vectors such that every signed sum of those vectors has operator norm at least  $\eta/\sqrt{k}$ .

In the first reduction, we use the following four orthogonal vectors:

$$\mathbf{q}_1 \stackrel{\text{def}}{=} \frac{1}{5} \begin{pmatrix} 1 \\ 4 \\ -2 \\ -2 \end{pmatrix} \quad \mathbf{q}_2 \stackrel{\text{def}}{=} \frac{1}{5} \begin{pmatrix} 4 \\ 1 \\ 2 \\ 2 \end{pmatrix} \quad \mathbf{q}_3 \stackrel{\text{def}}{=} \frac{1}{5} \begin{pmatrix} -2 \\ 2 \\ -1 \\ 4 \end{pmatrix} \quad \mathbf{q}_4 \stackrel{\text{def}}{=} \frac{1}{5} \begin{pmatrix} -2 \\ 2 \\ 4 \\ -1 \end{pmatrix}.$$

► **Lemma 7.** *For every  $\mathbf{z} \in \{\pm 1\}^4$  that doesn't equal  $\pm \mathbf{1}$ , for every  $\mathbf{w} \in \{\pm 1\}^3$ , and for every  $1 \leq j \leq 4$ ,*

$$\left| \sum_{i=1}^4 (1/4) z(i) (\mathbf{q}_i(j))^2 + \sum_{h=1}^3 (1/4) \mathbf{w}(h) \right| \geq 1/50.$$

**Proof.** The multiset of values of  $(\mathbf{q}_i(j))^2$  as  $i$  varies from 1 through 4 is  $(1/25, 4/25, 4/25, 16/25)$ . Thus, every non-constant signed sum of these numbers must be an odd multiple of  $1/25$  with absolute value less than 1 and every non-constant signed sum of  $1/4$  times these numbers must have absolute value between  $1/100$  and  $23/100$ . As the term  $\sum_{h=1}^3 (1/4) \mathbf{w}(h)$  can only take values in  $\{\pm 25/100, \pm 75/100\}$ , the total sum must have absolute value at least  $2/100 = 1/50$ . ◀

We model our first reduction on the one from the previous section, but using these vectors. Let  $S_1, \dots, S_m$  be an instance of the (3,2-2) Set Splitting Problem on variables  $x_1, \dots, x_n$ . For each  $i$  let  $A_i$  be the indices of the sets in which variable  $x_i$  appears. If variable  $x_i$  appears in  $k$  sets, introduce  $4 - k$  new coordinates for that variable, and call the set of them  $B_i$ . As  $k \leq 3$ ,  $B_i$  will not be empty. Let  $T_i = A_i \cup B_i$ . Define four vectors  $\mathbf{q}_{i,h}$  that are zero everywhere except on coordinates in  $T_i$ , on which they equal  $(1/2) \mathbf{q}_h$ . For each variable and each  $j \in B_i$ , we introduce vectors  $\mathbf{r}_{j,h} = (1/2) \mathbf{e}_{j,h}$  for  $j \in B_i$  and  $1 \leq h \leq 3$ . Let  $\mathcal{V}$  consist of the vectors  $\{\mathbf{q}_{i,h}\}$  and  $\{\mathbf{r}_{j,h}\}$ . This collection of vectors is  $1/4$ -Weaver. Let  $A = \{1, \dots, m\}$ ,  $B = \cup_i B_i$ , and note that  $|B| \leq 3m$ .

► **Lemma 8.** *Let  $\mathbf{z}(i, h)$  be  $\{\pm 1\}$  variables for  $1 \leq i \leq n$  and  $1 \leq h \leq 4$ . Also let  $\mathbf{w}(j, h)$  be in  $\pm 1$  for  $j \in B$  and  $1 \leq h \leq 3$ . If there are  $k$  values of  $i$  for which  $\mathbf{z}(i, h)$  is not constant over  $h$ , the matrix  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  must have at least  $k$  diagonal entries in columns in  $B$  with absolute value at least  $1/50$ .*

**Proof.** For every  $i$  for which  $\mathbf{z}(i, h)$  is not constant over  $h$ , Lemma 7 tells us that every diagonal indexed by  $B_i$  must have absolute value at least  $1/50$ . ◀

► **Lemma 9.** *Let  $\mathcal{V}$  be the vectors produced by this reduction on a (3,2-2) Set Splitting Problem instance. Every vector in  $\mathcal{V}$  has at most 4 non-zero entries, and no coordinate is in the support of more than 7 of the vectors. If the set splitting instance is satisfiable, then  $W(\mathcal{V}) = 0$ . If the set splitting instance is  $\gamma$ -unsatisfiable, then for every  $\mathbf{z}$  and  $\mathbf{w}$ , at least a  $\gamma/12$  fraction of the diagonal entries of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  must have absolute value at least  $1/50$ .*

**Proof.** If the set splitting instance is satisfiable, let  $\mathbf{x}$  be the vector that satisfies it. We then set  $\mathbf{z}(i, h) = \mathbf{x}(i)$  for each  $i$ , and for each  $j$  in  $B_i$  we set  $\mathbf{w}(j, 1) = \mathbf{x}(i)$  and  $\mathbf{w}(j, 2) = \mathbf{w}(j, 3) = -\mathbf{x}(i)$ . With this signing,  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  becomes the all-0 matrix.

Now, assume that the set splitting instance is  $\gamma$ -unsatisfiable. Let  $K$  be the set of  $i$  for which  $\mathbf{z}(i, h)$  is not constant in  $h$ . That is, for which there exist  $h$  and  $\tilde{h}$  for which  $\mathbf{z}(i, h) \neq \mathbf{z}(i, \tilde{h})$ . Lemma 8 tells us that at least  $k = |K|$  of the diagonals of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  indexed by  $B$  have absolute value at least  $1/50$ . As the dimension of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  is at most  $4m$ , it suffices to prove that at least  $(\gamma/3)m$  of its diagonals have absolute value at least  $1/50$ .

If  $k \geq (\gamma/3)m$ , this finishes the proof. If not, define  $\hat{\mathbf{z}}(i, h) = \mathbf{z}(i, 1)$  for  $i \notin K$ , and  $1 \leq h \leq 4$ , and set  $\hat{\mathbf{z}}(i, h) = 1$  for  $i \in K$ . As the set splitting instance is  $\gamma$ -unsatisfiable, at least  $\gamma m$  of the diagonals of  $M(\mathcal{V}, \hat{\mathbf{z}}, \mathbf{w})$  in columns in  $A$  have absolute value at least  $1/4$ . It remains to see how these diagonals change between  $\hat{\mathbf{z}}$  and  $\mathbf{z}$ .

For each  $i$ , the variables  $\mathbf{z}(i, h)$  only appear in 3 diagonals indexed by  $A$ . So,  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  and  $M(\mathcal{V}, \widehat{\mathbf{z}}, \mathbf{w})$  can differ in at most  $3k$  diagonals in columns in  $A$ . Thus, at least  $\gamma m - 3k$  diagonals of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  in columns of  $A$  have absolute value at least  $1/4$ . In total, we find that the number of diagonals that have absolute value at least  $1/50$  is at least

$$\gamma m - 3k + k \geq (\gamma/3)m, \quad \text{for } k \leq (\gamma/3)m. \quad \blacktriangleleft$$

Fix an integer  $k$ . We let  $\Pi$  be a  $(k-1)$ -by- $k$  matrix whose rows are an orthonormal basis of the nullspace of the all-1 vector in  $k$  dimensions. Let  $B$  be the  $k$ -by- $\binom{k}{2}$  matrix whose columns contain all  $\binom{k}{2}$  vectors with two non-zero entries, the first of which is 1 and the second of which is  $-1$ . Our reduction uses the matrix  $G \stackrel{\text{def}}{=} \Pi B / \sqrt{k}$ .

► **Lemma 10.** *The matrix  $G$  is a  $(k-1)$ -by- $\binom{k}{2}$  matrix such that*

- a. *every column of  $G$  has norm  $\sqrt{2/k}$ ,*
- b.  *$GG^* = I$ , and*
- c. *for every  $\binom{k}{2}$ -dimensional square diagonal matrix  $D$ ,*

$$\|GDG^*\| \geq \frac{1}{k} \sqrt{\frac{2}{k-1}} \|D\|_F,$$

where  $\|D\|_F$  is the Frobenius norm of  $D$  – the square root of the sum of the squares of its entries.

**Proof.** As every column of  $B$  is orthogonal to the all-ones vector, so multiplying by  $\Pi$  does not change its norm. As these columns have norm  $\sqrt{2}$ , the columns of  $G$  have norm  $\sqrt{2/k}$ .

To compute  $GG^*$ , first observe that  $BB^* = kI - J$ , where  $J$  is the all-ones matrix of dimension  $k$ . This matrix has eigenvalue  $k$  with multiplicity  $k-1$  and one eigenvalue of 0. As  $\Pi$  is a projection orthogonal to the nullspace of  $B$ ,  $\Pi BB^* \Pi$  equals  $kI_{k-1}$ .

Every diagonal entry of  $D$  appears twice as an off-diagonal of the matrix  $BDB^*$ . One easy way to see this is to index the columns of  $B$  by pairs  $(i, j)$  with  $i < j$ , where column  $(i, j)$  equals  $\mathbf{e}_i - \mathbf{e}_j$ . If we index the diagonal entries of  $D$  similarly and label them  $d_{i,j}$  we have

$$BDB^* = \sum_{i < j} d_{i,j} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^*.$$

Thus,

$$\|BDB^*\|_F^2 \geq 2 \|D\|_F^2.$$

As the columns of  $B$  have sum 0, they lie in the span of the rows of  $\Pi$ . So,

$$\|\Pi B D (\Pi B)^*\|_F^2 = \|BDB^*\|_F^2,$$

and we may conclude that

$$\|GDG^*\|_F^2 \geq \frac{1}{k^2} \|BDB^*\|_F^2.$$

As the Frobenius norm is the sum of the squares of the  $(k-1)$  eigenvalues of  $GDG^*$ ,

$$\|GDG^*\|_2^2 \geq \frac{1}{k-1} \|GDG^*\|_F^2 \geq \frac{1}{(k-1)k^2} \|BDB^*\|_F^2 \geq \frac{2}{(k-1)k^2} \|D\|_F^2. \quad \blacktriangleleft$$

We now describe the second reduction. Let  $\mathcal{V}$  be the set of vectors produced by the first reduction and described by Lemma 9, and let  $m_1$  be the dimension of the space in which they reside. We now partition the coordinates of these vectors,  $\{1, \dots, m_1\}$  into at most 22 classes so that for each vector and each class, the vector has at most one non-zero entry a coordinate in that class. To see that this is possible, and that such a partition is computable efficiently, note this is a problem of 22-coloring a graph with maximum degree at most 21: the vertices are the coordinates, the edges go between coordinates that are in the support of the same vector, and the graph has degree at most 21. So, a greedy coloring algorithm will do the job. Let  $C_1, \dots, C_{22}$  be the classes of coordinates, and let  $c_i = |C_i|$  for each  $i$ .

Given a choice of  $k$ , we would like to partition each class  $C_i$  into sets of size  $\binom{k}{2}$ . As this is not necessarily possible, for each  $i$  let  $a_i$  be the integer between 0 and  $\binom{k}{2} - 1$  so that  $c_i + a_i$  is divisible by  $\binom{k}{2}$ , and let  $a = \sum_i a_i$ . We add  $a$  additional coordinates, and assign  $a_i$  of them to class  $C_i$  for each  $i$ . Let  $m_2 = m_1 + a$  be the number of coordinates after these are added. We then create a new list of vectors,  $\mathcal{U}$  by embedding each vector of  $\mathcal{V}$  into the  $m_2$  dimensional space by setting each extra coordinate to 0, and for each of the  $a$  new coordinates,  $j$ , adding 4 vectors  $\mathbf{r}_{j,h} = (1/2)\mathbf{e}_j$  for  $1 \leq h \leq 4$ . The list of vectors  $\mathcal{U}$  is  $(1/4)$ -Weaver.

If  $W(\mathcal{V}) = 0$ , then  $W(\mathcal{U}) = 0$  as well: use the same signing for each vector derived from  $\mathcal{V}$ , and then for each new coordinate  $j$  assign half of the  $\mathbf{r}_{j,h}$  a positive sign and half a negative sign.

Now, partition each class of coordinates into groups of size  $\binom{k}{2}$ , and call the resulting  $l \stackrel{\text{def}}{=} m_2 / \binom{k}{2}$  classes  $D_1, \dots, D_l$ . We now describe a rectangular matrix  $F$  with  $m_2$  columns and  $(k-1)l$  rows. Partition the rows of  $F$  into  $l$  sets of size  $k-1$ , which we call  $E_1, \dots, E_l$ . This partition can be arbitrary, but to ease visualization one could make each set consecutive. We define  $F$  to be zero everywhere, except on submatrices consisting of rows indexed by  $E_i$  and the columns indexed by  $D_i$ , on which it equals  $G$ . The final set of vectors produced by our reduction,  $\mathcal{W}$ , is the result of multiplying each vector in  $\mathcal{U}$  by  $F$ .

► **Lemma 11.** *Let  $\mathcal{V}$  be the set of vectors produced by the first reduction and analyzed in Lemma 9. Let  $m_1$  be the dimension of the space in which the vectors in  $\mathcal{V}$  lie, and assume that  $m_1 \geq 22\binom{k}{2}$ . Let  $\mathcal{W}$  be the result of the second reduction. The vectors  $\mathcal{W}$  are  $1/2k$ -Weaver. If  $W(\mathcal{V}) = 0$ , then  $W(\mathcal{W}) = 0$ . If for every  $\pm 1$  vector  $\mathbf{x}$  at least a  $\phi$  fraction of the diagonals of  $M(\mathcal{V}, \mathbf{x})$  have absolute value greater than  $\delta$ , then*

$$W(\mathcal{W}) \geq \delta \sqrt{\frac{\phi}{2k}}$$

**Proof.** We exploit the algebraic characterization of the second reduction:

$$M(\mathcal{W}, \mathbf{x}) = FM(\mathcal{U}, \mathbf{x})F^*.$$

This immediately tells us that an  $\mathbf{x}$  that makes the right side zero will also make the left side zero. It also implies that for every  $i$  the submatrix of  $M(\mathcal{U}, \mathbf{x})$  indexed by rows and columns in  $D_i$  is diagonal. This is because every vector in  $\mathcal{U}$  has at most one nonzero entry indexed by  $D_i$ , and the matrix  $M(\mathcal{U}, \mathbf{x})$  is a signed sum of outer products of vectors in  $\mathcal{U}$ .

To see that  $\mathcal{W}$  is  $1/2k$ -Weaver, we first compute the norms of these vectors. As the non-zero entries of each vector in  $\mathcal{U}$  appear in disjoint blocks, and every column of  $F$  has norm  $\sqrt{2/k}$ , the squared norm of  $F$  times any vector in  $\mathcal{U}$  is  $(2/k)$  times the squared norm of that vector:  $(1/4)(2/k) = 1/2k$ . Also note that  $FF^* = I$ , so

$$M(\mathcal{W}, \mathbf{1}) = FM(\mathcal{U}, \mathbf{1})F^* = FIF^* = I.$$

Consider a vector  $\mathbf{x}$  for which at least a  $\phi$  fraction of the diagonals of  $M(\mathcal{V}, \mathbf{x})$  have absolute value at least  $\delta$ . Note that  $a \leq 22\binom{k}{2}$ , so the assumption that  $m_1 \geq 22\binom{k}{2}$  implies  $m_2 \leq 2m_1$ . This means that at least a  $\phi/2$  fraction of the diagonals of  $M(\mathcal{U}, \mathbf{x})$  have absolute value at least  $\delta$ . As the sets  $D_1, \dots, D_l$  partition the columns of this matrix, there must be some set of columns  $D_i$  such that at least a  $\phi/2$  fraction of the diagonals in the rows and columns indexed by  $D_i$  have absolute value at least  $\delta$ . Call this submatrix  $M_i$ , and notice that it has squared Frobenius norm at least  $\binom{k}{2}\delta^2\phi/2$ . So,

$$\|M(\mathcal{W}, \mathbf{x})\| = \|FM(\mathcal{U}, \mathbf{x})F^*\| \geq \|GM_iG^*\| \geq \frac{1}{k}\sqrt{\frac{2}{k-1}}\|M_i\|_F \geq \delta\sqrt{\frac{\phi}{2k}}$$

where the second-to-last inequality follows from part c of Lemma 10. This implies  $W(\mathcal{W}) \geq \delta\sqrt{\frac{\phi}{2k}}$ .  $\blacktriangleleft$

**Proof of Theorem 6.** On input an instance of the (3,2-2) Set Splitting Problem, let  $\mathcal{V}$  be the set of vectors produced by the first reduction, and let  $\mathcal{W}$  be the set of vectors produced by the second. By applying Lemmas 9 and 11, we see that if the instance is satisfiable, then  $W(\mathcal{V}) = W(\mathcal{U}) = W(\mathcal{W}) = 0$ . On the other hand, if the instance is  $\gamma$ -unsatisfiable, then Lemma 9 implies that for all  $\pm 1$  vectors  $\mathbf{z}$  and  $\mathbf{w}$  at least a  $\phi = \gamma/12$  fraction of the diagonal entries of  $M(\mathcal{V}, \mathbf{z}, \mathbf{w})$  have absolute value at least  $\delta = 1/50$ . Lemma 11, then allows us to conclude that

$$W(\mathcal{W}) \geq \frac{1}{50}\sqrt{\frac{\gamma}{24k}} = \frac{\eta}{\sqrt{k}}, \quad \text{where } \eta \stackrel{\text{def}}{=} \frac{1}{100}\sqrt{\frac{\gamma}{6}}.$$

So, the problem of distinguishing whether a (3,2-2) set splitting instance is satisfiable or  $\gamma$ -unsatisfiable is polynomial-time reducible to the problem of distinguishing a set of  $1/2k$ -Weaver vectors  $\mathcal{W}$  with  $W(\mathcal{W}) = 0$  from a set for which  $W(\mathcal{W}) \geq \eta/\sqrt{k}$ .  $\blacktriangleleft$

We remark that this construction can be carried out whenever the original (3,2-2) Set Splitting instances has a number of sets that exceeds  $22\binom{k}{2}$ . This will result in a number of vectors that is at most a constant times the number of sets. Thus, we only require that  $k$  be at least some constant times the square root of the number of vectors.

## 6 Remarks

We first emphasize that our hardness results do not say that it is hard to find an  $\mathbf{x}$  giving an operator norm at or above the guarantee provided by [14, 4]. We only prove that it is hard to improve on this guarantee by a constant factor.

The original form of Weaver's conjecture  $KS_2$  states that there exist constants  $\alpha > 0$  and  $\beta < 1$  such that for vectors  $\mathbf{v}_i$  of norm at most  $\sqrt{\alpha}$  whose outer products have sum with operator norm less than 1, there exists a partition of those vectors into two sets so that in each set the sum of the outer products has operator norm at most  $\beta$ . These vectors could differ from those in  $\alpha$ -Weaver position in that the sum of their outer products does not need to equal the identity. Weaver proved that the conjecture is unchanged if one requires the sum of the outer products of the vectors to be the identity. Instead of considering the sum of the outer products in each set, we consider the difference of the sum of the outer products by assigning a  $+1$  to every vector in one set and a  $-1$  to every vector in the other. However, when we consider such signed sums the condition that the sum of the outer products is the identity is no longer equivalent to the condition that the sum has operator norm at most 1. To prove an upper bound on the discrepancy for vectors of bounded norm whose sum



of outer products has operator norm at most 1, one can use the results of Kyng, Luh, and Song [12]. Instead of outer products of vectors, Cohen [8] and Brändén [5] have shown that it is possible to prove analogous discrepancy results for sums of positive semidefinite matrices of bounded trace.

One may wonder what to make of our results when the vectors  $\mathcal{W}$  produced are not rational, because it is not clear that they can be represented exactly, and thus their representation in floating point might not be precisely  $\alpha$ -Weaver. One way to fix this is to round them to floating point numbers, and then apply a linear transformation that forces the sum of their outer products to be the identity. If done with enough precision, this will cause their norms to increase negligibly. We also observe that the vectors can be made rational whenever  $k$  is a square. If  $k = s^2$ , then one can choose  $\Pi$  to be the horizontal concatenation of the vector  $-\mathbf{1}_{k-1}/s$  with the matrix  $I_{k-1} - J_{k-1}(s/(k(s+1)))$ , where  $J_{k-1}$  is the  $(k-1)$ -dimensional square matrix with all entries 1.

---

## References

- 1 Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Nikhil Srivastava. Approximating the largest root and applications to interlacing families. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1015–1028. SIAM, 2018.
- 2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.
- 3 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 3–10. IEEE, 2010.
- 4 Marcin Bownik, Pete Casazza, Adam W Marcus, and Darrin Speegle. Improved bounds in weaver and feichtinger conjectures. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 2019(749):267–293, 2019.
- 5 Petter Brändén. Hyperbolic polynomials and the Kadison-Singer problem. *arXiv preprint*, 2018. [arXiv:1809.03255](https://arxiv.org/abs/1809.03255).
- 6 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.
- 7 Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for minimizing discrepancy. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1607–1614. SIAM, 2011.
- 8 Michael B. Cohen. Improved spectral sparsification and Kadison-Singer for sums of higher-rank matrices, 2016. URL: <http://www.birs.ca/events/2016/5-day-workshops/16w5111/videos/watch/201608011534-Cohen.html>.
- 9 Venkatesan Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. *Algorithmica*, 38(3):451–469, 2004.
- 10 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- 11 Ben Jourdan, Peter Macgregor, and He Sun. Is the algorithmic kadison-singer problem hard? *arXiv preprint*, 2022. [arXiv:2205.02161](https://arxiv.org/abs/2205.02161).
- 12 Rasmus Kyng, Kyle Luh, and Zhao Song. Four deviations suffice for rank 1 matrices. *Advances in Mathematics*, 375:107366, 2020. doi:10.1016/j.aim.2020.107366.
- 13 A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- 14 Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families ii: Mixed characteristic polynomials and the kadison—singer problem. *Annals of Mathematics*, pages 327–350, 2015.

- 15 G. A. Margulis. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24(1):39–46, July 1988.
- 16 Joel Spencer. Six standard deviations suffice. *Transactions of the American mathematical society*, 289(2):679–706, 1985.
- 17 Nik Weaver. The Kadison–Singer problem in discrepancy theory. *Discrete mathematics*, 278(1-3):227–239, 2004.

## A Hardness of 2-2 Set Splitting

The purpose of this section is to sketch a simple proof that it is NP-hard to distinguish satisfiable (3,2-2) set splitting instances from  $\gamma$ -unsatisfiable ones, for some constant  $\gamma > 0$ .

We first sketch a proof that 2-2 Set Splitting is NP-hard. We then explain why it is hard to distinguish satisfiable instances from  $\gamma$ -unsatisfiable ones, for some constant  $\gamma > 0$ , even when each variable appears in at most a constant number of sets.

Our notation follows that of Håstad [10] and Guruswami [9]. Whereas the purpose of those papers is to obtain tight hardness of approximation results, our purpose in this appendix is just to obtain simple proofs of hardness up to some constant.

We begin by recalling the NP-hardness of E3-SAT: 3-SAT in which every clause contains exactly 3 distinct variables. We will reduce this to NAE-E3-SAT, where we recall that the NAE-SAT problem consists of not-all-equal clauses that are satisfied when their terms are not all equal, and NAE- $Ek$ -SAT is the restriction of NAE-SAT to instances in which every clause contains exactly  $k$  distinct variables.

The standard reduction from E3-SAT to NAE-E4-SAT is obtained by creating one extra variable,  $z$ , and replacing every clause in a SAT instance with an NAE clause that contains the same terms along with  $z$ . If the SAT instance is satisfied by an assignment  $\mathbf{x}$ , then the NAE-SAT instance is satisfied by the same assignment and  $z$  set to false. Conversely, observe that satisfying assignments of NAE-SAT instances remain satisfying if one negates all the variables. So, if the NAE-E4-SAT instance is satisfiable, we may assume that  $z$  is false and that the remaining variables provide a satisfying assignment to the SAT instance.

We then reduce the NAE-E4-SAT instance to an NAE-E3-SAT instance by splitting up each NAE clause. For each NAE clause with terms  $t_1, t_2, t_3, t_4$ , we introduce one new variable  $y$ , and then replace the clause with two clauses: one with terms  $t_1, t_2, y$  and one with  $\bar{y}, t_3, t_4$ .

To reduce NAE-E3-SAT to 2-2 Set Splitting, we first show that we can reduce it to an NAE-E3-SAT problem in which no variable is negated in any NAE clause. We may accomplish this by introducing a gadget that forces variables to be the negations of each other. To force variables  $x$  and  $y$  to be negations of each other, we introduce extra variables  $a$ ,  $b$ , and  $c$ , and include the NAE-E3 clauses

$$(x, y, a), (x, y, b), (x, y, c), (a, b, c).$$

If  $x$  is the negation of  $y$ , then these clauses are satisfied by any choice of  $a$ ,  $b$ , and  $c$  that are not all equal. Conversely, if  $a$ ,  $b$ , and  $c$  are not all equal then these clauses can only be satisfied if  $x$  differs from  $y$ .

Finally, we may reduce NAE-E3-SAT to 2-2 Set Splitting by appending one extra variable to every clause, and making the result a set to be split.

If every variable occurs at most a constant number of times in the original E3-SAT instance, then every variable will occur at most a constant number of times in the 2-2 Set Splitting instance, except for the variable  $z$  which was added in the reduction from E3-SAT to NAE-E4-SAT. To fix this, we replace the variable  $z$  with many variables, and then force them

all to be equal. In particular, if the E3-SAT instance has  $m$  clauses, we introduce variables  $z_1, \dots, z_m$ , and add one to each clause to create an NAE-E4-SAT clause. We must then introduce gadgets that force those variables to be equal. For simplicity, for each  $1 \leq j < m$ , we could introduce a new variable  $w_j$ , and include the NAE-E3 clauses that force  $z_j \neq w_j$  and  $w_j \neq z_{j+1}$ . Of course, we do not need to split these clauses when we reduce the other NAE-E4-SAT clauses to NAE-E3-SAT clauses.

To preserve constant-factor unsatisfiability, we add more constraints than this to the variables  $z_1, \dots, z_m$ . First, we recall the formulation by Håstad [10, Theorem 2.24] of one of the main results of Arora *et. al.* [2]:

► **Theorem 12.** *There exists a constant  $c > 0$  such that it is NP-hard to distinguish a satisfiable E3-SAT instance in which every variable appears in at most 5 clauses from one that is  $c$ -unsatisfiable.*

The reductions we have described so far convert satisfiable E3-SAT instances to satisfiable 2-2 set splitting instances, and they ensure that if each variable appears at most 5 times in the original instance, then each variable appears in at most a constant number of sets in the set splitting instance. To make sure that each  $c$ -unsatisfiable E3-SAT instance is converted into a  $c'$ -unsatisfiable NAE-E3-SAT instance, we impose equality relations between  $z_1, \dots, z_m$  in the pattern of an expander graph.

For example, we could use a 4-regular Ramanujan graph [15, 13] on  $m$  or slightly more than  $m$  vertices. If the graph has exactly  $m$  vertices, then for every edge  $(i, j)$  in the graph, we use the gadgets described above to force  $z_i = z_j$ . If the graph has more than  $m$  vertices, when we introduce even more copies of  $z$  so that we have one for each vertex, and then proceed as before. The gadgets ensure that if more than  $k$  of the copies of  $z$  differ from the majority, then at least  $\omega k$  of the clauses in the gadgets will be unsatisfied, for some  $\omega > 0$ . If the E3-SAT instance is  $c$ -unsatisfiable and only a small enough fraction of the copies of  $z$  disagree with the majority, then some constant fraction of the other NAE-E3-SAT clauses must be unsatisfied.

As the other parts of the reduction only involve a constant number of locally substituted clauses, we may prove as in Lemma 2 that the  $c$ -unsatisfiable E3-SAT instances become constant-unsatisfiable 2-2 set splitting instances. As each variable appears in at most a constant number of sets, we can then use Lemma 2 to ensure that each variable occurs in at most three sets.

# Relative Survivable Network Design

Michael Dinitz ✉

Johns Hopkins University, Baltimore, MD, USA

Ama Koranteng ✉

Johns Hopkins University, Baltimore, MD, USA

Guy Kortsarz ✉

Rutgers University, Camden, NJ, USA

---

## Abstract

One of the most important and well-studied settings for network design is edge-connectivity requirements. This encompasses uniform demands such as the Minimum  $k$ -Edge-Connected Spanning Subgraph problem ( $k$ -ECSS), as well as nonuniform demands such as the Survivable Network Design problem. A weakness of these formulations, though, is that we are not able to ask for fault-tolerance larger than the connectivity. Taking inspiration from recent definitions and progress in graph spanners, we introduce and study new variants of these problems under a notion of *relative* fault-tolerance. Informally, we require not that two nodes are connected if there are a bounded number of faults (as in the classical setting), but that two nodes are connected if there are a bounded number of faults *and the two nodes are connected in the underlying graph post-faults*. That is, the subgraph we build must “behave” identically to the underlying graph with respect to connectivity after bounded faults.

We define and introduce these problems, and provide the first approximation algorithms: a  $(1 + 4/k)$ -approximation for the unweighted relative version of  $k$ -ECSS, a 2-approximation for the weighted relative version of  $k$ -ECSS, and a  $27/4$ -approximation for the special case of Relative Survivable Network Design with only a single demand with a connectivity requirement of 3. To obtain these results, we introduce a number of technical ideas that may of independent interest. First, we give a generalization of Jain’s iterative rounding analysis that works even when the cut-requirement function is not weakly supermodular, but instead satisfies a weaker definition we introduce and term *local* weak supermodularity. Second, we prove a structure theorem and design an approximation algorithm utilizing a new decomposition based on *important separators*, which are structures commonly used in fixed-parameter algorithms that have not commonly been used in approximation algorithms.

**2012 ACM Subject Classification** Theory of computation → Routing and network design problems

**Keywords and phrases** Fault Tolerance, Network Design

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.41

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2206.12245> [13]

**Funding** *Michael Dinitz:* Supported in part by NSF award CCF-1909111.

*Ama Koranteng:* Supported in part by NSF award CCF-1909111.

## 1 Introduction

Fault-tolerance has been a central object of study in approximation algorithms, particularly for network design problems where the graphs that we study represent some physical objects which might fail (communication links, transportation links, etc.). In these settings it is natural to ask for whatever object we build to be fault-tolerant. The precise definition of “fault-tolerance” is different in different settings, but a common formulation is edge



© Michael Dinitz, Ama Koranteng, and Guy Kortsarz;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 41; pp. 41:1–41:19



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

fault-tolerance, which typically takes the form of edge connectivity. Informally, these look like guarantees of the form “if up to  $k$  edges fail, then the nodes I want to be connected are still connected.” For example, consider the following two classical problems.

- The Minimum  $k$ -Edge Connected Subgraph problem ( $k$ -ECSS), where we are given a graph  $G$  and a value  $k$  and are asked to find the  $k$ -edge connected subgraph of  $G$  of minimum size (or weight). In other words, if fewer than  $k$  edges fail, the graph should still be connected.
- The more general Survivable Network Design problem (SND, sometimes referred to as Generalized Steiner Network), where we are given a graph  $G$  and demands  $\{(s_i, t_i, k_i)\}_{i \in [\ell]}$ , and are supposed to find the minimum-weight subgraph  $H$  of  $G$  so that there are at least  $k_i$  edge-disjoint paths between  $s_i$  and  $t_i$  for every  $i \in [\ell]$ . In other words, for every  $i \in [\ell]$ , if fewer than  $k_i$  edges fail then  $s_i$  and  $t_i$  will still be connected in  $H$  even after failures.

Both of these problems have been studied extensively (for a small sample, see [24, 19, 10, 17]), and are paradigmatic examples of network design problems. But there is a different notion of fault-tolerance which is stronger, and in some ways more natural: *relative* fault-tolerance. Relative fault-tolerance makes guarantees that rather than being absolute (“if at most  $k$  edges fail the network still functions”) are relative to an underlying graph or system (“if at most  $k$  edges fail, the subgraph functions just as well as the original graph post-failures”). This allows us to generalize the traditional definition: if the underlying graph has strong enough connectivity properties then the two definitions are the same, but the relative version allows us to make interesting and nontrivial guarantees even when the underlying graph does not have strong connectivity properties.

For example, the definition of Survivable Network Design has an important limitation: if  $G$  itself can only support a small number of edge disjoint  $s_i - t_i$  paths (e.g., 3), then of course we cannot ask for a subgraph with more edge-disjoint paths. There simply would be no feasible solution. But this is somewhat unsatisfactory. For example, while we cannot guarantee that  $s_i$  and  $t_i$  would be connected after *any* set of 5 faults (since those faults may include an  $(s_i, t_i)$  cut of size 3), clearly there could be *some* set of 5 faults which do not in fact disconnect  $s_i$  from  $t_i$  in  $G$ . And if these faults occur, it is natural to want  $s_i$  and  $t_i$  to still be connected in (what remains) of  $H$ . In other words: just because there exists a small cut, why should we give up on being tolerant to a larger number of faults which do not contain that cut?

## 1.1 Our Results and Techniques

In this paper we initiate the study of relative fault-tolerance in network design, by defining relative versions of Survivable Network Design and  $k$ -ECSS.

► **Definition 1.** *In the Relative Survivable Network Design problem (RSND), we are given a graph  $G = (V, E)$  with edge weights  $w : E \rightarrow \mathbb{R}_{\geq 0}$  and demands  $\{(s_i, t_i, k_i)\}_{i \in [\ell]}$ . A feasible solution is a subgraph  $H$  of  $G$  where for all  $i \in [\ell]$  and  $F \subseteq E$  with  $|F| < k_i$ , if there is a path in  $G \setminus F$  from  $s_i$  to  $t_i$  then there is also a path in  $H \setminus F$  from  $s_i$  to  $t_i$ . Our goal is to find the minimum weight feasible solution.*

► **Definition 2.** *The  $k$ -Edge Fault-Tolerant Subgraph problem ( $k$ -EFTS) is the special case of RSND where there is a demand between all pairs and every  $k_i$  is equal to  $k$ . In other words, we are given a graph  $G = (V, E)$  with edge weights  $w : E \rightarrow \mathbb{R}_{\geq 0}$ . A feasible solution is a subgraph  $H$  of  $G$  where for all  $F \subseteq E$  with  $|F| < k$ , any two nodes which have a path between them in  $G \setminus F$  also have a path between them in  $H \setminus F$  (the connected components of  $H \setminus F$  are identical to the connected components of  $G \setminus F$ ). Our goal is to find the minimum weight feasible solution.*

For both of these problems, we say that they are *unweighted* if all edges have the same weight (or equivalently  $w(e) = 1$  for all  $e \in E$ ). Note that if  $s_i$  and  $t_i$  are  $k_i$ -connected in  $G$  for every  $i \in [\ell]$ , then RSND is exactly the same as SND, and if  $G$  is  $k$ -connected then  $k$ -EFTS is exactly the same as  $k$ -ECSS. Hence we have generalized these classical problems.

We note that the fault-tolerance we achieve is really “one less” than the given number (there are strict inequalities in the definitions). This is “off-by-one” from the related relative fault-tolerance literature [9, 5, 6], but makes the connection to SND and  $k$ -ECSS cleaner.

**Difficulties.** Before discussing our results or techniques, we briefly discuss what makes these problems difficult. The non-relative versions are classical and have been studied extensively: why can’t we just re-use the ideas and techniques developed for them? Particularly since there is only a difference in the setting when there are small cuts in the graph, in which case we already know that the edges of those cuts must be included in any feasible solution?

Unfortunately, it turns out that this seemingly minor change has a dramatic impact on the structure of the problem. Most importantly, the *cut requirement function* has dramatically different properties. In  $k$ -ECSS, Menger’s theorem implies that  $H$  is a valid solution if and only if for all  $S \subset V$  with  $S \neq \emptyset$ , there are at least  $k$  edges between  $S$  and  $\bar{S}$ . Hence we can rephrase  $k$ -ECSS as the problem of finding a minimum cost subgraph such that there are at least  $f(S)$  edges across the cut  $(S, \bar{S})$  for all  $S \subset V$  with  $S \neq \emptyset$ , where  $f(S) = k$ . Similarly, we can rephrase SND as the same problem but where  $f(S) = \max_{i \in [\ell]: s_i \in S, t_i \notin S} k_i$  (as was shown in [24]). Thus both problems can be thought of as choosing a minimum cost subgraph subject to satisfying some cut-requirement covering function  $f : 2^V \rightarrow \mathbb{R}$ . So a natural starting point for any approximation algorithm is to write the natural covering LP relaxation which has a covering constraint of  $f(S)$  for every cut  $S$ . And indeed, the covering LP using the cut-requirement function was the starting point for both the primal-dual  $O(\max_{i \in \ell} k_i)$ -approximation for SND of [24] and the seminal 2-approximation for SND using iterative rounding due to Jain [19]. It has also been used for  $k$ -ECSS [17], although (unlike SND) there are also purely combinatorial approximations [10].

Hence the natural starting point for us to study RSND and  $k$ -EFTS would be to formulate them in terms of cut-requirement functions and try the same approaches as were used in SND and  $k$ -ECSS. But this is easier said than done. The functions are a little more complicated, but it is not too hard to construct a cut requirement function that characterizes feasible solutions. However, in order to use the iterative rounding technique of Jain [19] (or any of the weaker techniques which it superceded), the cut requirement function needs to have a structural property known as *weak* (or *skew*) *supermodularity* [19]. This turns out to be crucial, and there are still (to the best of our knowledge) no successful uses of iterative rounding in settings without weak supermodularity. And unfortunately, it turns out that our cut requirement functions *are not* weakly supermodular. So while we can phrase our problems as satisfying a cut requirement function, we cannot actually use iterative rounding, uncrossing, or any other part of the extensive toolkit that has grown around [19].

**Our approaches.** We get around this difficulty in two ways. For  $k$ -EFTS, we define a new property of cut requirement functions which we call *local* weak supermodularity, and prove that our cut requirement function has this property and that it is sufficient for iterative rounding. This is, to the best of our knowledge, *the first* use of iterative rounding without weak supermodularity. For RSND with a single demand, we use an entirely different combinatorial approach based on decomposing the graph into a chain of connected components using *important separators* [22], an important tool from fixed-parameter tractability that, to the best of our knowledge, has not been used before in approximation algorithms.



### 1.1.1 $k$ -Edge Fault-Tolerant Subgraph

We begin in Section 2 with  $k$ -EFTS, where we prove the following two theorems.

► **Theorem 3.** *There is a polynomial-time 2-approximation for the  $k$ -EFTS problem.*

► **Theorem 4.** *There is a polynomial-time  $(1 + 4/k)$ -approximation for the unweighted  $k$ -EFTS problem.*

Both of these theorems are consequences of a structural property we prove about the cut-requirement function for  $k$ -EFTS: while it is not weakly supermodular, it does have a weaker property which we term *local* weak supermodularity. We define this property formally in Section 2.1.2, but at a high level it boils down to proving that while the inequalities required for weak supermodularity do not hold everywhere (as would be required for weak supermodularity), they hold for *particular* sets (i.e., they hold *locally*) which are the sets where the inequalities are actually applied by Jain’s analysis. In other words, we prove that the places in the function where weak supermodularity are violated are precisely the places where we do not care if weak supermodularity holds. After overcoming a few more technical complications (we actually need local weak supermodularity even in the “residual” problem to use iterative rounding), this means that we can apply Jain’s algorithm to prove Theorem 3.

To prove Theorem 4, it was observed for unweighted  $k$ -ECSS by [17] (with later improvements by [16]) that one of the main pieces of Jain’s approach, the fact that the tight constraints can be “uncrossed” to get a laminar family with the same span, implies a  $(1 + 4/k)$ -approximation via a trivial threshold rounding. They pointed out that the fact that the linearly independent tight constraints form a laminar family implies that there are only  $2n$  linearly independent tight constraints, while there are  $m$  variables, and hence at any basic feasible solution the remaining  $m - 2n$  tight constraints defining the point must be the bounding constraints. These bounding constraints being tight means that the associated variables are in  $\{0, 1\}$ , and hence there are only  $2n$  fractional variables in any basic feasible solution. Rounding all of these variables to 1 increases the cost by  $2n$ , but since  $OPT \geq kn/2$  (since the input graph  $G$  must be  $k$ -connected) this results in a  $(1 + 4/k)$ -approximation.

Thanks to our local weak supermodularity characterization the laminar family result is still true even for  $k$ -EFTS, so it is still true that there are at most  $2n$  nonzero variables at any extreme point. But since we are not guaranteed that  $G$  is  $k$ -connected we are not guaranteed that  $OPT \geq kn/2$ , and so this does not imply the desired approximation. Instead, we prove that the number of fractional variables at any basic feasible solution is at most  $2n_h$ , where  $n_h$  is the number of “high-degree” nodes. It is then easy to argue that  $OPT \geq n_h k/2$ , which gives Theorem 4.

### 1.1.2 Relative Survivable Network Design With a Single Demand

For  $k$ -EFTS, we strongly used the property that all pairs have the same demand. This is not true for RSND, which makes the problem vastly more difficult. We still do not know whether there exists a cut requirement function which characterizes the problem and is locally weakly supermodular. In this paper, we study the simplest case where not all demands are the same: when there is a single nonzero demand  $(s, t, k)$ , and  $k$  is either 2 or 3 (the case of  $k = 1$  is simply the shortest-path problem). It turns out to be relatively straightforward to prove a 2-approximation for  $k = 2$  even when there are many demands (see Section 3), but the  $k = 3$  case is surprisingly difficult. We prove the following theorem in Section 4.



► **Theorem 5.** *In any RSND instance with a single demand  $(s, t, 3)$ , there is a polynomial-time  $7 - \frac{1}{4} = \frac{27}{4}$ -approximation.*

To prove this, we start with the observation that if the minimum  $s - t$  cut is at least 3 then this is actually just the traditional SND problem (and in fact, the even simpler problem of finding 3 edge-disjoint paths of minimum weight between  $s$  and  $t$ , which can be solved efficiently via min-cost flow). So the only difficulty is when there are cuts of size 1 or 2. Cuts of size 1 can be dealt with easily (see Section 3), but cuts of size 2 are more difficult. To get rid of them, we construct a “chain” of 2-separators (cuts of size 2 that are also *important separators* [22]). Inside each component of the chain there are no 2-cuts between the incoming separator and the outgoing separator, which allows us to characterize the connectivity requirement of any feasible solution restricted to that component. These connectivity requirements turn out to be quite complex even though we started with only a single demand, as fault sets with different structure can force complicated connectivity requirements in intermediate components. The vast majority of the technical work is proving a structure lemma which characterizes them. With this lemma in hand, though, we can simply approximate the optimal solution in each component.

Interestingly, to the best of our knowledge this is the first use of important separators in approximation algorithms, despite their usefulness in fixed-parameter algorithms [22].

## 1.2 Related Work

The most directly related work is the 2-approximation of Jain for Survivable Network Design [19], which introduced iterative rounding (see [20] for a detailed treatment of iterative rounding in combinatorial optimization). This built off of an earlier line of work on survivable network design beginning over 50 years ago with [23]. Since the success of Jain’s approach for SND, there has been a significant amount of work on vertex-connectivity versions rather than edge-connectivity, which is a significantly more difficult setting. This has culminated in the state of the art approximation of [11]. There is also a long line of work on  $k$ -ECSS, most notably including [10, 17].

While not technically related, the basic problems in this paper are heavily inspired by recent work on relative notions of fault-tolerance in graph spanners and other non-optimization network design settings. A relative definition of fault-tolerance for graph spanners which is very similar to ours (but which takes distances into account due to the spanner setting) was introduced by [9], who gave bounds on the size of  $f$ -fault-tolerant  $t$ -spanners for both edge and vertex notions of fault-tolerance. This spawned a line of work which improved these bounds for both vertex and edge fault-tolerance [14, 4, 7, 15, 5, 6], culminating in [5] for vertex faults and [6] for edge faults. The basic spanner definition also inspired work on relative fault-tolerant versions of related problems, including emulators [3], distance sensitivity oracles for multiple faults [8], and single-source reachability subgraphs [2, 21]. What all of these results shared, though, was that they were not doing optimization: they were looking for existential bounds (and algorithms to achieve them) for these objects. In this paper, by contrast, we take the point of view of optimization and approximation algorithms and compare to the instance-specific optimal solution.

## 2 $k$ -Edge Fault-Tolerant Subgraph

Both Theorems 3 and 4 depend on the same LP relaxation, which is based on a modification of the “obvious” cut-requirement function. So we begin by discussing this relaxation, and then use it to prove the two main theorems.

## 2.1 LP Relaxation

### 2.1.1 Basics

The natural place to start is the LP used by Jain [19], but with a cut requirement function  $f(S) = \min(|\delta_G(S)|, k)$ . Unfortunately, while this results in a valid LP relaxation, it is not weakly supermodular (see Section 2.1.2 for the definition, and Appendix A for a counterexample). So instead we modify this cut requirement function by removing edges which are “forced”. For every subset  $S$  of  $V$ , let  $\delta_G(S)$  be the set of edges with exactly one endpoint in  $S$ . Let  $F = \{e \in E \mid \exists S \text{ where } e \in \delta_G(S) \text{ and } |\delta_G(S)| \leq k\}$ . In other words,  $F$  is the set of all edges that are in some cut of size at most  $k$ . Clearly we can compute  $F$  in polynomial time by simply checking for every edge  $(u, v)$  whether the minimum  $u - v$  cut in  $G$  has size at most  $k$ . For every set  $S \subset V$  with  $S \neq \emptyset$ , we define the cut requirement function  $f_F(S) = \min(k, |\delta_G(S)|) - |\delta_G(S) \cap F|$ . Note that every edge in  $F$  must be in any feasible solution, since if any edge is missing then a fault set consisting of the rest of the cut (at most  $k - 1$  edges) would disconnect the endpoints of the missing edge in the solution but not in  $G$ , giving a contradiction. Then  $f_F(S)$  is essentially the “remaining requirement” after  $F$  has been removed.

Since iterative rounding will add other edges and remove them from the residual problem, we will want to define a similar cut requirement function for supersets: formally, for any  $F' \supseteq F$ , let  $f_{F'}(S) = \min(k, |\delta_G(S)|) - |\delta_G(S) \cap F'|$ . For any  $F' \supseteq F$ , consider the following linear program which we call  $\text{LP}(F')$ , which has a variable  $x_e$  for every edge  $e \in E \setminus F'$ :

$$\begin{array}{ll}
 \min & \sum_{e \in E \setminus F'} w(e)x_e \\
 \text{s.t.} & \sum_{e \in \delta_G(S) \setminus F'} x_e \geq f_{F'}(S) \quad \forall S \subseteq V \\
 & 0 \leq x_e \leq 1 \quad \forall e \in E \setminus F'
 \end{array} \tag{LP(F')}$$

It is not hard to see that this is a valid LP relaxation (when combined with  $F'$ ), but we prove this for completeness.

► **Lemma 6.** *Let  $H$  be a valid  $k$ -EFTS and let  $F' \supseteq F$ . For every edge  $e \in E \setminus F'$ , let  $x_e = 1$  if  $e \in H$ , and let  $x_e = 0$  otherwise. Then  $x$  is a feasible integral solution to  $\text{LP}(F')$ .*

**Proof.** Clearly  $0 \leq x_e \leq 1$  for all  $e \in E \setminus F'$ . Consider some  $S \subseteq V$ . Since  $H$  is a valid  $k$ -EFTS, the number of edges in  $H \cap \delta_G(S)$  is at least  $\min(k, |\delta_G(S)|)$  (or else the edges in  $H \cap \delta_G(S)$  would be a fault set of size less than  $k$  such that the connected components of  $H$  post-faults are different from the connected components of  $G$  post-faults). Hence

$$\begin{aligned}
 \sum_{e \in \delta_G(S) \setminus F'} x_e &= |(H \cap \delta_G(S)) \setminus F'| = |H \cap \delta_G(S)| - |H \cap \delta_G(S) \cap F'| \\
 &\geq |H \cap \delta_G(S)| - |\delta_G(S) \cap F'| \geq \min(k, |\delta_G(S)|) - |\delta_G(S) \cap F'| = f_{F'}(S),
 \end{aligned}$$

as required. ◀

► **Lemma 7.** *Let  $F' \supseteq F$  and let  $x$  be an integral solution to  $\text{LP}(F')$ . Let  $E' = \{e : x_e = 1\}$ . Then  $H = E' \cup F'$  is a valid  $k$ -EFTS.*

**Proof.** Suppose for contradiction that  $H$  is not a valid  $k$ -EFTS. Then there are two nodes  $u, v \in V$  and a minimal set  $A \subseteq E$  with  $|A| < k$  so that  $u, v$  are not connected in  $H \setminus A$  but are connected in  $G \setminus A$ . Let  $S$  be the nodes reachable from  $u$  in  $G \setminus A$ , and so by minimality of  $A$  we know that  $A = H \cap \delta_G(S)$ .

Note that  $|\delta_G(S)| > k$ , or else all edges of  $\delta_G(S)$  would be in  $F$ , implying that  $E \cap \delta_G(S) = H \cap \delta_G(S) = A$  and so  $u$  and  $v$  would not be connected in  $G \setminus A$ . Thus

$$\begin{aligned} \sum_{e \in \delta_G(S) \setminus F'} x_e &= |H \cap \delta_G(S)| - |F' \cap \delta_G(S)| = |A| - |\delta_G(S) \cap F'| \\ &< \min(k, |\delta_G(S)|) - |\delta_G(S) \cap F'| = f_{F'}(S), \end{aligned}$$

which contradicts  $x$  being a feasible solution to  $\text{LP}(F')$ .  $\blacktriangleleft$

These lemmas (together with the fact that every edge in  $F$  must be in any valid solution) imply that if we can solve and round this LP while losing some factor  $\alpha$ , then we can add  $F$  to the rounded solution to get an  $\alpha$ -approximation. Hence we are interested in solving and rounding this LP.

We first argue that we can solve the LP using the Ellipsoid algorithm with a separation oracle. Note that unlike  $k$ -ECSS, here a violated constraint does not just correspond to a cut with LP values less than  $k$ , since our cut-requirement function is more complicated. Indeed, if we compute a global minimum cut (with respect to the LP values) then we may end up with a small cut which is not violated even though there are violated constraints. So we need to argue more carefully that we can find a violated cut when one exists.

► **Lemma 8.** *For every  $F' \supseteq F$ ,  $\text{LP}(F')$  can be solved in polynomial time.*

**Proof.** We give a separation oracle, which when combined with the Ellipsoid algorithm implies the lemma [18]. Consider some vector  $x$  indexed by edges of  $E \setminus F'$ . Suppose that  $x$  is not a feasible LP solution, so we need to find a violated constraint. Obviously if there is some  $x_e \notin [0, 1]$  then we can find this in linear time. So without loss of generality, we may assume that there is some  $S \subseteq V$  such that  $\sum_{e \in \delta_G(S) \setminus F'} x_e < f_{F'}(S)$ . This implies that  $f_{F'}(S) > 0$  and that there is some edge  $e^* \in \delta_G(S) \setminus F'$  with  $x_{e^*} < 1$  (since otherwise the LP would not be satisfiable, contradicting Lemma 6 and the fact that  $G$  itself is a valid  $k$ -EFTS). Let  $e^* = \{u, v\}$ . Since  $e^* \notin F'$ , and  $F \subseteq F'$ , we know that  $e^*$  cannot be part of any cuts in  $G$  of size at most  $k$ , and thus the minimum  $u - v$  cut in  $G$  has more than  $k$  edges.

On the other hand, if we extend  $x$  to  $F'$  by setting  $x_e = 1$  for all  $e \in F'$ , then since  $S$  is a violated constraint we have that

$$\begin{aligned} \sum_{e \in \delta_G(S)} x_e &= \sum_{e \in \delta_G(S) \setminus F'} x_e + |F' \cap \delta_G(S)| < f_{F'}(S) + |F \cap \delta_G(S)| \\ &= \min(k, |\delta_G(S)|) - |\delta_G(S) \cap F'| + |\delta_G(S) \cap F'| \\ &= k. \end{aligned}$$

Thus if we interpret  $x$  as edge weights (with  $x_e = 1$  for all  $e \in F$ ), if we compute the minimum  $s - t$  cut we will find a cut  $S'$  with more than  $k$  edges (since all  $u - v$  cuts have more than  $k$  edges) with total edge weight strictly less than  $k$ . Let  $S'$  be this cut. Thus  $\sum_{e \in \delta_G(S') \setminus F'} x_e < k - |\delta_G(S') \setminus F'| = f_{F'}(S')$ , so  $S'$  is also a violated constraint.

Hence for our separation oracle we simply compute a minimum  $s - t$  cut using  $x$  as edge weights for all  $s, t \in V$ , and if any cut we finds corresponds to a violated constraint then we return it. By the above discussion, if there is some violated constraint then this procedure will find some violated constraint. Thus this is a valid separation oracle.  $\blacktriangleleft$

After solving this LP, we apply an obvious transformation used also in [19]: we delete every edge  $e$  with  $x_e = 0$ . This allows us to assume without loss of generality that every edge has LP value  $x_e > 0$  in our LP solution.

### 2.1.2 Local Weak Supermodularity

As discussed in Section 1.1.1, it would be nice if this LP were *weakly supermodular*, as this would immediately let us apply Jain’s iterative rounding algorithm to obtain a 2-approximation. Recall the definition of weak supermodularity from [19].

► **Definition 9.** *Let  $f : 2^V \rightarrow \mathbb{Z}$ . Then  $f$  is weakly supermodular if for every  $A, B \subseteq V$ , either  $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$ , or  $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$ .*

Unfortunately, our cut requirement function is not weakly supermodular; see Appendix A for a counterexample. But we can make a simple observation that, to the best of our knowledge, has not previously been noticed or utilized in iterative rounding: Jain’s iterative rounding algorithm does not actually need the weak supermodularity conditions to hold for *all* pairs of sets  $A, B$ . It only needs weak supermodularity to “uncross” the tight sets of an LP solution into a laminar family of tight sets with the same span. Recall that a set is tight in a given LP solution if its corresponding cut constraint is tight, i.e., is satisfied with equality. Moreover, note that in our setting, depending on our choice of  $F'$  some cuts might be entirely included in  $F'$ . These cuts would not have any edges remaining, resulting in an “empty” constraint in  $\text{LP}(F')$ . Such a constraint cannot be tight by definition, and also is not linearly independent with any other set of constraints.

Hence in order to use Jain’s iterative rounding, we simply need our cut-requirement function  $f_{F'}$  to satisfy the weak supermodularity requirements for  $A, B$  where there is actually a nontrivial constraint for  $A, B$  and where  $F' \supseteq F$  (here  $F'$  will consist of  $F$  together with edges that Jain’s iterative rounding algorithm has already set to 1). We formalize this as follows. Given  $F' \supseteq F$ , we say that  $S$  is an *empty cut* if  $\delta_G(S) \cap F' = \delta_G(S)$ , and otherwise it is *nonempty*.

► **Definition 10.** *Given a graph  $G = (V, E)$ , a set  $F' \subseteq E$ , and a function  $g : 2^V \rightarrow \mathbb{Z}$ , we say that  $g$  is locally weakly supermodular with respect to  $F'$  if for every  $A, B \subseteq V$  with both  $A$  and  $B$  nonempty cuts, at least one of the following conditions holds:*

- $g(A) + g(B) \leq g(A \setminus B) + g(B \setminus A)$ , or
- $g(A) + g(B) \leq g(A \cap B) + g(A \cup B)$ .

We will now prove that for any  $F' \supseteq F$ , the function  $f_{F'}$  is locally weakly supermodular with respect to any  $F'$ . This is the key technical idea enabling Theorems 3 and 4.

We say that  $S$  is *large* if  $|\delta_G(S)| > k$ , and otherwise  $S$  is *small*. Note that since  $F' \supseteq F$ , any small cut is also an empty cut. We first prove a useful lemma.

► **Lemma 11.** *Let  $F' \supseteq F$ . If  $A$  and  $B$  are nonempty cuts for  $f_{F'}$ , then either  $A \setminus B$  and  $B \setminus A$  are nonempty cuts, or  $A \cap B$  and  $A \cup B$  are nonempty cuts.*

**Proof.** Let

$$\begin{aligned} S_1 &= \delta_G(A \setminus B, V \setminus (A \cup B)), & S_2 &= \delta_G(A \setminus B, B \setminus A), & S_3 &= \delta_G(A \setminus B, A \cap B), \\ S_4 &= \delta_G(B \setminus A, V \setminus (A \cup B)), & S_5 &= \delta_G(B \setminus A, A \cap B), & S_6 &= \delta_G(A \cap B, V \setminus (A \cup B)). \end{aligned}$$

Suppose that  $A \setminus B$  and  $A \cap B$  are both empty cuts. Each edge in  $\delta_G(A)$  is in  $S_1, S_2, S_5$ , or  $S_6$ . Additionally,  $S_1$  and  $S_2$  are subsets of  $\delta_G(A \setminus B)$ , while  $S_5$  and  $S_6$  are subsets of  $\delta_G(A \cap B)$ . This means that every edge in  $\delta_G(A)$  is in an empty cut, and so all edges in  $\delta_G(A)$  are in  $F'$ . Thus  $A$  is an empty cut, contradicting the assumption of the lemma. Thus at least one of  $A \setminus B$  and  $A \cap B$  is nonempty. If we instead assume that  $B \setminus A$  and  $A \cap B$  are empty cuts, then we can use a similar argument to prove that  $B$  is an empty cut. This proves that at least one of  $B \setminus A$  and  $A \cap B$  are nonempty. Hence if  $A \cap B$  is empty, then both  $A \setminus B$  and  $B \setminus A$  are nonempty, proving the lemma.

Now suppose that  $A \setminus B$  and  $A \cup B$  are both empty cuts. Each edge in  $\delta_G(B)$  is in  $S_2$ ,  $S_3$ ,  $S_4$ , or  $S_6$ . Additionally,  $S_2$  and  $S_3$  are subsets of  $\delta_G(A \setminus B)$ , while  $S_4$  and  $S_6$  are subsets of  $\delta_G(A \cup B)$ . This means that every edge in  $\delta_G(B)$  is in an empty cut, and so all edges in  $\delta_G(B)$  are in  $F'$ . Thus  $B$  is an empty cut, contradicting the assumption of the lemma. Thus at least one of  $A \setminus B$  and  $A \cup B$  is nonempty. If we instead assume that  $B \setminus A$  and  $A \cup B$  are empty cuts, then we can use a similar argument to prove that  $A$  is empty, and hence at least one of  $B \setminus A$  and  $A \cup B$  is nonempty. Hence if  $A \cup B$  is empty, then both  $A \setminus B$  and  $B \setminus A$  are nonempty, proving the lemma.

Thus either both  $A \setminus B$  and  $B \setminus A$  are nonempty, or both  $A \cap B$  and  $A \cup B$  are nonempty, proving the lemma.  $\blacktriangleleft$

We can now prove the main technical result:  $f_{F'}$  is locally weakly supermodular.

► **Theorem 12** (Local Weak Supermodularity). *For any  $F' \supseteq F$ , the cut requirement function  $f_{F'}$  is locally weakly supermodular with respect to  $F'$ .*

**Proof.** Let  $F' \supseteq F$ , and suppose  $A$  and  $B$  are nonempty cuts. Let

$$\begin{aligned} S_1 &= \delta_G(A \setminus B, V \setminus (A \cup B)), & S_2 &= \delta_G(A \setminus B, B \setminus A), & S_3 &= \delta_G(A \setminus B, A \cap B), \\ S_4 &= \delta_G(B \setminus A, V \setminus (A \cup B)), & S_5 &= \delta_G(B \setminus A, A \cap B), & S_6 &= \delta_G(A \cap B, V \setminus (A \cup B)). \end{aligned}$$

We also let  $s_i = |S_i \cap F'|$  for  $i \in [6]$ .

$A$  and  $B$  are nonempty cuts, so  $A$  and  $B$  must be large cuts and  $\min(k, |\delta_G(A)|) = \min(k, |\delta_G(B)|) = k$ . Each edge in  $\delta_G(A)$  is in exactly one of  $S_1$ ,  $S_2$ ,  $S_5$ , and  $S_6$ , and each edge in  $\delta_G(B)$  is in exactly one of  $S_2$ ,  $S_3$ ,  $S_4$ , and  $S_6$ , so we have that  $|\delta_G(A) \cap F'| = s_1 + s_2 + s_5 + s_6$  and  $|\delta_G(B) \cap F'| = s_2 + s_3 + s_4 + s_6$ . We therefore have the following:

$$\begin{aligned} f_{F'}(A) &= \min(k, |\delta_G(A)|) - |\delta_G(A) \cap F'| = k - s_1 - s_2 - s_5 - s_6 \\ f_{F'}(B) &= \min(k, |\delta_G(B)|) - |\delta_G(B) \cap F'| = k - s_2 - s_3 - s_4 - s_6 \\ \implies f_{F'}(A) + f_{F'}(B) &= 2k - s_1 - 2s_2 - s_3 - s_4 - s_5 - 2s_6. \end{aligned} \tag{1}$$

$A$  and  $B$  are nonempty so by Lemma 11, either  $A \setminus B$  and  $B \setminus A$  are nonempty cuts, or  $A \cap B$  and  $A \cup B$  are nonempty cuts. Suppose first that  $A \setminus B$  and  $B \setminus A$  are nonempty cuts, which implies that  $\min(k, |\delta_G(A \setminus B)|) = \min(k, |\delta_G(B \setminus A)|) = k$ . Each edge in  $\delta_G(A \setminus B)$  is in exactly one of  $S_1$ ,  $S_2$ , and  $S_3$ , and each edge in  $\delta_G(B \setminus A)$  is in exactly one of  $S_2$ ,  $S_4$ , and  $S_5$ , so we have that  $|\delta_G(A \setminus B) \cap F'| = s_1 + s_2 + s_3$  and  $|\delta_G(B \setminus A) \cap F'| = s_2 + s_4 + s_5$ . Putting this all together, we get the following for  $f_{F'}(A \setminus B)$  and  $f_{F'}(B \setminus A)$ :

$$\begin{aligned} f_{F'}(A \setminus B) &= \min(k, |\delta_G(A \setminus B)|) - |\delta_G(A \setminus B) \cap F'| = k - s_1 - s_2 - s_3 \\ f_{F'}(B \setminus A) &= \min(k, |\delta_G(B \setminus A)|) - |\delta_G(B \setminus A) \cap F'| = k - s_2 - s_4 - s_5 \\ \implies f_{F'}(A \setminus B) + f_{F'}(B \setminus A) &= 2k - s_1 - 2s_2 - s_3 - s_4 - s_5. \end{aligned}$$

This and (1) imply that  $f_{F'}(A) + f_{F'}(B) \leq f_{F'}(A \setminus B) + f_{F'}(B \setminus A)$  if  $A \setminus B$  and  $B \setminus A$  are nonempty cuts.

Now suppose that  $A \cap B$  and  $A \cup B$  are nonempty cuts, and so  $\min(k, |\delta_G(A \cap B)|) = \min(k, |\delta_G(A \cup B)|) = k$ . Each edge in  $\delta_G(A \cap B)$  is in exactly one of  $S_3$ ,  $S_5$ , and  $S_6$ , and each edge in  $\delta_G(A \cup B)$  is in exactly one of  $S_1$ ,  $S_4$ , and  $S_6$ , so we have that  $|\delta_G(A \cap B) \cap F'| = s_3 + s_5 + s_6$  and  $|\delta_G(A \cup B) \cap F'| = s_1 + s_4 + s_6$ . Putting this all together, we get the following for  $f_{F'}(A \cap B)$  and  $f_{F'}(A \cup B)$ :

$$\begin{aligned} f_{F'}(A \cap B) &= \min(k, |\delta_G(A \cap B)|) - |\delta_G(A \cap B) \cap F'| = k - s_3 - s_5 - s_6 \\ f_{F'}(A \cup B) &= \min(k, |\delta_G(A \cup B)|) - |\delta_G(A \cup B) \cap F'| = k - s_1 - s_4 - s_6 \\ \implies f_{F'}(A \cap B) + f_{F'}(A \cup B) &= 2k - s_1 - s_3 - s_4 - s_5 - 2s_6. \end{aligned}$$

This and (1) imply that  $f_{F'}(A) + f_{F'}(B) \leq f_{F'}(A \cap B) + f_{F'}(A \cup B)$  if  $A \cap B$  and  $A \cup B$  are nonempty cuts.  $\blacktriangleleft$

## 2.2 Unweighted $k$ -EFTS

To prove Theorem 4 we need to look inside [19]. The following two lemmas from [19] are the main “uncrossing” lemmas which depend on weak supermodularity, and in which we can use local weak supermodularity instead without change. As in [19], for each  $S \subseteq V$  we use  $\mathcal{A}_G(S)$  to denote the row of the constraint matrix corresponding to  $S$ . In other words  $\mathcal{A}_G(S)$  is a vector indexed by elements of  $E \setminus F$  which has a 1 in the entry for  $e$  if  $e \in \delta_G(S) \setminus F$ , and otherwise has a 0 in that entry.

► **Lemma 13** (Lemma 4.1 of [19]). *If two sets  $A$  and  $B$  are tight then at least one of the following must hold*

1.  $A \setminus B$  and  $B \setminus A$  are also tight, and  $\mathcal{A}_G(A) + \mathcal{A}_G(B) = \mathcal{A}_G(A \setminus B) + \mathcal{A}_G(B \setminus A)$
2.  $A \cap B$  and  $A \cup B$  are also tight, and  $\mathcal{A}_G(A) + \mathcal{A}_G(B) = \mathcal{A}_G(A \cap B) + \mathcal{A}_G(A \cup B)$

Let  $\mathcal{T}$  denote the family of all tight sets. For any family  $\mathcal{F}$  of tight sets, let  $\text{Span}(\mathcal{F})$  denote the vector space spanned by  $\{\mathcal{A}_G(S) : S \in \mathcal{F}\}$ .

► **Lemma 14** (Lemma 4.2 of [19]). *For any maximal laminar family  $\mathcal{L}$  of tight sets,  $\text{Span}(\mathcal{L}) = \text{Span}(\mathcal{T})$ .*

Recall that  $n_h$  is the number of high-degree nodes, i.e., nodes of degree at least  $k$  in  $G$ . Then we have the following lemma, which is a modification of Lemma 4.3 of [19] where we give a stronger bound on the number of sets.

► **Lemma 15.** *The dimension of  $\text{Span}(\mathcal{T})$  is at most  $2n_h - 1$ .*

**Proof.** Let  $\mathcal{L}$  be a maximal laminar family of tight sets. Lemma 14 implies that  $\text{Span}(\mathcal{L}) = \text{Span}(\mathcal{T})$ , so it suffices to upper bound the number of sets in  $\mathcal{L}$ . And since we care about the span, if there are two sets  $S, S'$  with  $\mathcal{A}_G(S) = \mathcal{A}_G(S')$  then we can remove one of them from  $\mathcal{L}$  arbitrarily, so no two sets in  $\mathcal{L}$  have identical rows in the constraint matrix.

Any set that consists of exclusively low degree nodes cannot be tight, since the set has no corresponding row in the constraint matrix. Thus, all sets in  $\mathcal{L}$  must contain at least one high degree node, and hence all minimal sets in  $\mathcal{L}$  have at least one high degree node.

Let  $S \in \mathcal{L}$ , and let  $S' \supset S$  so that every node in  $S' \setminus S$  is a low-degree node. Then every edge in  $(\delta_G(S) \setminus \delta_G(S')) \cup (\delta_G(S') \setminus \delta_G(S))$  must be incident on at least one low-degree node and hence is in  $F$ . Thus  $\mathcal{A}_G(S) = \mathcal{A}_G(S')$ , and hence  $S'$  is not in  $\mathcal{L}$ . Therefore, any superset  $S'$  in the laminar family of some other set  $S$  in the laminar family must have at least one more high degree node than  $S$ .

Since any minimal set in  $\mathcal{L}$  has at least one high degree node, and every set in  $\mathcal{L}$  contains at least one more high degree node than any set in  $\mathcal{L}$  that it contains, if we restrict each set in  $\mathcal{L}$  to the high-degree nodes then we have a laminar family on the high-degree nodes. Thus  $|\mathcal{L}| \leq 2n_h - 1$ . ◀

We can now prove Theorem 4.

**Proof of Theorem 4.** We first solve  $\text{LP}(F)$  using Lemma 8 to get some basic feasible solution  $x$ . Since there are  $|E \setminus F|$  variables, this point is defined by  $|E \setminus F|$  linearly independent tight constraints. Lemma 15 implies that at most  $2n_h - 1$  of these are from tight sets, and hence all of the other tight constraints must be of the form  $x_e = 0$  or  $x_e = 1$  for some edge  $e \in E \setminus F$ . Thus at most  $2n_h - 1$  edges are assigned a fractional value in  $x$ . Hence if we include all such edges in our solution  $H$ , together with all edges with  $x_e = 1$  and all edges in

$F$ , we have a solution which is feasible (by Lemma 7). Note that any high-degree node must have degree at least  $k$  in any feasible solution, and thus  $OPT \geq \frac{k}{2}n_h$ . Hence our solution  $H$  has size at most

$$|H| \leq \sum_{e \in E \setminus F} x_e + |F| + 2n_h \leq OPT + 2n_h \leq OPT + \frac{4}{k}OPT = \left(1 + \frac{4}{k}\right)OPT. \quad \blacktriangleleft$$

### 2.3 Weighted $k$ -EFTS

Jain's approximation algorithm solves the initial LP, rounds up and removes any edges with  $x_e \geq 1/2$  which results in a residual problem, and repeats. This is obviously a 2-approximation (see [19] for details), but requires proving that there is always at least one edge with  $x_e \geq 1/2$  so we can make progress (even in the residual problems). This is accomplished by proving Lemmas 13 and 14 to show that the tight constraints can be “uncrossed” into a laminar family. This requires weak supermodularity, but as discussed, since in our LP every tight constraint must be a nonempty constraint, it is sufficient to replace this with local weak supermodularity. Jain then uses a complex counting argument based on this laminar family of tight constraints to prove that some edge  $e$  must have  $x_e \geq 1/2$ . Importantly, nothing in this counting argument depends on the cut requirement having any particular structure (e.g., weak supermodularity); it depends only on the fact that the family of tight constraints can be uncrossed to be laminar.

Since local weak supermodularity is sufficient to uncross the tight constraints into a laminar family, we can simply apply Jain's counting argument on this family for  $LP(F')$  to obtain the following lemma (as in Theorem 3.1 of [19]).

► **Lemma 16.** *For all  $F' \supseteq F$ , in any basic feasible solution  $x$  of  $LP(F')$  there is at least one  $e \in E \setminus F'$  with  $x_e \geq 1/2$ .*

Hence we have the following iterative rounding algorithm for weighted  $k$ -EFTS:

- Let  $F' = F$
- While  $F'$  is not a feasible solution:
  - Let  $x$  be a basic feasible solution for  $LP(F')$  (obtained in polynomial time using Lemma 8)
  - Let  $E_{1/2} = \{e \in E \setminus F' : x_e \geq 1/2\}$ , which must be nonempty by Lemma 16
  - Add  $E_{1/2}$  to  $F'$

This clearly returns a feasible solution, and the analysis of [19] (particularly Theorem 3.2) implies that this is a 2-approximation, which implies Theorem 3.

## 3 2-Connectivity and $k = 2$

We will now move on from  $k$ -EFTS to the more general RSND problem. It turns out to be relatively straightforward to handle cuts of size 1: removing such cuts gives a tree of 2-connected components, and we can essentially run an algorithm independently inside each component. This gives the following theorem, the proof of which can be found in the full version [13].

► **Theorem 17.** *If there exists an  $\alpha$ -approximation algorithm for RSND on 2-edge connected graphs, then there is an  $\alpha$ -approximation algorithm for RSND on general graphs.*



Extending this slightly gives the following theorem (proof in the full version [13]), where 2-RSND denotes the special case of the RSND problem where  $k_i \leq 2$  for all  $i$ .

► **Theorem 18.** *There is a 2-approximation algorithm for 2-RSND.*

#### 4 RSND with a Single Demand: $k = 3$

In this section we prove Theorem 5. In the Single Demand RSND problem, we are given a graph  $G = (V, E)$  (possibly with edge weights  $w : E \rightarrow \mathbb{R}^+$ ) and a  $k$ -relative fault tolerance demand for a single vertex pair  $(s, t)$ . In other words, the set of connectivity demands is just  $\{(s, t, k)\}$ . We give a  $7 - \frac{1}{4} = \frac{27}{4}$ -approximation algorithm for the  $k = 3$  Single Demand RSND problem. The main idea is to partition the input graph using important separators, prove a structure lemma which characterizes the required connectivity guarantees within each component of the partition, and then achieve these guarantees using a variety of subroutines: a min-cost flow algorithm, a 2-RSND approximation algorithm (Theorem 18), and a Steiner Forest approximation algorithm [1].

##### 4.1 Decomposition

By Theorem 17, an  $\alpha$ -approximation algorithm for RSND on 2-connected graphs implies an  $\alpha$ -approximation algorithm for RSND on general graphs. Hence going forward, we will assume the input graph  $G$  is 2-connected. In this section we define important separators and describe how to construct what we call the  $s - t$  2-chain of  $G$ .

► **Definition 19.** *Let  $X$  and  $Y$  be vertex sets of a graph  $G$ . An  $(X, Y)$ -separator of  $G$  is a set of edges  $S$  such that there is no path between any vertex  $x \in X$  and any vertex  $y \in Y$  in  $G \setminus S$ . An  $(X, Y)$ -separator  $S$  is minimal if no subset  $S' \subset S$  is also an  $(X, Y)$ -separator. If  $X = \{x\}$  and  $Y = \{y\}$ , we say that  $S$  is an  $(x, y)$ -separator.*

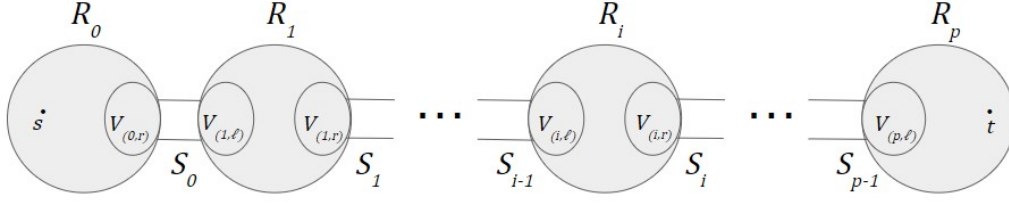
The next definition, which is a slight modification of the definition due to [22], is a formalization of a notion of a “closest” separator.

► **Definition 20.** *Let  $S$  be an  $(X, Y)$ -separator of graph  $G$ , and let  $R$  be the vertices reachable from  $X$  in  $G \setminus S$ . Then  $S$  is an important  $(X, Y)$ -separator if  $S$  is minimal and there is no  $(X, Y)$ -separator  $S'$  such that  $|S'| \leq |S|$  and  $R' \subset R$ , where  $R'$  is the set of vertices reachable from  $X$  in  $G \setminus S'$ .*

This definition corresponds to a “closest” separator, while the original definition of [22] correspond to a “farthest” separator. Important separators have been studied extensively due to their usefulness in fixed-parameter tractable algorithms, and so much is known about them. For our purposes, we will only need the following lemma, which follows directly from Theorem 2 of [22].

► **Lemma 21.** *Let  $X, Y \subseteq V$  be two sets of vertices in graph  $G = (V, E)$ , and let  $d \geq 0$ . An important  $(X, Y)$ -separator of size  $d$  can be found in time  $4^d \cdot n^{O(1)}$  (if one exists), where  $n = |V|$ .*

By Lemma 21, we can find an important  $(X, Y)$ -separator of size 2 in polynomial time. We now describe how to use this to construct what we call the  $s - t$  2-chain of  $G$ . First, if there are no important  $(s, t)$ -separators of size 2 in  $G$ , then every  $(s, t)$ -separator has size at least 3. Hence we can just use the 2-approximation for Survivable Network Design [19] with demand  $(s, t, 3)$  to solve the problem (or can exactly solve it by finding the cheapest three pairwise disjoint  $s - t$  paths in polynomial time using a min-cost flow algorithm).



■ **Figure 1** The  $s - t$  2-chain of  $G$ .

If such an important separator exists, then we first find an important  $(s, t)$ -separator  $S_0$  of size 2 in  $G$ , and let  $R_0$  be the set of vertices reachable from  $s$  in  $G \setminus S_0$ . We let  $V_{(0,r)}$  be the nodes in  $R_0$  incident on  $S_0$ , and let  $V_{(1,l)}$  be the nodes in  $V \setminus R_0$  incident on  $S_0$ . We then proceed inductively. Given  $V_{(i,l)}$ , if there is no important  $(V_{(i,l)}, t)$  separator of size 2 in  $G \setminus (\cup_{j=0}^{i-1} R_j)$  then the chain is finished. Otherwise, let  $S_i$  be such a separator, let  $R_i$  be the nodes reachable from  $V_{(i,l)}$  in  $(G \setminus (\cup_{j=0}^{i-1} R_j)) \setminus S_i$ , let  $V_{(i,r)}$  be the nodes in  $R_i$  incident on  $S_i$ , and let  $V_{(i+1,l)}$  be the nodes in  $V \setminus (\cup_{j=0}^i R_j)$  incident on  $S_i$ .

After this process completes we have our  $s - t$  2-chain, consisting of components  $R_0, \dots, R_p$  along with important separators  $S_0, \dots, S_{p-1}$  between the components. See Figure 1.

We can now use this chain construction to give a structure lemma which characterizes feasible solutions. Informally, the lemma states that a subgraph  $H$  of  $G$  is a feasible solution if and only if in the  $s - t$  2-chain of  $G$ , all edges between components are in  $H$ , and in every component  $R_i$  certain connectivity requirements between  $V_{(i,l)}$  and  $V_{(i,r)}$  are met.

Let  $G = (V, E)$  be a graph, and let  $H$  be a subgraph of  $G$ . Going forward, we will say that in  $H$ , a vertex set  $A \subset V$  has a path to (or is reachable from) another vertex set  $B \subset V$  if there is a path from a vertex  $a \in A$  to a vertex  $b \in B$  in  $H$ . Additionally, let  $X$  and  $Y$  be vertex sets. We also say that  $H$  satisfies the RSND demand  $(X, Y, k)$  on input graph  $G$  if the following is true: for every  $F \subseteq E$  with  $|F| < k$ , if there is a path from at least one vertex in  $X$  to at least one vertex in  $Y$  in  $G \setminus F$  then there is a path from at least one vertex in  $X$  to at least one vertex in  $Y$  in  $H \setminus F$ . The demand  $(X, Y, k)$  on input  $G$  is equivalent to contracting all nodes in  $X$  to create super node  $v_X$ , contracting all nodes in  $Y$  to create super node  $v_Y$ , and including demand  $(v_X, v_Y, k)$ . We will also let  $G[R_i]$  and  $H[R_i]$  be the subgraphs of  $G$  and  $H$ , respectively, induced by the component  $R_i$ .

► **Lemma 22 (Structure Lemma).** *Let  $G$  be the input graph, and let  $H$  be a subgraph of  $G$ . Additionally, let  $R_0, \dots, R_p$  denote the components in the  $s - t$  2-chain of  $G$ , and let  $S_0, \dots, S_{p-1}$  denote the edge sets between components in the chain, as defined previously. Let  $G_i = G[R_i]$ , and  $H_i = H[R_i]$ . Then  $H$  is a feasible solution to the  $k = 3$  Single Demand RSND problem if and only if all edges in  $S_0, \dots, S_{p-1}$  are included in  $H$ , and  $H_i$  has the following properties for every  $i$ :*

1. *There are at least 3 edge-disjoint paths from  $V_{(i,l)}$  to  $V_{(i,r)}$ .*
2.  *$H_i$  is a feasible solution to RSND on input graph  $G_i$  with demands*

$$\{(V_{(i,l)}, v_r, 2) : v_r \in V_{(i,r)}\} \cup \{(V_{(i,r)}, v_l, 2) : v_l \in V_{(i,l)}\}.$$

3.  *$H_i$  is a feasible solution to RSND on input graph  $G_i$  with demands*

$$\{(u, v, 1) : (u, v) \in V_{(i,l)} \times V_{(i,r)}\}.$$

The proof of this structure lemma is a highly technical case analysis, which due to space constraints can be found in the full version [13]. At a very high level, though, our proof is as follows. For the “only if” direction, we first assume that we are given some feasible

solution  $H$ . Then for each of the properties in Lemma 22, we assume it is false and derive a contradiction by finding a fault set  $F \subseteq E$  with  $|F| \leq 2$  where there is a path from  $s$  to  $t$  in  $G \setminus F$ , but not in  $H \setminus F$ . The exact construction of such an  $F$  depends on which of the properties of Lemma 22 we are analyzing.

For the more complicated “if” direction, we assume that  $H$  satisfies the conditions of Lemma 22 and consider a fault set  $F \subseteq E$  with  $|F| \leq 2$  where  $s$  and  $t$  are connected in  $G \setminus F$ . We want to show that  $s$  and  $t$  are connected in  $H \setminus F$ . We analyze two subchains of the  $s - t$  2-chain of  $G$ : the minimal prefix of the chain which contains at least 1 fault, and the minimal prefix of the chain which contains both faults. We first show that the set of vertices reachable from  $s$  at the end of the first subchain is the same in  $G \setminus F$  and in  $H \setminus F$ . We then use this to show that there is at least one reachable vertex at the end of the second subchain in  $H \setminus F$ , even though (unlike the first subchain) the set of reachable vertices at the end of the second subchain may be smaller in  $H \setminus F$  than in  $G \setminus F$ . From there we show that there is a path to  $t$  in  $H \setminus F$  from this one reachable vertex. There are a large number of cases depending on the structure of  $F$  (whether it intersects some of the separators in the chain, whether both faults are in the same component, etc.), and we have to use different properties of Lemma 22 in different cases, making this proof technically involved.

## 4.2 Algorithm and Analysis

We can now use Lemma 22 to give a  $7 - \frac{1}{4} = \frac{27}{4}$ -approximation algorithm for the  $k = 3$  setting of Single Demand RSND on 2-connected graphs which, by Theorem 17, gives a  $\frac{27}{4}$ -approximation algorithm for the  $k = 3$  Single Demand RSND problem on general graphs.

Our algorithm uses a variety of subroutines, including an algorithm for min-cost flow, the 2-RSND approximation algorithm of Theorem 18, and a Steiner Forest approximation algorithm. For reference, we state the latter of these.

► **Lemma 23** ([1]). *There is a  $(2 - \frac{1}{k})$ -approximation algorithm for the Steiner Forest problem, where  $k$  is the number of terminal pairs in the input.*

We can now give our algorithm. Given a graph  $G = (V, E)$  with edge weights  $w : E \rightarrow \mathbb{R}_{\geq 0}$  and demand  $\{(s, t, 3)\}$ , we first create the  $s - t$  2-chain of  $G$  in polynomial time, as described in Section 4.1. After building the chain, within each component we run a set of algorithms to satisfy the demands characterized by Lemma 22: a combination of min-cost flow, 2-RSND, and Steiner Forest algorithms. We include the outputs of these algorithms in our solution  $H$ , together with all edges in the separators  $S = S_1 \cup S_2 \cup \dots \cup S_{p-1}$ .

We first create an instance of min-cost flow on  $G[R_i]$  (in polynomial time). Contract the vertices in  $V_{(i,\ell)}$  and contract the vertices in  $V_{(i,r)}$  to create super nodes  $v_\ell$  and  $v_r$ , respectively. Let  $v_\ell$  be the source node and  $v_r$  be the sink node. For each edge  $e \in E(R_i)$  set the capacity of  $e$  to 1 and set the cost of  $e$  to  $w(e)$ . Require a minimum flow of 3, and run a polynomial-time min-cost flow algorithm on this instance [12]. Since all capacities are integers the algorithm will return an integral flow, so we add to  $H$  all edges with non-zero flow.

We then create our first instance of 2-RSND on  $G[R_i]$ . Contract the vertices in  $V_{(i,\ell)}$  to create super node  $v_\ell$ , and set demands  $\{(v_\ell, u, 2) : u \in V_{(i,r)}\}$ . For our second instance of 2-RSND on  $R_i$ , contract  $V_{(i,r)}$  to create super node  $v_r$ , and set demands  $\{(u, v_r, 2) : u \in V_{(i,\ell)}\}$ . We run the 2-RSND algorithm (Theorem 18) on each of these instances and include all selected edges in  $H$ .

Finally, we create an instance of the Steiner Forest problem on  $G[R_i]$ . For each vertex pair  $(v_\ell, v_r) \in V_{(i,\ell)} \times V_{(i,r)}$ , we check in polynomial time if  $v_\ell$  and  $v_r$  are connected in  $G[R_i]$ . If they are connected, then we include  $(v_\ell, v_r)$  as a terminal pair in the Steiner Forest instance. Additionally, for  $e \in E(R_i)$ , we set the cost of  $e$  to  $w(e)$ . We run the Steiner Forest approximation algorithm (Lemma 23) on this instance, and add all selected edges to  $H$ .

The following lemma is essentially directly from Lemma 22 (the structure lemma) and the description of our algorithm.

► **Lemma 24.**  *$H$  is a feasible solution.*

**Proof.** For each  $i$ , let  $H_i$  denote the subgraph of  $H$  induced by  $R_i$  and let  $G_i$  denote the subgraph of  $G$  induced by  $R_i$ . We will show that  $H$  satisfies the conditions of Lemma 22, and hence is feasible. By construction,  $H$  contains all edges  $S$  in the important separators.

To show property 1 of Lemma 22, recall that in each  $H_i$  we included the edges selected via a min-cost flow algorithm from  $V_{(i,\ell)}$  to  $V_{(i,r)}$  with flow 3. Since there are at least three edge-disjoint paths from  $V_{(i,\ell)}$  to  $V_{(i,r)}$  in  $G_i$  (by Lemma 22 since  $G$  itself is feasible), this will return three edge-disjoint paths from  $V_{(i,\ell)}$  to  $V_{(i,r)}$ . Hence  $H$  satisfies the first property.

Property 2 of Lemma 22 is direct from the algorithm, since  $H_i$  includes the output of the 2-RSND algorithm from Theorem 18 when run on demands  $\{(V_{(i,\ell)}, v_r, 2) : v_r \in V_{(i,r)}\} \cup \{(V_{(i,r)}, v_\ell, 2) : v_\ell \in V_{(i,\ell)}\}$ . Similarly, within each component  $H_i$  in the  $s - t$  2-chain, the edges selected by the Steiner Forest algorithm form a path from vertex  $v_\ell \in V_{(i,\ell)}$  to vertex  $v_r \in V_{(i,r)}$  if  $v_\ell$  and  $v_r$  are connected in  $G$ . This satisfies Property 3 in Lemma 22. ◀

Let  $H^*$  denote the optimal solution, and for any set of edges  $A \subseteq E$ , let  $w(A) = \sum_{e \in A} w(e)$ . The next lemma follows from combining the approximation ratios of each of the subroutines used in our algorithm.

► **Lemma 25.**  $w(H) \leq \frac{27}{4} \cdot w(H^*)$

**Proof.** Let  $H_i = H[R_i]$  be the subgraph of  $H$  induced by  $R_i$ , and let  $H_i^* = H^*[R_i]$  be the subgraph of the optimal solution induced by  $R_i$ . We also let  $H_i^M$  denote the subgraph of  $H_i$  returned by the min-cost flow algorithm run on  $R_i$  (i.e., the set of edges with non-zero flow), let  $H_i^{N^1}$  and  $H_i^{N^2}$  denote the subgraphs returned by the first and second 2-approximation 2-RSND algorithms run on  $R_i$ , respectively, and we let  $H_i^F$  denote the subgraph of  $H_i$  returned by the Steiner Forest algorithm on  $R_i$ . We also let  $M_i^*$  be the optimal solution to the Minimum-Cost Flow instance on  $R_i$ , let  $N_i^{1*}$  and  $N_i^{2*}$  be the optimal solutions to the first and second 2-RSND instances on  $R_i$ , respectively, and let  $F_i^*$  be the optimal solution to the Steiner Forest instance on  $R_i$ . Subgraph  $H_i^M$  is given by an exact algorithm, subgraphs  $H_i^{N^1}$  and  $H_i^{N^2}$  are given by a 2-approximation algorithm, and subgraph  $H_i^F$  is given by a  $(2 - \frac{1}{k})$ -approximation algorithm. Note that there are at most 4 terminal pairs in the Steiner Forest instance, so  $k \leq 4$  and the algorithm gives a  $\frac{7}{4}$ -approximation. Hence we have the following for each component  $R_i$ :

$$w(H_i^M) = w(M_i^*), \quad w(H_i^{N^1}) \leq 2w(N_i^{1*}), \quad w(H_i^{N^2}) \leq 2w(N_i^{2*}), \quad w(H_i^F) \leq \frac{7}{4}w(F_i^*).$$

Summing over all components in the chain, we get the following:

$$\begin{aligned} \sum_{i=0}^p w(H_i^M) &= \sum_{i=0}^p w(M_i^*), & \sum_{i=0}^p w(H_i^{N^1}) &\leq 2 \cdot \sum_{i=0}^p w(N_i^{1*}), \\ \sum_{i=0}^p w(H_i^{N^2}) &\leq 2 \cdot \sum_{i=0}^p w(N_i^{2*}), & \sum_{i=0}^p w(H_i^F) &\leq \frac{7}{4} \cdot \sum_{i=0}^p w(F_i^*). \end{aligned}$$

We also have that

$$w(H_i) \leq w(H_i^M) + w(H_i^{N^1}) + w(H_i^{N^2}) + w(H_i^F).$$

Summing over all components in the chain and then substituting the above, we get the following:

$$\begin{aligned} \sum_{i=0}^p w(H_i) &\leq \sum_{i=0}^p w(H_i^M) + \sum_{i=0}^p w(H_i^{N^1}) + \sum_{i=0}^p w(H_i^{N^2}) + \sum_{i=0}^p w(H_i^F) \\ &\leq \sum_{i=0}^p w(M_i^*) + 2 \cdot \sum_{i=0}^p w(N_i^{1*}) + 2 \cdot \sum_{i=0}^p w(N_i^{2*}) + \frac{7}{4} \cdot \sum_{i=0}^p w(F_i^*). \end{aligned}$$

The optimal subgraph  $H^*$  is a feasible solution, so by Lemma 22, each property in the lemma statement must be met on subgraph  $H_i^*$  for all  $i$ . For all properties in the lemma to be satisfied on  $H_i^*$ , the set of edges  $E(H_i^*)$  must be a feasible solution to each of the Minimum-Cost Flow, 2-RSND, and Steiner Forest instances on  $R_i$ . Therefore, the cost of  $H_i^*$  must be at least the cost of the optimal solution to each of the Minimum-Cost Flow, 2-RSND, and Steiner Forest instances. We therefore have the following:

$$\sum_{i=0}^p w(H_i) \leq \sum_{i=0}^p w(H_i^*) + 2 \sum_{i=0}^p w(H_i^*) + 2 \sum_{i=0}^p w(H_i^*) + \frac{7}{4} \sum_{i=0}^p w(H_i^*) \leq \frac{27}{4} \sum_{i=0}^p w(H_i^*).$$

Finally, we must account for the edges between components in the  $s - t$  2-chain. Let  $S$  be the set of edges between components in the chain that are included in the algorithm solution, and let  $S^*$  be the set of edges between components included in the optimal solution. By Lemma 22, any feasible solution must include all edges between the components of the chain. We therefore have that  $S = S^*$  and we get the following:

$$\begin{aligned} w(H) &= \sum_{i=0}^p w(H_i) + w(S) \leq \frac{27}{4} \sum_{i=0}^p w(H_i^*) + w(S) \leq \frac{27}{4} \left( \sum_{i=0}^p w(H_i^*) + w(S^*) \right) \\ &\leq \frac{27}{4} w(H^*). \end{aligned} \quad \blacktriangleleft$$

Theorem 5 is directly implied by Lemmas 24 and 24 together with the obvious observation that our algorithm runs in polynomial time.

---

## References

- 1 Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.
- 2 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: Generic and optimal. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 509–518, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897648.
- 3 Greg Bodwin, Michael Dinitz, and Yasamin Nazari. Vertex fault-tolerant emulators. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPICs*, pages 25:1–25:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.25.
- 4 Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1884–1900. SIAM, 2018.

- 5 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Optimal vertex fault-tolerant spanners in polynomial time. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, 2021.
- 6 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Optimal vertex fault-tolerant spanners in polynomial time. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 2924–2938. SIAM, 2022. doi:10.1137/1.9781611976465.174.
- 7 Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC '19*, pages 541–543, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331588.
- 8 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty.  $f$ -sensitivity distance oracles and routing schemes. In Mark de Berg and Ulrich Meyer, editors, *Algorithms – ESA 2010*, pages 84–96, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- 9 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM J. Comput.*, 39(7):3403–3423, 2010.
- 10 Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size  $k$ -connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000. doi:10.1137/S009753979833920X.
- 11 Julia Chuzhoy and Sanjeev Khanna. An  $o(k^3 \log n)$ -approximation algorithm for vertex-connectivity survivable network design. *Theory Comput.*, 8(1):401–413, 2012. doi:10.4086/toc.2012.v008a018.
- 12 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
- 13 Michael Dinitz, Ama Koranteng, and Guy Kortsarz. Relative survivable network design, 2022. doi:10.48550/ARXIV.2206.12245.
- 14 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178, 2011.
- 15 Michael Dinitz and Caleb Robelle. Efficient and simple algorithms for fault-tolerant spanners. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing*, pages 493–500. ACM, 2020. doi:10.1145/3382734.3405735.
- 16 Harold N. Gabow and Suzanne R. Gallagher. Iterated rounding algorithms for the smallest  $k$ -edge connected spanning subgraph. *SIAM Journal on Computing*, 41(1):61–103, 2012. doi:10.1137/080732572.
- 17 Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. Approximating the smallest  $k$ -edge connected spanning subgraph by lp-rounding. *Networks*, 53(4):345–357, 2009.
- 18 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. doi:10.1007/978-3-642-97881-4.
- 19 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.
- 20 Lap-Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, USA, 1st edition, 2011.
- 21 Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. A brief note on single source fault tolerant reachability, 2019. doi:10.48550/ARXIV.1904.08150.
- 22 Dániel Marx. Important separators and parameterized algorithms. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 5–10. Springer, 2011.
- 23 K. Steiglitz, P. Weiner, and D. Kleitman. The design of minimum-cost survivable networks. *IEEE Transactions on Circuit Theory*, 16(4):455–460, 1969. doi:10.1109/TCT.1969.1083004.
- 24 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995. doi:10.1007/BF01299747.



## A

 Counterexamples from Section 2

We show some counterexample to obvious approaches to  $k$ -EFTS; in particular, we show that our cut requirement function  $f_F$  is not weakly supermodular, and the most obvious cut requirement function  $f(S) = \min(k, |\delta_G(S)|)$  is also not weakly supermodular.

Recall that  $\delta_G(S)$  denotes the edges in  $G$  with exactly one endpoint in  $S$ . We extend this notation for disjoint sets  $A, B$  by letting  $\delta_G(A, B)$  denote the edges with one endpoint in  $A$  and one endpoint in  $B$ .

► **Theorem 26.** *The function  $f_F$  is not weakly supermodular.*

**Proof.** Consider the following example. Set  $k = 100$ . We create a graph  $G = (V, E)$  which has two sets  $A, B \subseteq V$  with the following properties.

$$\begin{aligned} |\delta_G(A \setminus B, V \setminus (A \cup B))| &= 49 & |\delta_G(B \setminus A, V \setminus (A \cup B))| &= 105 \\ |\delta_G(A \cap B, V \setminus (A \cup B))| &= 3 & |\delta_G(A \setminus B, B \setminus A)| &= 0 \\ |\delta_G(A \setminus B, A \cap B)| &= 2 & |\delta_G(B \setminus A, A \cap B)| &= 49 \end{aligned}$$

Anything not specified is extremely dense and well-connected, so an edge is in  $F$  if and only if it is part of a small cut made up of the above sets. It is not hard to see that the small cuts are precisely  $A \setminus B$  (since  $|\delta_G(A \setminus B)| = 49 + 0 + 2 = 51 < 100$ ) and  $A \cap B$  (since  $|\delta_G(A \cap B)| = 3 + 2 + 49 = 54 < 100$ ). All other cuts are large. Hence  $F$  consists of all edges involving  $A$  or  $B$  other than  $\delta_G(B \setminus A, V \setminus (A \cup B))$ , or more specifically,

$$\begin{aligned} F = & \delta_G(A \setminus B, V \setminus (A \cup B)) \cup \delta_G(A \cap B, V \setminus (A \cup B)) \\ & \cup \delta_G(A \setminus B, A \cap B) \cup \delta_G(B \setminus A, A \cap B). \end{aligned}$$

We can now calculate  $f_F$  on the subsets we care about:

$$\begin{aligned} f_F(A) &= 100 - 49 - 3 - 49 = -1 \\ f_F(B) &= 100 - 3 - 2 = 95 \\ f_F(A \setminus B) &= 0 & (A \setminus B \text{ is small}) \\ f_F(B \setminus A) &= 100 - 49 = 51 \\ f_F(A \cap B) &= 0 & (A \cap B \text{ is small}) \\ f_F(A \cup B) &= 100 - 49 - 3 = 48 \end{aligned}$$

Thus

$$f_F(A) + f_F(B) = 94 \quad f_F(A \setminus B) + f_F(B \setminus A) = 51 \quad f_F(A \cup B) + f_F(A \cap B) = 48$$

Hence  $f_F$  is not weakly supermodular. ◀

Note that the above example is not a contradiction of  $f$  being *locally* weakly supermodular since  $A$  is an empty cut.

► **Theorem 27.** *The function  $f = \min(k, |\delta_G(S)|)$  is not weakly supermodular.*


**Proof.** Consider the following example. Set  $k = 100$ . We create a graph  $G = (V, E)$  which has two sets  $A, B \subseteq V$  with the following properties. All of  $A \setminus B$  and  $B \setminus A$  and  $A \cap B$  and  $V \setminus (A \cup B)$  are extremely large and dense (e.g., large cliques). There are no edges between  $A \setminus B$ ,  $B \setminus A$ , or  $A \cap B$ . The other cut sizes are:

$$\begin{aligned} |\delta_G(A \cap B, V \setminus (A \cup B))| &= 55 \\ |\delta_G(A \setminus B, V \setminus (A \cup B))| &= 95 \\ |\delta_G(B \setminus A, V \setminus (A \cup B))| &= 95 \end{aligned}$$



Then it is easy to see that

$$\begin{array}{ll} f(A) = 100 & f(b) = 100 \\ f(A \setminus B) = 95 & f(B \setminus A) = 95 \\ f(A \cup B) = 100 & f(A \cap B) = 55 \end{array}$$

Hence  $f$  is not weakly supermodular. 



# Bypassing the XOR Trick: Stronger Certificates for Hypergraph Clique Number

Venkatesan Guruswami ✉

University of California Berkeley, CA, USA

Pravesh K. Kothari ✉

Carnegie Mellon University, Pittsburgh, PA, USA

Peter Manohar ✉

Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

Let  $\mathcal{H}(k, n, p)$  be the distribution on  $k$ -uniform hypergraphs where every subset of  $[n]$  of size  $k$  is included as an hyperedge with probability  $p$  independently. In this work, we design and analyze a simple spectral algorithm that certifies a bound on the size of the largest clique,  $\omega(H)$ , in hypergraphs  $H \sim \mathcal{H}(k, n, p)$ . For example, for any constant  $p$ , with high probability over the choice of the hypergraph, our spectral algorithm certifies a bound of  $\tilde{O}(\sqrt{n})$  on the clique number in polynomial time. This matches, up to polylog( $n$ ) factors, the best known certificate for the clique number in random graphs, which is the special case of  $k = 2$ .

Prior to our work, the best known refutation algorithms [4, 1] rely on a reduction to the problem of refuting random  $k$ -XOR via Feige's XOR trick [6], and yield a polynomially worse bound of  $\tilde{O}(n^{3/4})$  on the clique number when  $p = O(1)$ . Our algorithm bypasses the XOR trick and relies instead on a natural generalization of the Lovász theta semidefinite programming relaxation for cliques in hypergraphs.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Planted clique, Average-case complexity, Spectral refutation, Random matrix theory

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.42

**Category** APPROX

**Funding** *Venkatesan Guruswami*: Supported in part by NSF grant CCF-1908125 and a Simons Investigator award.

*Pravesh K. Kothari*: Supported in part by an NSF CAREER Award #2047933, a Google Research Scholar Award, and a Sloan Fellowship.

*Peter Manohar*: Supported in part by an ARCS Scholarship, NSF Graduate Research Fellowship (under grant numbers DGE1745016 and DGE2140739), and NSF CCF-1814603. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## 1 Introduction

In this work, we study the average-case search problem of finding refutations, i.e., certificates of the tightest possible upper bounds, on the clique number  $\omega(H)$  (size of the largest clique) of random  $k$ -uniform hypergraphs  $H$  drawn from the distribution  $\mathcal{H}(k, n, p)$ , where each hyperedge is included in  $H$  independently with probability  $p$ . A clique in a  $k$ -uniform hypergraph  $H$  is a set  $S$  of vertices such that all subsets  $C \subseteq S$  with  $|C| = k$  are edges in the hypergraph  $H$ , and we will adopt the convention that if  $|S| \leq k - 1$ , then  $S$  is trivially



© Venkatesan Guruswami, Pravesh K. Kothari, and Peter Manohar;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 42; pp. 42:1–42:7



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a clique. With high probability, the clique number of such hypergraphs is  $O(\log(n)^{\frac{1}{k-1}})$  for constant  $p$ , and approaches  $n$  as  $p$  approaches  $1 - O(n^{-(k-1)})$  [11]. Our goal is to find polynomial time algorithms that certify a bound as close to this ground truth as possible.

In the case of Erdős-Renyi random graphs from  $G(n, p)$ , i.e., when  $k = 2$ , the Lovász theta function provides a semidefinite programming relaxation that certifies a bound of  $O(\sqrt{n})$  in polynomial time with high probability over the draw of the graph, when  $p = O(1)$ . A long line of work [7, 8, 13, 5, 9, 14, 3] has explored the power of spectral methods and semidefinite programming hierarchies for improving on this bound. This question is also closely related to the *planted clique* problem [10, 12, 2, 7, 8], where the size of the cliques that can be efficiently recovered is similar to the best known polynomially-certifiable upper bounds on the clique number of random graphs.

For  $k > 2$ , the problem was first studied by Coja-Oghlan, Goerdts and Lanka [4], who provided a polynomial time algorithm based on spectral methods to certify an upper bound of  $\varepsilon n$  on the clique number of random 3- and 4-uniform hypergraphs, where  $\varepsilon$  is a constant. Unlike the case of  $k = 2$ , their algorithm relies on a reduction, via the famous XOR trick of Feige [6], to the problem of refuting random  $k$ -XOR formulas. Specifically, they construct a polynomial  $f(x) = \frac{1}{\binom{n}{k}} \sum_{C \in \binom{[n]}{k}} b_C \prod_{i \in C} x_i$ , where  $b_C = 1 - p$  if  $C \in H$  and  $b_C = -p$  if  $C \notin H$ , and they show that (1) if  $\omega(H)$  is large, then  $f(x)$  is large for some  $x \in [-1, 1]^n$ , and (2) with high probability over  $H \sim \mathcal{H}(k, n, p)$ , their  $k$ -XOR refutation algorithm certifies a nontrivial upper bound on  $\max_{x \in [-1, 1]^n} f(x)$ , and thus on  $\omega(H)$ . This connection was later utilized by Allen, O’Donnell and Witmer [1] who used their  $k$ -XOR refutation algorithms to improve the bounds and handle the case of all  $k \geq 3$ . For any  $k$ , when  $p = O(1)$ , the algorithm of [1] certifies  $\omega(H) \leq \tilde{O}(n^{3/4})$ .<sup>1</sup>

In this paper, we show that the certificates obtained via the “XOR method” are in fact suboptimal by providing a substantially improved certificate for the clique numbers of random hypergraphs at all densities  $p$ . Our certificates are based on a natural generalization of the “direct” Lovász SDP for clique numbers of graphs. The bounds obtained by our algorithm for any fixed  $k$  match those obtained in the case of graphs (i.e.,  $k = 2$ ) up to polylogarithmic factors in  $n$ .<sup>2</sup> Specifically, we show:

► **Theorem 1** (Theorem 3, specialized to  $\text{poly}(n)$ -time and  $p = O(1)$ ). *There is an algorithm that takes as input a  $k$ -uniform hypergraph  $H$  on  $n$  vertices and a parameter  $p \in [0, 1]$  with  $p = O(1)$ , and outputs in  $n^{O(k)}$ -time a value  $\omega_{\text{alg}}(H) \in [0, n]$  with the following two properties:*

- (1) *Completeness:*  $\omega(H) \leq \omega_{\text{alg}}(H)$ , for all  $H$ .
- (2) *High probability bound:* If  $H \sim \mathcal{H}(k, n, p)$ , then with probability  $1 - 1/\text{poly}(n)$ ,  $\omega_{\text{alg}}(H) \leq \tilde{O}(\sqrt{n})$ .

The result above is based on a (surprisingly) simple spectral algorithm that is a natural analog, for  $k$ -uniform hypergraphs, of the algorithm that uses the spectral norm of the adjacency matrix of a graph to certify an upper bound on its clique number. This is in contrast to the methods from [4, 1] that rely on a reduction to refuting random  $k$ -XOR formulas.

<sup>1</sup> We use the notation  $\tilde{O}(f(n))$  to mean  $O(f(n)\text{polylog}(n))$ .

<sup>2</sup> We believe that with a more fine-grained analysis, our bound can be improved from  $\tilde{O}(\sqrt{n})$  to  $O(\sqrt{n})$ . However, for simplicity we use “off-the-shelf” concentration inequalities, which lose  $\text{polylog}(n)$  factors over sharper methods.

It is easy to observe that a clique in  $H$  is an independent set in the complement hypergraph  $\bar{H} := \{C : C \notin H\}$ , and that  $\bar{H} \sim \mathcal{H}(k, n, 1-p)$  when  $H \sim \mathcal{H}(k, n, p)$ . Thus, Theorem 1 certifies a bound of  $\tilde{O}(\sqrt{n})$  on the size of the maximum independent set in a random  $H \sim \mathcal{H}(k, n, 1-p)$  with high probability, and hence also certifies with high probability that the chromatic number of a random  $H$  is at least  $\tilde{\Omega}(\sqrt{n})$ .

Theorem 1 is a special case of our more general theorem (Theorem 3), which we present in full in Section 2. The algorithm in Theorem 3 has a tradeoff between the runtime and the strength of the certificate, and also handles the more general case of nonconstant  $p$ .

► **Remark 2 (Detecting planted cliques vs. refutation).** We note that it is easy to *distinguish* between a random  $H \sim \mathcal{H}(k, n, 1/2)$  and a random  $H \sim \mathcal{H}(k, n, 1/2)$  with a *planted* clique of size  $\tilde{\Theta}(\sqrt{n})$ ; the extra  $\text{polylog}(n)$  factor makes the degrees of vertices in the planted clique be noticeably larger than other vertices, so one can easily extract the planted clique. However, we are considering the formally harder task of *refutation*, so merely being able to distinguish a random  $H \sim \mathcal{H}(k, n, 1/2)$  from a  $H$  from this particular planted distribution is insufficient. For example, it is easy to construct a hypergraph  $H$  where the vertices in the planted clique do not have larger-than-average degree, which would, e.g., trivially fool the aforementioned simple distinguisher.

## 2 The Spectral Algorithm

In this section, we prove the following theorem, which has Theorem 1 as a special case.

► **Theorem 3.** *There is an algorithm that takes as input a  $k$ -uniform hypergraph  $H$  on  $n$  vertices, a parameter  $p := p(n) \in [0, 1]$ , and an integer  $d := d(n) \geq 1$ , and outputs in  $n^{O(d+k)}$ -time a value  $\omega_{\text{alg}}(H) \in [0, n]$  with the following two properties:*

- (1) *Completeness:*  $\omega(H) \leq \omega_{\text{alg}}(H)$ , for all  $H$ .
- (2) *High probability bound:* If  $H \sim \mathcal{H}(k, n, p)$ , then with probability  $1 - 1/\text{poly}(n)$ ,

$$\omega_{\text{alg}}(H) \leq d + O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k'}} \sqrt{\max(np^{\binom{d}{k'-1}}, d \log n)},$$

where  $k' = k$  if  $k$  is even, and  $k' = k - 1$  if  $k$  is odd.

Before continuing with the proof, we first interpret Theorem 3 and compare it to the works of [4, 1].

When  $k = 3$ , [4] certifies  $\omega(H) \leq \varepsilon n$  for constant  $\varepsilon$  when  $p \leq 1 - O(n^{-3/2})$ , and when  $k = 4$ , [4] certifies  $\omega(H) \leq \varepsilon n$  for constant  $\varepsilon$  for  $p \leq 1 - O(\frac{1}{n^2})$ . [1] improves upon [4] and certifies  $\omega(H) \leq \tilde{O}(n^{3/4+\theta/2k})$  when  $p = 1 - \tilde{O}(n^{-\theta})$ ; in particular, for, e.g.,  $p = \frac{1}{2}$ , they certify that  $\omega(H) \leq \tilde{O}(n^{3/4})$ , and this bound gets worse as  $p$  increases. When  $p = O(1)$ , our algorithm certifies a bound of  $\omega \leq \tilde{O}(\sqrt{n})$  in polynomial time for any fixed  $k$ . Theorem 3 thus beats the current best known algorithm in [1] by a factor of  $n^{1/4}$ , i.e., a *polynomial* factor.

More generally, for any  $d \in \mathbb{N}$ , our algorithm certifies a bound of  $\omega \leq \tilde{O}(1) + \tilde{O}(\sqrt{np^{\binom{d}{k'-1}}})$  in time  $n^{O(d+k)}$ . On the other hand, for  $H \sim \mathcal{H}(k, n, p)$  with  $p \leq 1 - O(n^{-(k-1)})$ , [11] shows that with high probability, the true clique size is  $\omega(H) \leq O((1-p)^{-\frac{1}{k-1}} (\log(n^{k-1}(1-p)))^{\frac{1}{k-1}})$ . So, for, e.g.,  $d = O(\log n)$  and  $p \leq O(1)$ , our algorithm outputs  $\omega_{\text{alg}}(H) \leq \tilde{O}(1)$ , which is the true clique number up to  $\text{polylog}(n)$  factors.

We now turn to the proof of Theorem 3.

**Proof of Theorem 3.** We break the proof of Theorem 3 into three steps. First, we give a simple refutation algorithm that achieves the guarantees of Theorem 3 when  $k$  is even and  $d = 1$ . Then, we prove the case when  $k$  is even and  $d$  is arbitrary by reduction to the case when  $d = 1$ . Finally, we reduce the case of odd  $k$  to the case of even  $k$ .

For a set  $S$  and integer  $t$ , we will let  $\binom{S}{t}$  denote the set of all subsets of  $S$  of size exactly  $t$ . E.g.,  $\binom{[n]}{k} := \{C \subseteq [n], |C| = k\}$ .

**The basic refutation algorithm.** We first give a basic refutation algorithm. This algorithm achieves the guarantees of Theorem 3 in the case when  $k$  is even and  $d = 1$ .

► **Lemma 4.** *Let  $k$  be even. There is an algorithm  $\mathcal{A}$  that takes as input a  $k$ -uniform hypergraph  $H$  on  $n$  vertices and a parameter  $p := p(n) \in [0, 1]$ , and outputs in  $n^{O(k)}$ -time a value  $\omega_{\text{alg}}(H) \in [0, n]$  with the following two properties:*

- (1) *Completeness:*  $\omega(H) \leq \omega_{\text{alg}}(H)$ , for all  $H \subseteq \binom{[n]}{k}$ .
- (2) *High probability bound:* If  $H \sim \mathcal{H}(k, n, p)$ , then for any  $c \geq 1$ , with probability  $1 - n^{-c}$ ,

$$\omega_{\text{alg}}(H) \leq O(k) \left( \frac{c \log n}{1-p} \right)^{\frac{2}{k}} \sqrt{n}.$$

We prove Lemma 4 in Section 2.1.

**Case 1:  $k$  is even.** We now prove Theorem 3 when  $k$  is even. To do this, we need the following claim.

► **Claim 5.** Let  $H$  be a  $k$ -uniform hypergraph on  $n$  vertices, let  $d \geq k - 1$  be a positive integer, and let  $J \subseteq [n]$  be a set of size  $d$ . Let  $V_J = \{i \in [n] \setminus J : \forall J' \in \binom{J}{k-1}, J' \cup \{i\} \in H\}$ , and let  $H_J = \binom{V_J}{k} \cap H$ . Then,  $\omega(H) \leq d + \max_{J \in \binom{[n]}{d}} \omega(H_J)$ . Moreover, if  $H \sim \mathcal{H}(k, n, p)$ , then  $|V_J| \sim \text{Bin}(n - d, p^{\binom{d}{k-1}})$ , and conditioned on  $V_J$ ,  $H_J \sim \mathcal{H}(k, |V_J|, p)$ .

We prove Claim 5 in Section 2.2

With Claim 5 in hand, we finish the proof of Theorem 3 when  $k$  is even. Let  $\mathcal{A}$  be the algorithm in Lemma 4, and let the algorithm  $\mathcal{A}'$  operate as follows: on input  $H$ , (1) enumerate over all  $J \in \binom{[n]}{d}$  and compute  $\mathcal{A}(H_J)$ , and then (2) output  $d + \max_{J \in \binom{[n]}{d}} \mathcal{A}(H_J)$ . Clearly,  $\mathcal{A}'$  runs in  $n^{O(d)} \cdot n^{O(k)} = n^{O(d+k)}$  time. We observe that by Lemma 4 and Claim 5, we clearly have that  $\omega(H) \leq d + \max_{J \in \binom{[n]}{d}} \omega(H_J) \leq d + \max_{J \in \binom{[n]}{d}} \mathcal{A}(H_J) = \mathcal{A}'(H)$ , so Item 1 holds.

We now prove Item 2. Fix  $J \in \binom{[n]}{d}$ . We observe that by Claim 5,  $V_J \sim \text{Bin}(n - d, p^{\binom{d}{k-1}})$ . We bound  $|V_J|$  using the standard Chernoff bound, which we recall below.

► **Fact 6 (Chernoff Bound).** *Let  $X \sim \text{Bin}(n, p)$ , and let  $\delta \geq 0$ . Then,  $\Pr[X \geq (1 + \delta)np] \leq \exp(-\frac{\delta^2 np}{2 + \delta})$ .*

Fact 6 implies that  $|V_J| \leq \max(np^{\binom{d}{k-1}}, d \log n)$  with probability  $\geq 1 - n^{-2d}$ . Now, conditioned on  $|V_J|$ , by Claim 5 we have that  $H_J \sim \mathcal{H}(k, |V_J|, p)$ . Hence, if  $|V_J| \geq 1$ , setting  $c = O(d \log n)$  in Lemma 4, we have that with probability  $\geq 1 - n^{-2d}$ ,  $\mathcal{A}(H_J) \leq O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k}} \sqrt{|V_J|}$ . (If  $|V_J| = 0$ , then  $\mathcal{A}(H_J) = k - 1$ .) Hence, for a fixed  $J$ , with probability  $\geq 1 - 2n^{-2d}$ , we have  $\mathcal{A}(H_J) \leq O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k}} \sqrt{\max(np^{\binom{d}{k-1}}, d \log n)}$ . By union bound over all  $J$ , we thus conclude that  $\mathcal{A}'(H) \leq d + O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k}} \sqrt{\max(np^{\binom{d}{k-1}}, d \log n)}$  with probability  $\geq 1 - 2n^{-d} = 1 - 1/\text{poly}(n)$ . This finishes the proof of Theorem 3 when  $k$  is even.

**Case 2:  $k$  is odd** We now turn to the case when  $k$  is odd. For the odd case, we use the following claim, which we prove in Section 2.3.

▷ **Claim 7.** Let  $H$  be a  $k$ -uniform hypergraph on  $n$  vertices with  $k \geq 3$ . For each  $i \in [n]$ , let  $H_i = \{C \setminus \{i\} : C \in H \wedge i \in C\}$ . Then,  $\omega(H) \leq 1 + \max_{i \in [n]} \omega(H_i)$ . Moreover, if  $H \sim \mathcal{H}(k, n, p)$ , then for any fixed  $i \in [n]$ ,  $H_i \sim \mathcal{H}(k, n-1, p)$ .

Let  $\mathcal{A}'$  be the algorithm in Theorem 3 when  $k$  is even, described earlier. Let  $\mathcal{A}''$  be the algorithm that operates as follows: on input  $H$ , a  $k$ -uniform hypergraph where  $k$  is odd, (1) for each  $i \in [n]$ , compute  $\mathcal{A}'(H_i)$ , (2) output  $1 + \max_{i \in [n]} \mathcal{A}'(H_i)$ . Clearly,  $\mathcal{A}''$  runs in  $n^{O(d+k)}$  time, and by Claim 7, we have that  $\omega(H) \leq 1 + \max_{i \in [n]} \omega(H_i) \leq 1 + \max_{i \in [n]} \mathcal{A}'(H_i) = \mathcal{A}''(H)$ . Thus, Item 1 holds. To see Item 2, we observe that by Claim 7,  $H_i \sim \mathcal{H}(k, n-1, p)$ . Hence, with probability  $\geq 1 - 2n^{-d}$ , it holds that  $\mathcal{A}'(H_i) \leq d + O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k-1}} \sqrt{\max(np^{\binom{d}{k-2}}, d \log^2 n)}$ . By union bound over the choice of  $i$ , we see that with probability  $1 - 1/\text{poly}(n)$ ,

$$\mathcal{A}''(H) \leq d + O(k) \left( \frac{d \log^2 n}{1-p} \right)^{\frac{2}{k-1}} \sqrt{\max(np^{\binom{d}{k-2}}, d \log^2 n)},$$

which finishes the proof of Theorem 3. ◀

## 2.1 The basic algorithm: proof of Lemma 4

**Proof.** For  $C \in \binom{[n]}{k}$ , let  $A_C \in \mathbb{R}^{\binom{[n]}{k/2} \times \binom{[n]}{k/2}}$  be the matrix where  $A_C(S, T) = 1$  if  $S \cup T = C$ , and 0 otherwise. Note that this implies that  $S \cap T = \emptyset$  also.

Let  $A = \sum_{C \in \binom{[n]}{k}} b_C A_C$ , where  $b_C = 1 - p$  if  $C \in H$ , and  $b_C = -p$  if  $C \notin H$ . The output of the algorithm is  $\omega_{\text{alg}}(H) := \omega = \max(k-1, k \left( \frac{p}{1-p} \binom{k}{k/2} \cdot \|A\|_2^2 \right)^{\frac{1}{k}})$ . We observe that  $A$  can be constructed in  $n^{O(k)}$  time and has size  $n^{O(k)}$ , and so we can compute  $\|A\|_2$  (and thus also  $\omega$ ) in  $n^{O(k)}$  time.

We now prove Item 1. Let  $I \subseteq [n]$  be a clique in  $H$ . If  $|I| \leq k-1$ , then we are done, as  $\omega \geq k-1$  always holds. So, suppose that  $|I| \geq k$ . Let  $x \in \mathbb{R}^{\binom{[n]}{k/2}}$  be defined as  $x_S = 1$  if  $S \subseteq I$ , and 0 otherwise. Note that  $\|x\|_2^2 = \binom{|I|}{k/2}$ . We observe that  $x^\top A x$  is simply  $\sqrt{\frac{1-p}{p}} \binom{|I|}{k} \cdot \binom{k}{k/2}$ . This is because for each  $C \in \binom{[n]}{k}$ , there are  $\binom{k}{k/2}$  ways to partition  $C$  into  $(S, T)$ , and all such  $C$  are in  $H$ , and thus  $b_C = 1 - p$ . We thus conclude that

$$\begin{aligned} \|A\|_2 &\geq \frac{x^\top A x}{\|x\|_2^2} = (1-p) \binom{|I|}{k} \cdot \binom{k}{k/2} \cdot \frac{1}{\binom{|I|}{k/2}} \\ &\geq (1-p) \sqrt{\binom{|I|}{k} \binom{k}{k/2}} \geq (1-p) \sqrt{\left( \frac{|I|}{k} \right)^k \binom{k}{k/2}}, \end{aligned}$$

where we use that  $\frac{\binom{|I|}{k}}{\binom{|I|}{k/2}^2} \geq \frac{1}{\binom{k}{k/2}}$ , and that  $\binom{|I|}{k} \geq \left( \frac{|I|}{k} \right)^k$ . Hence, we have shown that

$$\omega \geq k \left( \frac{1}{(1-p)^2} \binom{k}{k/2} \cdot \|A\|_2^2 \right)^{\frac{1}{k}} \geq |I|$$

for any clique  $I$  in  $H$  with  $|I| \geq k$ . This finishes the proof of Item 1.

We now prove Item 2. The key step here is to show an upper bound on  $\|A\|_2$ , with high probability over  $H \sim \mathcal{H}(k, n, p)$ . We will do this by applying the standard Matrix Bernstein concentration inequality, which we recall below.



► **Fact 8** (Matrix Bernstein, Theorem 1.4 of [15]). *Let  $X_1, \dots, X_k$  be independent random  $n \times n$  symmetric matrices with  $\mathbb{E}[X_i] = 0$  and  $\|X_i\|_2 \leq R$  for all  $i$ . Let  $\sigma^2 \geq \|\mathbb{E}[\sum_{i=1}^k X_i^2]\|_2$ . Then for any  $c > 0$ ,  $\Pr[\|\sum_{i=1}^k X_i\|_2 \geq O(Rc \log n + \sigma\sqrt{c \log n})] \leq n^{-c}$ .*

We observe that  $A = \sum_{C \in \binom{[n]}{k}} b_C A_C$  is the sum of  $\binom{n}{k}$  independent, mean 0 random matrices. We have that  $\|A_C\|_2 \leq R := \max(p, 1-p) = 1$  for every  $C$ , as each row/column of  $A_C$  has at most one nonzero entry, which is at most  $R$  in magnitude. We also observe that  $\mathbb{E}[A^2] = \sum_C A_C^2$  is a diagonal matrix, where the  $S$ -th diagonal entry is  $\leq \binom{n-k/2}{k-k/2} = \binom{n-k/2}{k/2} \leq n^{k/2}$ , as each  $A_C^2$  is diagonal, has  $\mathbb{E}[b_C^2]$  in the  $S$ -th diagonal entry if  $S \subseteq C$ , and  $\mathbb{E}[b_C^2] \leq 1$ . Hence, by Fact 8, with probability  $1 - n^{-c}$ , we have that

$$\begin{aligned} \|A\|_2 &\leq O(Rc \log n^{k/2}) + O(\sqrt{cn^{k/2} \log n^{k/2}}) \leq O(n^{\frac{k}{4}} ck \log n) \\ \implies k \left( \frac{1}{(1-p)^2} \binom{k}{k/2} \cdot \|A\|_2^2 \right)^{\frac{1}{k}} &\leq k \left( \frac{1}{(1-p)^2} \binom{k}{k/2} \cdot O(n^{k/2} c^2 k^2 \log^2 n) \right)^{\frac{1}{k}} \\ &\leq O(k) \left( \frac{c \log n}{1-p} \right)^{\frac{2}{k}} \sqrt{n}. \end{aligned}$$

Finally, we have that

$$\omega = \max(k-1, O(k) \left( \frac{c \log n}{1-p} \right)^{\frac{2}{k}} \sqrt{n}) = O(k) \left( \frac{c \log n}{1-p} \right)^{\frac{2}{k}} \sqrt{n}. \quad \blacktriangleleft$$

## 2.2 Reduction for larger $d$ : proof of Claim 5

**Proof.** Let  $I$  be a clique in  $H$  with  $|I| = \omega(H)$ . Let  $J \subseteq I$  be an arbitrary subset of size  $d$ . We claim that  $I' := I \setminus J$  is a clique in  $H_J$ . Indeed, we first observe that for each  $i \in I'$ , we have  $i \in V_J$ , as for any  $J' \in \binom{J}{k-1}$ , we have  $J' \cup \{i\}$  is a subset of  $I$  of size  $k$ , and hence is in  $H$ . Next, let  $C \subseteq I'$  be any subset of size  $k$  (if  $|I'| \leq k-1$ , so that no such  $C$  exists, then  $I'$  is trivially a clique in  $H_J$ ). Then,  $C \in H$ , as  $I$  was a clique, and so  $C \in H_J$ . Hence,  $I'$  is a clique in  $H_J$ . As  $\omega(H) = |J| + |I'| = d + |I'| \leq d + \omega(H_J)$ , this proves the first part of the claim.

For the second part of the claim, we think of sampling  $H$  as follows. First, for every  $J' \in \binom{J}{k-1}$  and  $i \in [n]$ , add  $J' \cup \{i\}$  to  $H$  with probability  $p$ . Then, add every other  $C \in \binom{[n]}{k}$  to  $H$  with probability  $p$ . We note that  $H \sim \mathcal{H}(k, n, p)$  clearly, and that after the first step, we have determined  $V_J$ . In the first step, we see that  $i$  is added to  $V_J$  independently for each  $i \notin J$ , and each  $i$  is added with probability  $p^{\binom{d}{k-1}}$ . Hence,  $|V_J| \sim \text{Bin}(n-d, p^{\binom{d}{k-1}})$ . As all the hyperedges in  $H_J$  are sampled in the second step, the claim follows.  $\blacktriangleleft$

## 2.3 Reduction from odd $k$ to even $k$ : proof of Claim 7

**Proof.** Let  $I \in H$  be a clique with  $|I| = \omega(H)$ . If  $|I| = k-1$ , then we are done, as  $\omega(H_i) \geq k-2$  for all  $i$  since  $k \geq 3$ . So, suppose  $|I| \geq k$ . Let  $i \in I$ , and let  $J = I \setminus \{i\}$ . Then,  $J$  is a clique in  $H_i$ . Indeed, for any  $C' \in \binom{J}{k-1}$ , we must have  $C' \cup \{i\} \in H$ , and therefore we have  $C' \in H_i$ . So, it follows that  $\omega(H_i) \geq |J| = |I| - 1$ , which finishes the proof of the first part of the claim.

For the second part, we observe that if  $H \sim \mathcal{H}(k, n, p)$ , then each  $C \in \binom{[n]}{k}$  with  $i \in C$  is added to  $H$  independently with probability  $p$ . So, each  $C' \in \binom{[n] \setminus \{i\}}{k-1}$  is added to  $H_i$  independently with probability  $p$ , which finishes the proof.  $\blacktriangleleft$

## References

- 1 Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 689–708, 2015.
- 2 Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. In *SODA*, pages 594–598. ACM/SIAM, 1998.
- 3 Boaz Barak, Samuel B. Hopkins, Jonathan A. Kelner, Pravesh Kothari, Ankur Moitra, and Aaron Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. In *FOCS*, pages 428–437. IEEE Computer Society, 2016.
- 4 Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random  $k$ -sat. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques*, volume 3122 of *Lecture Notes in Computer Science*, pages 310–321. Springer, 2004.
- 5 Yash Deshpande and Andrea Montanari. Improved sum-of-squares lower bounds for hidden clique and hidden submatrix problems. In *COLT*, volume 40 of *JMLR Workshop and Conference Proceedings*, pages 523–562. JMLR.org, 2015.
- 6 Uriel Feige. Relations between average case complexity and approximation complexity. In *STOC*, pages 534–543. ACM, 2002.
- 7 Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000.
- 8 Uriel Feige and Robert Krauthgamer. The probable value of the lovász-schrijver relaxations for maximum independent set. *SIAM J. Comput.*, 32(2):345–370, 2003.
- 9 Samuel B. Hopkins, Pravesh K. Kothari, and Aaron Potechin. Sos and planted clique: Tight analysis of MPW moments at all degrees and an optimal lower bound at degree four. *CoRR*, abs/1507.05230, 2015.
- 10 Mark Jerrum. Large cliques elude the metropolis process. *Random Struct. Algorithms*, 3(4):347–360, 1992.
- 11 Michael Krivelevich and Benny Sudakov. The chromatic numbers of random hypergraphs. *Random Struct. Algorithms*, 12(4):381–403, 1998.
- 12 Ludek Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.
- 13 Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In *STOC*, pages 87–96. ACM, 2015.
- 14 Prasad Raghavendra and Tselil Schramm. Tight lower bounds for planted clique in the degree-4 SOS program. *CoRR*, abs/1507.05136, 2015.
- 15 Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, August 2012. doi:10.1007/s10208-011-9099-z.



# Approximating CSPs with Outliers

Suprovat Ghoshal ✉

University of Michigan, Ann Arbor, MI, USA

Anand Louis ✉

Indian Institute of Science, Bangalore, India

---

## Abstract

Constraint satisfaction problems (CSPs) are ubiquitous in theoretical computer science. We study the problem of STRONG-CSPs, i.e. instances where a large induced sub-instance has a satisfying assignment. More formally, given a CSP instance  $\mathcal{G}(V, E, [k], \{\Pi_{ij}\}_{(i,j) \in E})$  consisting of a set of vertices  $V$ , a set of edges  $E$ , alphabet  $[k]$ , a constraint  $\Pi_{ij} \subseteq [k] \times [k]$  for each  $(i, j) \in E$ , the goal of this problem is to compute the largest subset  $S \subseteq V$  such that the instance induced on  $S$  has an assignment that satisfies all the constraints.

In this paper, we study approximation algorithms for UNIQUEGAMES and related problems under the STRONG-CSP framework when the underlying constraint graph satisfies mild expansion properties. In particular, we show that given a STRONGUNIQUEGAMES instance whose optimal solution  $S^*$  is supported on a regular low threshold rank graph, there exists an algorithm that runs in time exponential in the threshold rank, and recovers a large satisfiable sub-instance whose size is independent on the label set size and maximum degree of the graph. Our algorithm combines the techniques of Barak-Raghavendra-Steurer (FOCS'11), Guruswami-Sinop (FOCS'11) with several new ideas and runs in time exponential in the threshold rank of the optimal set. A key component of our algorithm is a new threshold rank based spectral decomposition, which is used to compute a “large” induced subgraph of “small” threshold rank; our techniques build on the work of Oveis Gharan and Rezaei (SODA'17), and could be of independent interest.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis

**Keywords and phrases** Constraint Satisfaction Problems, Strong Unique Games, Threshold Rank

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.43

**Category** APPROX

**Related Version** Full Version: <https://arxiv.org/pdf/2205.11328.pdf>

**Funding** Anand Louis: Supported in part by SERB Award ECR/2017/003296, a Pratiksha Trust Young Investigator Award, and an IUSSTF virtual center on “Polynomials as an Algorithmic Paradigm”.

**Acknowledgements** The authors thank the anonymous reviewers for their helpful suggestions and comments.

## 1 Introduction

An instance of a 2-Constraint Satisfaction Problem (2-CSP)  $\mathcal{G}(V, E, [k], \{\Pi_{ij}\}_{(i,j) \in E})$  consists of a set of vertices  $V$ , a set of edges  $E$ , alphabet  $[k]$ , and a constraint  $\Pi_{ij} \subseteq [k] \times [k]$  for each  $(i, j) \in E$ . The goal of this problem is to compute an assignment  $f : V \rightarrow [k]$  such that the fraction of constraints satisfied is maximized; this optimal fraction is also called the *value* of this instance, and is formally denoted by  $\text{Val}(\mathcal{G})$ . Many common optimization problems such as Max Cut, Unique Games, Graph Coloring, 2-SAT, etc. are 2-CSPs. Designing approximation algorithms for specific CSPs are central problems in the study of algorithms and have been studied extensively, for e.g., Max-Cut [20], Unique Games [11, 12], etc. There is also a long line of work which deal with algorithms for general CSPs (see [35, 36, 8, 21]).



© Suprovat Ghoshal and Anand Louis;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 43; pp. 43:1–43:16



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A particular parameter regime of interest is when the CSP instance is “almost” fully satisfiable. There are several ways for quantifying this, one of which is by asking the value of the CSP instance be close to 1. This can also be viewed as the setting where deleting a small number of edges from the instance results in an instance that is fully satisfiable. There has been extensive work on designing algorithms for CSPs in this regime; we give a brief survey in Section 1.2. Another way a CSP can be almost satisfiable is if a small number of *outlier* vertices can be deleted (all the edges incident on these vertices would also be deleted) to obtain an instance which is fully satisfiable. The main focus of our work is to study algorithms for CSPs in this model; we define it below formally.

► **Problem 1 (STRONG-CSP).** *Given an instance  $\mathcal{G}(V, E, [k], \{\Pi_{ij}\}_{(i,j) \in E})$  consisting of a set of vertices  $V$ , a set of edges  $E$ , alphabet  $[k]$ , and a constraint  $\Pi_{ij} \subseteq [k] \times [k]$  for each  $(i, j) \in E$ , compute the largest subset  $S \subseteq V$  such that the instance induced on  $S$  has value 1.*

We refer to an optimal set of vertices for Problem 1 as *good vertices*<sup>1</sup>, and denote them by  $V_{\text{good}}$ . A naturally arising such instantiation of STRONG-CSP’s is the ODDCYCLETRANSVERSAL problem. Here, given a graph  $G = (V, E)$  as input, the objective is to delete the smallest fraction of vertices so that the graph induced on the remaining vertices is bipartite. This is easily seen as an instance of a STRONG-CSP – here the predicate on the edges is the “Not Equals” predicate on the label set  $\{0, 1\}$ . ODDCYCLETRANSVERSAL is a well studied problem. In general, it is known to be constant factor inapproximable [7] (assuming the Unique Games Conjecture), and the best known upper bounds (in terms of fraction of vertices deleted) are  $O(\delta \sqrt{\log |V|})$  [1] and  $O(\sqrt{\delta \log d})$  [18] – where  $\delta$  is the optimal fraction of vertices to be deleted and  $d$  is the maximum degree of the graph – the latter bound is also tight upto constant factors assuming the Unique Games Conjecture [18]. Given these worst case bounds, one might ask if there are natural classes of instances under which ODDCYCLETRANSVERSAL admits better approximation?

For the specific setting of ODDCYCLETRANSVERSAL, there are several such classes which exhibit improved approximation guarantees. For instance, for the setting of planar graphs, the natural linear programming relaxation is known to be exact [14], and therefore admits an exact polynomial time algorithm. Furthermore, for  $K_r$ -minor closed graphs, Alev and Lau [2] gave an  $O(r)$ -approximation algorithm. On the other hand, since ODDCYCLETRANSVERSAL is fixed parameter tractable with respect to treewidth [26], it admits exact polynomial time algorithms for graphs with bounded treewidth. Note that these also happen to be characterizations which end up implying easy instances for Max-CSPs. Motivated by this connection, we investigate whether there are spectral characterizations under which ODDCYCLETRANSVERSAL (and more generally, STRONG-CSP’s) admit improved approximation. In particular, we study instances which are expanding, or more generally, have low threshold rank. Formally, the threshold rank of a graph is defined as follows.

► **Definition 2 (Threshold rank).** *Given an undirected graph  $G = (V, E)$ , let  $A$  denote its weighted adjacency matrix and let  $D$  denote the diagonal matrix where  $D(i, i)$  is the weighted degree of vertex  $i$ . The  $(1 - \varepsilon)$  threshold rank of  $G$ , denoted by  $\text{rank}_{\geq 1-\varepsilon}(G)$  is defined as the number of eigenvalues of  $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$  that are greater than or equal to  $1 - \varepsilon$ .*

In the setting of CSPs, low threshold rank instances have been studied extensively – the study of such instances was instrumental in the development of sub-exponential time algorithms for UNIQUEGAMES and SMALLSETEdgeEXPANSION [24, 3, 8]. In particular, for

<sup>1</sup> Note that such a set of vertices may not be unique, in which case, we will fix such a collection of vertices, and call it the set of good vertices.

the edge deletion analogue of ODDCYCLETRANSVERSAL i.e., MAX-CUT, [8] gave a  $(1/\lambda_t)$ -approximation algorithm running in time  $n^{\text{poly}(t)}$ , where  $\lambda_t$  is the  $t^{\text{th}}$  largest eigenvalue of the normalized Laplacian. Surprisingly, to the best of our knowledge, no such analogous results are known for ODDCYCLETRANSVERSAL. Furthermore, random instances of CSPs are expanding, and naturally have low threshold rank. This motivates us to explore the approximability of ODDCYCLETRANSVERSAL and other STRONG-CSP's in low threshold instances. In fact, we study them under the more stringent setting where only the graph induced on good vertices (constituting the fully satisfiable sub-instance) is assumed to have low threshold rank, as opposed to the full graph having low threshold rank.

**Max-CSPs vs. StrongCSPs.** This choice of the setting, in addition to making the task more challenging, is also motivated by our wish to exhibit a separation between the approximability of edge deletion and vertex deletion problems, i.e., namely MAX-CSPs and STRONG-CSP's. We point out that under an identical setting (where only a  $(1 - \delta)$ -sized subset has low threshold rank), MAX-CSPs can be arbitrarily hard to approximate. Indeed, consider a MAX-CSP instance where the  $(1 - \delta)$ -sized subset  $V_{\text{good}}$  induces a constant degree expander with trivially satisfiable constraints, and the edges going across  $V_{\text{good}}, V_{\text{good}}^c$  encode a denser hard to approximate UNIQUE GAME instance with large gap and larger vertex degrees. It is easy to see that such instances do not admit efficient constant factor approximation guarantees with respect to the edge satisfaction objective i.e., that of finding an assignment that satisfies the maximum fraction of constraints. On the other hand, our results in the current work show that the same instances when interpreted as STRONG-CSP's are easy (i.e, with respect to the vertex deletion objective, see Definition 1). Therefore, it is not immediately obvious if the conditions under which MAX-CSPs are easy also translate to conditions under which STRONG-CSP's are easy, and vice versa. Consequently, the broader agenda of identifying clean characterizations under which there is a separation in the approximability of the two classes of problems might yield useful insights towards understanding the limitations of the approximation techniques for problems from either class.

**Connection to Fortification.** A final motivation for studying STRONG-CSP's in the above setting is that the problem of finding slightly smaller sub-instances with better “local” approximation guarantees is closely related to notion of *fortification*. Informally, a MAX-CSP instance is said to be fortified if every large sub-instance of the CSP has (relative) optimal value no larger than the global optimal. Fortification is widely studied in the context of parallel repetition [32, 9, 33], and in particular, recent works [33] show that fortified UNIQUE GAME instances with hypercontractive small set expansion profiles can be used to bypass bottlenecks towards establishing *strong parallel repetition* for UNIQUE GAME instances. Consequently, this reduces the task of establishing UGC to that of showing that a family of fortified Boolean CSPs on small-set-expanders are hard. Given that STRONG-CSP's can be re-interpreted as the task of deciding whether an instance is fortified (in the perfect completeness regime), and the tight connections between small-set-expansion and threshold rank (e.g., [3, 25, 28]), these considerations further motivate the study of STRONG-CSP's even in the simpler setting where the full underlying constraint graph has low threshold rank. Motivated by the above considerations, we study the STRONGUNIQUEGAMES and related problems in this setting:

► **Problem 3 (STRONGUNIQUEGAMES).** *Given an instance  $\mathcal{G}(V, E, [k], \{\Pi_{ij}\}_{(i,j) \in E})$  consisting of a set of vertices  $V$ , a set of edges  $E$ , alphabet  $[k]$ , and a bijection  $\pi_{ij} \subseteq [k] \times [k]$  for each  $(i, j) \in E$ , the goal of this problem is to compute the largest  $S \subseteq V$  such that the instance induced on  $S$  has value 1.*

The STRONGUNIQUEGAMES problem is a natural variant of UNIQUEGAMES, it and its variants express several well studied problems such as ODDCYCLETRANSVERSAL, among others. There have been extensive work on the above problems, see Section 1.2 for a detailed review. Our main results in this paper are improved approximation algorithms for these problems in the setting where the induced graph on the good vertices has low threshold rank.

## 1.1 Our Results

Our main result is a new approximation algorithm for the STRONGUNIQUEGAMES problem where the induced sub-graph on the satisfiable set has low threshold rank. In order to make the theorem statements concise, we will define the notion of a subset being  $\lambda^*$ -good.

► **Definition 4** ( $\lambda^*$ -good). *Given a CSP constraint graph  $G = (V, E)$ , a subset  $V^* \subseteq V$  is said to be  $\lambda^*$ -good if the following conditions hold.*

1.  $\text{rank}_{\geq 1-\lambda^*}(\mathcal{G}[V^*]) \leq (1/\lambda^*)^{10}$ .<sup>2</sup>
2.  $G[V^*]$  is regular.

The above is a quantitative characterization of induced low-rank instances studied in this paper – all of our results are based on the above setting. Our first result is for STRONGUNIQUEGAMES instances with small vertex induced low threshold rank, as stated in the following theorem.

► **Theorem 5.** *Let  $\delta, \lambda^* \in (0, 1)$  be such that  $\delta \leq (\lambda^*)^{100}$ . Let  $\mathcal{G}(V, E, [k], \{\pi_e\}_{e \in E})$  be a STRONGUNIQUEGAMES instance such that there exists<sup>3</sup> a  $\lambda^*$ -good subset  $V_{\text{good}}$  of size at least  $(1 - \delta)n$  such that  $\text{Val}(\mathcal{G}[V_{\text{good}}]) = 1$ . Then there exists a randomized algorithm that runs in time  $n^{\text{poly}(k/\delta)}$  and outputs a subset  $\tilde{V} \subseteq V$  of size at least  $(1 - \delta^{1/12})n$  and a partial labeling  $\sigma : \tilde{V} \rightarrow [k]$  such that  $\sigma$  satisfies all induced constraints in  $\mathcal{G}[\tilde{V}]$ .*

The above theorem illustrates the tractability of the STRONGUNIQUEGAMES problem in the setting where just the instance induced on the satisfiable set has low threshold rank. To put the above result in perspective, [18] showed that given a STRONGUNIQUEGAMES instance with value  $(1 - \delta)$ , it is Unique Games hard to output a satisfiable subset of relative size  $(1 - \Omega(\sqrt{\delta \log d \log k}))$ , where  $d$  is the maximum degree of the graph and  $k$  is the label set size. We remark that the exponent in the fraction of vertices deleted (i.e.,  $\delta^{1/12}$ ) might be improvable and we have not made further attempts towards optimizing it. Theorem 5 almost directly leads to quantitatively similar results for the ODDCYCLETRANSVERSAL and BALANCEDVERTEXSEPARATOR problems, stated as corollaries.

► **Corollary 6.** *Let  $\delta, \lambda^* \in (0, 1)$  be such that  $\delta \leq (\lambda^*)^{100}$ . Let  $\mathcal{G} = (V, E)$  be a graph for which there exists a  $\lambda^*$ -good subset  $V_{\text{good}} \subseteq V$  of size at least  $(1 - \delta)n$  such that  $G[V_{\text{good}}]$  is bipartite. Then there exists an algorithm which runs in times  $n^{\text{poly}(1/\delta)}$  which outputs a set  $V' \subseteq V$  of size at least  $(1 - \delta^{1/12})n$  such that  $G[V']$  is bipartite.*

► **Theorem 7.** *Let  $\delta, \lambda^* \in (0, 1)$  be such that  $\delta \leq (\lambda^*)^{100}$ . Let  $\mathcal{G} = (V, E)$  be a graph for which there exists a  $\lambda^*$ -good subset  $V_{\text{good}} \subseteq V$  of size at least  $(1 - \delta)n$  such that the following holds. There exists a partition  $V_{\text{good}} = A \uplus B$  such that  $E_{G[V_{\text{good}}]}(A, B) = \emptyset$  i.e.,  $A$  is disconnected from  $B$  in  $G[V_{\text{good}}]$ . Then there exists a randomized algorithm which runs in time  $n^{\text{poly}(1/\delta)}$  and outputs a set  $S$  of size at most  $O(\delta^{1/12}n)$  and a partition  $A', B'$  of  $V \setminus S$  such that (a)  $E_{G[V \setminus S]}(A, B) = \emptyset$  and (b)  $(\gamma - \delta^{1/12})n \leq \min(|A'|, |B'|) \leq (\gamma + \delta^{1/12})n$  where  $\gamma = \min(|A|, |B|)/n$ .*

<sup>2</sup> The constant 10 in the exponent is arbitrary, and can be chosen to any large constant  $C$ , at the cost of loss in  $\text{poly}(C)$ -multiplicative factors in the fraction of vertices deleted by the algorithm. We instantiate it to be 10 for ease of notation.

<sup>3</sup> We do not assume that such a set is unique, we just need the existence at least one such subset.



For both `ODDCYCLETRANSVERSAL` as well as `BALANCEDVERTEXSEPARATOR`, the best known approximation algorithm for general instances have an approximation guarantee of  $O(\sqrt{\log |V|})$  [13, 1]. Furthermore, [18] showed that given a  $(1 - \delta)$ -satisfiable instance of `ODDCYCLETRANSVERSAL`, assuming UGC, it is NP-Hard to find set of size  $(1 - \Omega(\sqrt{\delta \log d}))$  which induces a bipartite graph. *It is important to note that our results hold for more restrictive setting where we assume the low threshold rank guarantee on the good set.* In particular, the technical core of our results is a spectral decomposition theorem which can be used to find a large subset that induces a sub-graph with relatively small threshold rank. We state an informal version of it here for reference.

► **Theorem 8** (Informal version of Theorem 4.2 [19]). *The following holds for every  $0 < \delta \leq 0.1$ . Let  $G = (V, E)$  be a  $d$ -regular graph on  $n$ -vertices such that there exists a set  $V_{\text{good}} \subseteq V$  of size at least  $(1 - \delta)n$  such that  $\text{rank}_{\geq 1 - \delta^{0.1}}(G[V_{\text{good}}]) \leq K$ . Furthermore, suppose  $K \leq 1/\delta^{100}$ . Then there exists an efficient algorithm outputs a set  $V'' \subseteq V$  of size at least  $(1 - O(\delta^{1/10}))n$  such that  $\text{rank}_{\geq 1 - \delta^{0.1}}(G[V'']) \leq \text{poly}(1/\delta)$ . Moreover, the subset  $V''$  itself is a disjoint union of constant number of  $\Omega(n)$ -sized subsets, each of which induces an expander.*

The above decomposition result adds to the already extensive literature on spectral decomposition – however, the above decomposition result is incomparable in terms of its setting and guarantees to the ones existing in the literature. For comparison, we describe the two previous such results which are closest to this work in terms of the setting and the guarantees:

- In [3], Arora, Barak and Steurer show that any  $n$ -vertex graph can be decomposed into non-expanding subsets which induce sub-graphs of  $(1 - \varepsilon^5)$ -threshold rank at most  $n^\varepsilon$ . While their result does not require the graph to contain a large low threshold rank sub-graph, their decomposition result can only guarantee a substantially weaker threshold rank bound of  $n^\varepsilon$  (as opposed to the constant bounds guaranteed in Theorem 8). We clarify that their  $n$ -dependent bound on the threshold rank is indeed unavoidable, since they make no assumptions on the threshold rank structure of the graph [34].
- In [15], Oveis Gharan and Rezaei show that given a regular graph which contains a  $\kappa n$ -sized spectral expander, one can efficiently find subset of size at least  $3\kappa n/8$  with spectral gap multiplicatively comparable to that of the optimal induced expander. Again, their result is not directly comparable to ours since even in graphs which contain a  $(1 - \delta)n$ -sized induced expander, their algorithm is only guaranteed to output a  $3(1 - \delta)n/8$ -sized subset which induces an expander. In comparison, for similar instances, Theorem 8 guarantees a  $(1 - \delta^{0.1})$ -sized subset which induces a “low threshold rank graph” – which itself is guaranteed to be a union of linear sized expanders. On the other hand, our result only applies in the setting  $\kappa \rightarrow 1$ , whereas their result holds for any constant  $\kappa \in (0, 1)$ .

We point out that our actual spectral decomposition theorem (see Theorem 4.2 of [19]) differs from the informal version stated above (i.e., Theorem 8) in a couple of crucial ways. Firstly, we only assume that only the underlying good graph  $G[V_{\text{good}}]$  is regular (as opposed to the full graph being regular) and make no assumptions on the degree distribution of the set of outlier vertices  $V \setminus V_{\text{good}}$  – indeed, these assumptions allow us to include instances which show a separation between the approximability of the MAX-CSP and STRONG-CSP objectives. Secondly, our actual guarantee is slightly more robust in the following sense: given any  $(1 - \delta^{O(1)})$ -sized subset  $V' \subseteq V$  (where  $V' \not\subseteq V_{\text{good}}$ ), one can find another subset  $V'' \subseteq V'$  of size  $(1 - \delta^{O(1)})$  such that  $\text{rank}_{1 - \delta^{O(1)}}(G[V'']) \leq \text{poly}(1/\delta)$ . The structural fact that we can still recover a large low threshold rank subgraph within any large subset  $V'$  is interesting on its own, we are not aware of similar results in the previous literature on spectral decomposition.

► **Remark 9** (On the regularity assumption). We point out that our threshold rank decomposition result, and more generally the approximation guarantees from Theorem 5 and its corollaries also hold as long as  $V_{\text{good}}$  is  $\lambda^*$ -good (Definition 4) and is contained in any subset  $\tilde{V}$  (where  $\tilde{V}$  may strictly contain  $V_{\text{good}}$ ) for which  $G[\tilde{V}]$  induces a regular subgraph – this naturally subsumes the more commonly studied setting where the full graph has low threshold rank and is regular [3, 8]. As in these works, our results will also hold for the setting where the graph is non-regular; in that setting, the guarantees of the threshold decomposition result and our algorithm will involve bounds on the volume of the subset deleted by the algorithm (as opposed to bounds on the size of the subset).

### Hardness of STRONG-CSPs

Given our algorithmic results hold for structured instances i.e., the subgraph induced by the good set has low threshold rank, an immediate question is if it is possible to obtain quantitatively similar approximation guarantees without making any assumptions. Towards that, our first observation is that arbitrary Strong 2-CSPs can be almost polynomially hard to approximate, as stated by the following fact.

► **Observation 10** (Hardness of General Strong 2-CSPs). *The following holds for any small  $\varepsilon > 0$ . Given a 2-CSP  $\Psi(V, E, \{\psi\}_{e \in E})$  over label set  $\{0, 1\}$ , it is NP-Hard to find a subset  $V' \subseteq V$  of size  $|V'| \geq n^{1-\varepsilon}|V^*|$  such that all induced constraints on  $V'$  are satisfiable. Here  $V^*$  is a set of largest cardinality for which there exists a labeling which satisfies all the induced constraints on  $V^*$ .*

The fact follows simply by using the observation that the Maximum Independent Set problem can be modeled as STRONG-CSP on label set  $\{0, 1\}$  with arity 2 (see Appendix B of the full version [19] for a formal explanation). On the other hand, it is known that all general 2-CSPs admit constant factor approximation (when the label set size is a constant). For e.g., for any 2-CSP on  $\{0, 1\}$  just a random assignment itself satisfies at least  $1/4$ -fraction of constraints in expectation. This shows that STRONG-CSP's can be strictly harder than MAX-CSPs. Clearly, one can expect general STRONG-CSP's to only get harder for larger arities, so we choose to relax the requirements of STRONG-CSP's and ask the following question. Consider a MAX-CSP which is known to be hard to approximate to a factor of  $\alpha$ . Then it is natural to ask, if given such an instance, can we delete a few vertices, and then output a labeling on the remaining instance which has approximation factor strictly better than  $\alpha$ . The following theorem answers the question in the negative for the specific setting where the CSP is Max-4-Lin.

► **Theorem 11.** *The following holds for any constants  $\alpha, \eta, \nu \in (0, 1)$ . Given a system of equations  $\Psi$  of arity 4, on variables  $X_1, X_2, \dots, X_n$  taking values in  $\mathbb{F}_2$ , it is NP-Hard to distinguish between the following cases:*

- *There exists an assignment to the variables which satisfies at least  $(1 - \eta)$ -fraction of constraints in  $\Psi$ .*
- *No subset  $S \subseteq V$  of size at least  $\alpha n$  induces a system of equations for which there exists an assignment which satisfies at least  $(1/2 + \nu)$  fraction of the induced constraints.*

The above can be thought of as an instance of approximation resistance in a STRONG-CSP sense; it is a strengthening of  $(1/2 + \nu)$ -inapproximability for Max-3-Lin shown by Håstad in the seminal work [23]. We prove the above hardness result by combining the techniques from [23] with novel application of expansion properties of the inner and outer verifiers. In particular, Theorem 11 says that one cannot hope to do slightly better than its inapproximability factor (which is matched by the naive random guessing algorithm) on any smaller sub-instance for approximation resistant predicates.

## 1.2 Related Work

### Strong Unique Games

Ghoshal and Louis [18] gave an algorithm that takes as input an instance of STRONGUNIQUEGAMES having a satisfiable set of size  $(1 - \varepsilon)n$ , and outputs a satisfiable subset of size at least  $(1 - \tilde{O}(k^2) \varepsilon \sqrt{\log n}) n$ . In a similar setting, they also gave another algorithm which outputs satisfiable subsets of size  $(1 - \tilde{O}(k^2) \sqrt{\varepsilon \log d}) n$ , where  $d$  is the largest vertex degree of the instance. Complementing these upper bounds, they also showed that it is Unique Games hard (in certain regimes of parameters) to compute a set of size larger than  $1 - O(\sqrt{\varepsilon \log d \log k})$  such that the induced instance on this set is satisfiable. These results were obtained via a connection between STRONGUNIQUEGAMES and small-set vertex expansion in graphs, and used the machinery (hypergraph orthogonal separators) developed in the context of approximation algorithms for small-set vertex expansion in graphs and hypergraph small-set expansion [27] in obtaining their approximation algorithms.

### General CSPs

There have been several works which give approximation algorithms for 2-CSPs. [5] were the first to study UNIQUEGAMES in the setting where the underlying constraint graph is an expander; they gave an algorithm with the approximation factor depending on only the second largest eigenvalue of the normalized Laplacian matrix of the instance. Subsequent works by Barak, Raghavendra and Steurer [8] and Guruswami and Sinop [21] extended this framework to general 2-CSPs when the underlying constraint graph and the label extended graph have low threshold rank respectively, with the algorithms running time exponential in threshold rank. On the other hand, Kolla [24] gave spectral approximation algorithms for UNIQUEGAMES and SMALLSETEDGEEXPANSION. Building on this, Arora, Barak and Steurer [3] gave sub-exponential time algorithms for UNIQUEGAMES and SMALLSETEDGEEXPANSION. In a recent work, [6] give efficient algorithms for UNIQUEGAMES based on the Sum Of Squares (SoS) hierarchy, when the underlying constraint graph is an SoS certifiable small set expander.

### Graph Partitioning and CSPs with Cardinality Constraints

Graph partitioning with vertex/edge expansion objectives has been extensively studied under the lens of approximation algorithms. Feige, Lee and Hajhiyaghayi [13] and Louis, Raghavendra and Vempala [29] give approximation algorithms for finding small size balanced vertex separators and minimizing vertex expansion respectively. Guruswami and Sinop [21, 22] gave improved approximation algorithms for several graph partitioning problems dealing with edge expansion for low threshold rank instances. [30] studied a planted model of instances where the graph induced on either side of the planted cut satisfies a lower bound requirement on its spectral gap in addition to satisfying some other properties; they gave exact and constant factor bi-criteria approximation algorithms for balanced vertex expansion for various ranges of parameters. They also gave a constant factor bi-criteria approximation algorithm for balanced vertex expansion for instances where one side of the optimal cut has a subgraph on  $\Omega(n)$  vertices satisfying a lower bound requirement on its spectral gap. [31] gave some similar results for  $k$ -way edge expansion and  $k$ -way vertex expansion.

The problem of decomposing a graph into expanders is also a well studied problem and has several applications to approximation algorithms. In [38], Trevisan gave a decomposition of a graph into non-expanding set which induce expanders. There have been several subsequent works [3, 16, 17] which deal with the problem of partitioning a graph into expanding/low

threshold rank graphs. Oveis Gharan and Rezeai [15] study the problem of finding a large subset of vertices such that the graph induced on them is an expander; we discuss this more in Section 2.

## 2 Overview and Techniques

We begin by reviewing the by now standard *Propagation Rounding* based framework which was introduced informally in [5] and then later developed in [8, 21]. For simplicity, we shall restrict our discussion to the setting of **UNIQUEGAMES**. Consider the following convex program which is the  $R$ -level Sum-of-Squares (SoS) lifting of SDP relaxation for **UNIQUEGAMES**:

$$\min_{\substack{\mu \text{ is a degree-}R \\ \text{pseudo-distribution}^4}} \mathbb{E}_{(i,j)=e \sim E} \Pr_{(X_i, X_j) \sim \mu} [X_i \neq \pi_{j \rightarrow i}(X_j)]. \quad (1)$$

The above convex program is intended to minimize the number of unsatisfied edges by the (pseudo)-distribution. The algorithm proceeds along the following steps.

1. Solve the  $R$ -round Lasserre relaxation for the SDP where  $R$  is chosen large enough as a function of the error to be tolerated, and the threshold-rank of the instance. Let  $\mu := \{\mu_{S,\alpha}\}$  be the degree- $R$  pseudo-distribution corresponding to the optimal value of the relaxation.
2. Choose a subset  $S$  appropriately, sample an assignment  $x_S$  to the variables in  $S$  from the local distribution  $\mu_S$ .
3. Label the remaining vertices  $i \in V \setminus S$  by sampling from their respective conditional distributions  $\mu_{i|x_S}$  independently.

The main idea used in the aforementioned works for relating the expected value of the rounded solution to the SDP objective is the so called *local-to-global* correlation property [8, 21], which has the following key consequence. If the underlying constraint graph has constant threshold rank, then conditioning on constant levels of the SoS solution should result in pseudo-distributions that have small average local correlation i.e.,

$$\mathbb{E}_{(i,j) \sim E} [\text{Corr}_{\mu|x_S}(X_i, X_j)] \leq o(1).$$

Consequently, independent sampling from the marginals of conditional pseudo-distribution  $\mu|x_S$  will result in labelings that which have value close to optimal of the lifted SDP. While this recipe and its variants has been remarkably successful in dealing with **MAX-CSPs** [8, 21, 6], it is easy to see that this framework does not translate well to the framework of **STRONG-CSP**'s studied in this paper, as we briefly describe below.

Firstly, note that in the setting of **STRONG-CSP**'s, the emphasis is on deleting vertices to ensure that all surviving constraints are simultaneously satisfiable. This is in direct contrast to the aforementioned results where the algorithms are allowed to output labelings which satisfy “almost all”, but not necessarily, “all”, constraints. A naive approach towards extending the above to our setting would be to first find a good labeling that satisfies almost all edges, and then delete the vertices corresponding to the violated edges. However, doing so might result in approximation guarantees that are worse by a factor of the max-degree. Furthermore, this

<sup>4</sup> Informally, a degree- $R$  pseudo-distribution is a collection of local distributions  $\{\mu_S\}_S$  for every subset  $S \subseteq V$  of size at most  $R$ , which are pairwise consistent up to all variables sets of size at most  $R$  (see Section 3.2 of the full version [19] for more details).

approach can fail badly in instances where the induced sub-instance on the good vertices  $\mathcal{G}[V_{\text{good}}]$  is sparse (i.e, constant degree), but the full graph is relatively dense and almost non-satisfiable, since these algorithms are designed to compete against the global optimum for the edge-satisfaction version of the problem. A final hurdle is that the local-to-global correlation guarantee, which was the key property used to guarantee the goodness of the rounding algorithm, might not hold for the full constraint graph of  $\mathcal{G}$  since in our setting, the constant threshold-rank guarantee may only hold for the constraint graph induced on  $V_{\text{good}}$ . In fact, the threshold rank of the full graph can be as large as  $\Omega(|V \setminus V_{\text{good}}|)$  which implies that conditioning on constant levels of the SoS solution might result in pseudo-distributions that don't guarantee any local-to-global correlation like property. These issues taken together guide our approach to the design of our algorithm (described informally in Figure 1); we describe and motivate the various steps of the algorithm details in the remainder of this section.

**Input:** A UNIQUEGAMES instance  $\mathcal{G}(V_{\mathcal{G}}, E_{\mathcal{G}}, [k], \{\pi_e\}_{e \in E})$  satisfying the conditions of Theorem 5.

**Algorithm:**

- ▶ **Threshold Rank Decomposition.** As a first step, we compute a  $(1 - O(\delta^c))$ -sized subset  $V''$  of  $V_{\mathcal{G}}$  with low threshold-rank and bounded degree using our spectral decomposition algorithm.
- ▶ **SDP with Slack Variables.** We solve the  $R$ -level SoS relaxation of a modified SDP for UNIQUEGAMES instance induced on  $V''$  with the extended label set  $[k] \cup \{*\}$ , where the label  $*$  is meant to indicate vertices which are to be deleted.
- ▶ **Low Variance Rounding.** We sample an assignment  $\alpha$  for an appropriately chosen subset  $S$ , and then label the vertex  $i \in V$  with the label with the largest probability in the conditional marginal  $\mu_i|_{X_S=\alpha}$ .

■ **Figure 1** StrongUG-Informal.

### Finding a large bounded-degree low threshold-rank graph

Since the full instance in our setting can have arbitrarily large threshold rank (due to the edges incident on the set of outlier vertices), a natural way to overcome this issue would be to zoom into a large (i.e,  $(1 - o_\delta(1))$ -sized) subset of vertices which induces a subgraph with (comparably) low threshold rank. We do this by using a new threshold rank based spectral decomposition algorithm with the following guarantee: given a graph  $G = (V, E)$  for which there exists a  $(1 - \delta)|V|$  sized subset that induces a regular low threshold rank sub-graph, the algorithm returns a  $(1 - \delta^{O(1)})$ -sized subset with threshold<sup>5</sup> rank at most  $\text{poly}(1/\delta)$ . This algorithm is the main technical contribution of this paper; in particular, it combines a classical approximation algorithm for the partial vertex cover problem and extensions of spectral partitioning primitives from Oveis Gharan and Rezaei [15] to the setting of low threshold rank graphs. We defer a more detailed discussion of this step to Section 2.1 for now and proceed with our discussion of the subsequent steps of the full algorithm.

<sup>5</sup> Here the threshold parameter is dependent on  $\delta$  and the optimal value of the STRONGUNIQUEGAMES instance.

### Solve SoS relaxation with Slack Variables

In the next step, we consider the SDP for UNIQUEGAMES modified with slack variables. Specifically, let  $\mathcal{G}(V, E, [k], \{\pi_e\}_{e \in E})$  be the UNIQUEGAMES instance. Due to the above step, we can directly assume that the full graph has low threshold-rank and bounded vertex degrees. Furthermore, since the previous step only removes a tiny fraction of vertices, we can assume that there exists a subset  $V_{\text{sat}} \subseteq V$  such that  $|V_{\text{sat}}| \geq (1 - 2\delta)|V|$  and  $\mathcal{G}[V_{\text{sat}}]$  is fully satisfiable<sup>6</sup>. Now given  $\mathcal{G}$ , we consider a partial<sup>7</sup> Unique Game  $\mathcal{G}'(V, E, [k] \cup \{*\}, \{\Pi_e\}_{e \in E})$  with the global constraint that the fraction of vertices that can be labeled  $*$  is at most  $2\delta$ . Here the label  $*$  is meant to indicate vertices that are supposed to be deleted. Consequently, for any edge  $e \in E$ , we define the extended constraint set  $\Pi_e = \pi_e \cup (\{*\} \times \Sigma) \cup (\Sigma \times \{*\})$ . Note that this constraint is no longer a “unique game” constraint. The final SoS relaxation used is almost identical to Eq. 1, along with the following modifications:

$C_1$ : The pseudo-distribution is now over assignments to variables from the extended label set  $[k] \cup \{*\}$ .

$C_2$ : We add the global cardinality constraint  $\Pr_{i \sim V} \Pr_{X_i \sim \mu}[X_i = *] \leq 2\delta$ .

$C_3$ : We also add the constraint

$$\Pr_{(X_i, X_j) \sim \mu} [\pi_{i \rightarrow j}(X_i) \neq X_j] \leq \Pr_{X_i \sim \mu} [X_i = *] + \Pr_{X_j \sim \mu} [X_j = *],$$

for every edge  $(i, j) \in E$

The cardinality constraint ( $C_2$ ) is intended to ensure that conditioned on any assignment that is assigned a non-zero probability mass by the SDP solution, the fraction of vertices that are labeled  $*$  under the resulting conditional distribution is at most  $\delta$ . The edge violation constraints ( $C_3$ ) are intended to ensure that an edge constraint is allowed to be violated only when one of the end points is labeled  $*$ . It is easy to verify that this SDP is feasible for  $\mathcal{G}'$ . Furthermore, since the previous step guarantees that the max-degree of the surviving graph is at most a constant times the average degree, this implies that the optimal value of the SoS relaxation is at most  $\delta^{O(1)}$ .

### Low Variance Rounding

In the final step, we have to round the SDP solution to output a large set with the corresponding labeling which satisfies all induced constraints. As mentioned above, the *local-to-global* correlation argument in itself is not sufficient for this purpose, as it can only guarantee that a labeling which violates a small fraction of edges. However, it is well known that for certain kinds of CSPs e.g., UNIQUEGAMES, 3-COLORING, the low threshold-rank guarantee implies the stronger property of “*conditioning reduces variance*” [8, 21, 4]<sup>8</sup>, which says that for an appropriately chosen subset  $S \subseteq V$  we have

$$\mathbb{E}_{X_S \sim \mu_S} \left[ \mathbb{E}_{i \sim V} \text{Var} [X_i | X_S] \right] \leq \frac{\text{SDP}}{\lambda_m}, \quad (2)$$

whenever  $\text{rank}_{\geq 1 - \lambda_m}(G) \leq m$ . Note that this is a strictly stronger property than local-to-global correlation (see Appendix D of [19] for an example which separates the two properties). To see why this property is useful in constructing labelings which satisfy all induced constraints,

<sup>6</sup> Note that  $V_{\text{sat}}$  may be a strict subset of  $V_{\text{good}}$  since the previous step may remove a few vertices from  $V_{\text{good}}$ .

<sup>7</sup> The nomenclature “partial” Unique Game was introduced in [37] and refers to a Unique Game with the additional property that a fixed fraction of vertices are allowed to be left as unlabeled.

<sup>8</sup> [4] actually showed a variant of this statement tailored towards finding large independent sets.



consider a constraint  $(i, j) \in E_G$  such that for some partial assignment  $X_S \leftarrow \alpha$ , the conditional marginals of vertices  $i$  and  $j$  have low variance i.e.,  $\text{Var}[X_i | X_S = \alpha] \leq 0.1$  and  $\text{Var}[X_j | X_S = \alpha] \leq 0.1$ . Therefore, it follows that there exists labels  $a, b \in [k] \cup \{*\}$  for which  $\mu_{i=a | X_S=\alpha} \geq 0.9$  and  $\mu_{j=b | X_S=\alpha} \geq 0.9$ . Furthermore, suppose we assume that  $a, b \in [k]$ . Then we claim that the SDP constraints imply that  $\pi_{i \rightarrow j}(a) = b$ . This is because for any  $(a', b') \in [k] \times [k]$  which violates the edges  $(i, j)$ , using the edges violation constraints ( $C_3$ ) and a union bound we get that

$$\Pr_{(X_i, X_j) \sim \mu | X_S=\alpha} [X_i = a', X_j = b'] \leq 0.2.$$

On the other hand, our choice of labels  $a$  and  $b$  for vertices  $i$  and  $j$  (respectively) imply that

$$\Pr_{(X_i, X_j) \sim \mu | X_S=\alpha} [X_i = a, X_j = b] \geq 1 - \Pr_{X_i \sim \mu | X_S=\alpha} [X_i \neq a] - \Pr_{X_j \sim \mu | X_S=\alpha} [X_j \neq b] \geq 0.8.$$

Therefore, it must be that  $\pi_{j \rightarrow i}(a) = b$  i.e., the labeling  $(a, b)$  satisfies the edge  $(i, j)$ . In summary, low variance vertices whose leading labels are not  $'*'$  induce a satisfiable instance. We point out that a similar observation was also made by Arora and Ge [4] who used it to find large independent sets in low threshold-rank graphs. The above discussion naturally suggests the following rounding process:

1. Let  $S$  be the subset for which Eq. 2 holds. Sample an assignment  $\alpha \sim \mu_S$  for  $X_S$
2. Delete the vertices for which  $\text{Var}_{\mu | X_S=\alpha}[X_i] > 0.1$ .
3. For the remaining vertices  $i \in V$ , assign the maximum likelihood labeling

$$\sigma(i) = \underset{a \in [k] \cup \{*\}}{\text{argmax}} \Pr_{X_i \sim \mu | X_S=\alpha} [X_i = a].$$

4. Delete the vertices labeled as  $*$  and output the surviving vertices with the corresponding labeling.

The above discussion ensures that the set output by the rounding scheme is satisfiable. Combining (2) with the SDP bound and the threshold-rank bound established in the previous steps imply that  $O(\delta^{O(1)})$  vertices get deleted in step 3. Furthermore, the global cardinality constraint ensures that the fraction of vertices labeled  $'*'$  (and hence deleted in the round step) is  $O(\delta)$ . This with the bound on the vertices deleted in the previous steps imply that the total fraction of vertices deleted is  $\delta^{O(1)}$ , which concludes the analysis of the algorithm.

## 2.1 Threshold Rank based Spectral Partitioning

As mentioned above, the first step of our algorithm (i.e., the threshold rank decomposition step) is the key technical contribution of this work. Formally, our objective here is the following: given a graph  $G = (V, E)$  which contains a  $(1 - \delta)$ -sized subset  $V_{\text{good}}$  that induces a regular subgraph with low threshold rank, the objective is to recover a  $(1 - o_\delta(1))$ -sized subset that has relatively small threshold rank (say  $\text{poly}(1/\delta)$ ). This in itself is a well motivated question and various versions of it have been studied in the design of approximation algorithms for UNIQUEGAMES and SMALLSETEDGEEXPANSION (see [3] and references therein). However, we point out that the techniques from these earlier works do not immediately apply to our setting as in these works, the emphasis there is rather on finding sub-linear sized sets which induce graphs with threshold rank growing with the number of vertices (with of course, no assumption on the spectrum of the full graph). In our setting, we instead want to design algorithms that exploit the “almost low threshold” structure of the instance and output



sets that satisfy the stronger guarantees of being  $\approx (1 - o(1))$ -sized and having constant threshold rank. In the remainder of this section, we motivate and describe the design of such an algorithm.

**Finding a Linear Sized Low Rank Set.** To begin with, let us first consider the simpler setting where we assume that the max-degree of the graph is at most a constant times the degree of the induced good graph  $G[V_{\text{good}}]$  (we shall later discuss how to achieve this condition at the cost of deleting a few additional vertices). Furthermore, let us first address the even simpler goal of finding a linear (say  $n/1000$ ) sized subset which induces a low threshold rank graph. Again, this in itself is a well motivated problem, and several previous works [38, 15] study the related question when the induced subgraph has to be an expander<sup>9</sup>. Most of these works build on the following basic spectral partitioning primitive that also forms the basis of our algorithm:

$$\triangleright \quad \begin{aligned} &\text{There exists an efficient algorithm that given a graph } G = (V, E), \\ &\text{outputs a partition } \mathcal{P} := (S, T) \text{ of } V \text{ such that either } |S| \geq 3n/4 \\ &\text{and } G[S] \text{ is an expander, or } (S, T) \text{ is balanced}^{10} \text{ and has small expansion.} \end{aligned} \quad (3)$$

The above algorithm is a simple recursive application of the spectral partitioning algorithm from Cheeger's inequality (see Lemmas 3.1, 5.1 from the full version [19] for more details). Note that the above algorithm may either output a large set which induces an expander (in which case we are done), or a balanced partition (say  $\mathcal{P}_0$ ) with small expansion. How do we proceed if the latter is the case? Following an idea from [15], we again apply the spectral partitioning (i.e., (3)) to each set in the partition in partition  $\mathcal{P}_0$  to construct a refinement of the partition, say  $\mathcal{P}_1$ . Again, if  $\mathcal{P}_1$  contains a linear sized subset which induces an expander, then we are done – otherwise, we again keep repeating the above process. We iteratively continue constructing a sequence of refinements  $\mathcal{P}_0 \subseteq \mathcal{P}_1 \subseteq \dots \subseteq \mathcal{P}_t$  until one of the partitions contains a linear sized set which induces an expander. But then one can ask that how can we guarantee that the process terminates? This is where the *higher order Cheeger's inequality* [28, 25] comes to the rescue i.e., we show that if the process continues beyond some iteration  $t = t(\delta)$ , then  $\mathcal{P}_t$  is a balanced  $K := 2^{t-1}$ - partition of the vertex set. In particular, using the higher order Cheeger inequality and the fact that  $G[V_{\text{good}}]$  as low threshold rank, we can show that at least one of the  $K$ -sets in the partition must have large edge boundary i.e.,

$$\max_{i \in [K]} |\partial_G(S_i)| \geq \Omega(\varepsilon dn) \quad (\text{by an appropriate instantiation of parameters for (3).}) \quad (4)$$

On the other hand, note that since the algorithm proceeds beyond iteration  $t$ , it follows that for each application of (3) in each of the  $t$  iterations, the spectral partitioning algorithm returns a non-expanding partition (using the “or” guarantee from (3)), and hence the fraction of edges crossing the various sets in the partition  $\mathcal{P}_t$  must be small i.e.,

$$\sum_{i \in [K]} |\partial_G(S_i)| \leq O(\varepsilon^2 dn), \quad (5)$$

<sup>9</sup> Here we refer to any graph whose spectral gap is at least a constant as an expander.

<sup>10</sup> Here we say a partition  $V = S \sqcup T$  is *balanced* if  $|S|, |T| \in [|V|/4, 3|V|/4]$ .

which contradicts the upper bound on the expansion from (4). In summary, the above arguments taken together imply that the above process must terminate during some iteration  $t' \leq t$ , resulting in a subset of size at least  $2^{-t'} \cdot n = \tilde{\Omega}(n)^{11}$  which induces an expander.

**Finding Many Low Rank Sets.** Now that we have an algorithm that finds a  $\Omega(n)$ -sized set (say  $S_1$ ) which induces an expander, the next step is to find many such vertex disjoint sets in the graph. This is easily achieved by deleting the first such subset  $S_1$  recovered by the above algorithm, and then again running the above algorithm on the graph  $G[V \setminus S_1]$  to recover a linear sized subset  $S_2 \subseteq V \setminus S_1$  which again induces an expander in  $G$ . However, note that the induced sub-graph  $G[V \setminus S_1]$  does not automatically inherit the structural properties of  $G[V_{\text{good}}]$  and hence, additional care is need to ensure that the above algorithm will still succeed on the smaller induced sub-graph  $G[V \setminus S_1]$ . In particular, we shall again need to establish an analogue of (4) where we show that any balanced  $K$ -way partition  $\mathcal{P}$  of the smaller set  $V \setminus S_1$  will still have one expanding set. This is done by showing that any balanced  $K$ -way partition of  $V \setminus S_1$  can be carefully extended to a balanced  $K$ -way partition  $\mathcal{P}'$  of  $V$  such that

$$\max_{S \in \mathcal{P}} |\partial_{G[V \setminus S_1]}(S)| \gtrsim \max_{S' \in \mathcal{P}'} |\partial_{G[V]}(S')|, \quad (6)$$

Note that the above immediately implies the desired  $K$ -way expansion bound for  $\mathcal{P}$  since the latter term can again be lower bounded by combining the higher order Cheeger's inequality with the threshold rank guarantee of the full graph  $G[V]$ . We point out that establishing (6) is precisely where the bounded degree assumption on the graph comes in handy. The above (i.e., (6)), along with an appropriately tailored version of (5) will allow us to establish that the algorithm will again find a linear sized subset  $S_2 \subset V_1 \setminus S_1$  which induces an expander. Overall, we keep iteratively finding and removing linear sized subsets  $S_2, S_3, \dots$ , – each of which induces an expander – until only  $o_\delta(1)$ -vertices remain; this results in an almost<sup>12</sup> partition  $\mathcal{P} := \{S_i\}_{i \in [N]}$  of the vertex set where each of the subsets in partition has small edge boundary, is linear sized, and induces an expander in the full graph  $G$ .

**Stitching the sets together.** Recall that our final objective is not to find an almost partition consisting of induced low threshold rank subgraphs, but to find one large  $(1 - o_\delta(1))$ -subset  $V'$  that induces a low threshold rank subgraph. To that end, we just show that the set  $V' := \cup_{i \in [N]} S_i$  is itself such a set. To see this, observe that the adjacency matrix  $A[V']$  of induced subgraph  $G[V']$  is almost block diagonal (since the above step guarantees that only few edges cross the partition  $\{S_i\}_{i \in [N]}$ ). Hence with some additional work we can conclude that the number of large eigenvalues in  $A[V']$  must be at most the sum of number of large eigenvalues in each of the blocks  $A[S_1], \dots, A[S_N]$ , each of which is again small on account of  $G[S_i]$ 's being expanders i.e., we can conclude  $\text{rank}_{1-\delta^{O(1)}}(G[V']) \leq O(N)$ . Furthermore, since each of the sets in the partition  $\mathcal{P}$  is linear sized, this establishes that  $N$  is at most a constant (possibly depending on  $\delta$ ), which implies that the threshold rank of  $G[V']$  is at most  $O_\delta(1)$ .

**Reducing to the Bounded Degree Setting.** Lastly, we address the issue that in general the max degree of the underlying constraint graph can be arbitrarily large compared to the degree  $d$  of the underlying good graph  $G[V_{\text{good}}]$ . Towards addressing this, we introduce

<sup>11</sup> Here  $\tilde{\Omega}$  hides poly-logarithmic in  $\delta$  factors.

<sup>12</sup> An almost partition of a set  $[n]$  is a collection of disjoint sets whose union contain  $(1 - o(1))$ -fraction of the elements.

an additional pre-processing step which reduces the average degree of the remaining graph to  $O(d)$  by deleting a small number of vertices, and then additionally deletes the vertices in the remaining subgraph which have degree larger than  $d/\delta^{O(1)}$ . For the first part, we use a 2-factor approximation algorithm for the Partial Vertex Cover problem [10] that can be used to identify a small number of vertices that hits  $\approx (d_{\text{avg}}(G) - d)n/2$  edges (where  $d_{\text{avg}}$  denotes the average degree). The subsequent deletion step again just removes a small number of vertices; this follows from a simple application of Markov's inequality. Finally, we remark that this again perturbs the spectral structure of the graph used that is used in the subsequent steps, and hence additional care is needed to make all of the above arguments go through.

---

## References

- 1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev.  $O(\sqrt{\log n})$  approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 573–581, 2005.
- 2 Vedat Levi Alev and Lap Chi Lau. Approximating unique games using low diameter graph decomposition. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2017.
- 3 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *Journal of the ACM (JACM)*, 62(5):1–25, 2015.
- 4 Sanjeev Arora and Rong Ge. New tools for graph coloring. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 1–12. Springer, 2011.
- 5 Sanjeev Arora, Subhash A Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K Vishnoi. Unique games on expanding constraint graphs are easy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 21–28, 2008.
- 6 Mitali Bafna, Boaz Barak, Pravesh K Kothari, Tselil Schramm, and David Steurer. Playing unique games on certified small-set expanders. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1629–1642, 2021.
- 7 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 453–462. IEEE, 2009.
- 8 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *2011 IEEE 52nd annual symposium on foundations of computer science*, pages 472–481. IEEE, 2011.
- 9 Mohammad Bavarian, Thomas Vidick, and Henry Yuen. Parallel repetition via fortification: Analytic view and the quantum case. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 22:1–22:33, 2017.
- 10 Nader H Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 298–308. Springer, 1998.
- 11 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2006.
- 12 Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 687–696. IEEE, 2006.
- 13 Uriel Feige, MohammadTaghi Hajiaghayi, and James R Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

- 14 Jean Fonlupt, Ali Ridha Mahjoub, and JP Uhry. Compositions in the bipartite subgraph polytope. *Discrete mathematics*, 105(1-3):73–91, 1992.
- 15 Shayan Oveis Gharan and Alireza Rezaei. Approximation algorithms for finding maximum induced expanders. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1158–1169. SIAM, 2017.
- 16 Shayan Oveis Gharan and Luca Trevisan. A new regularity lemma and faster approximation algorithms for low threshold rank graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2013.
- 17 Shayan Oveis Gharan and Luca Trevisan. Partitioning into expanders. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1256–1266. SIAM, 2014.
- 18 Suprovat Ghoshal and Anand Louis. Approximation algorithms and hardness for strong unique games. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 414–433. SIAM, 2021.
- 19 Suprovat Ghoshal and Anand Louis. Approximating csps with outliers. *CoRR*, abs/2205.11328, 2022. doi:10.48550/arXiv.2205.11328.
- 20 Michel X. Goemans and David P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2sat. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 422–431. ACM, 1994. doi:10.1145/195058.195216.
- 21 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 482–491. IEEE Computer Society, 2011.
- 22 Venkatesan Guruswami and Ali Kemal Sinop. Approximating non-uniform sparsest cut via generalized spectra. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 295–305. SIAM, 2013.
- 23 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- 24 Alexandra Kolla. Spectral algorithms for unique games. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 122–130. IEEE Computer Society, 2010.
- 25 James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):1–30, 2014.
- 26 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 777–789. SIAM, 2011.
- 27 Anand Louis and Yury Makarychev. Approximation algorithms for hypergraph small-set expansion and small-set vertex expansion. *Theory of Computing*, 12(1):1–25, 2016.
- 28 Anand Louis, Prasad Raghavendra, Prasad Tetali, and Santosh Vempala. Many sparse cuts via higher eigenvalues. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1131–1140, 2012.
- 29 Anand Louis, Prasad Raghavendra, and Santosh Vempala. The complexity of approximating vertex expansion. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 360–369. IEEE, 2013.

- 30 Anand Louis and Rakesh Venkat. Semi-random Graphs with Planted Sparse Vertex Cuts: Algorithms for Exact and Approximate Recovery. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 101:1–101:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 31 Anand Louis and Rakesh Venkat. Planted Models for k-Way Edge and Vertex Expansion. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 32 Dana Moshkovitz. Parallel repetition from fortification. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 414–423. IEEE Computer Society, 2014.
- 33 Dana Moshkovitz. Strong parallel repetition for unique games on small set expanders. *arXiv preprint arXiv:2103.08743*, 2021.
- 34 Guy Moshkovitz and Asaf Shapira. Decomposing a graph into expanding subgraphs. *Random Structures & Algorithms*, 52(1):158–178, 2018.
- 35 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC*, pages 245–254, 2008. doi:10.1145/1374376.1374414.
- 36 Prasad Raghavendra and David Steurer. How to round any CSP. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 586–594, 2009. doi:10.1109/FOCS.2009.74.
- 37 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 755–764. ACM, 2010.
- 38 Luca Trevisan. Approximation algorithms for unique games. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 197–205. IEEE Computer Society, 2005.

# Submodular Dominance and Applications

Frederick Qiu ✉

Department of Computer Science, Princeton University, NJ, USA

Sahil Singla ✉

School of Computer Science, Georgia Tech, Atlanta, GA, USA

---

## Abstract

In submodular optimization we often deal with the expected value of a submodular function  $f$  on a distribution  $\mathcal{D}$  over sets of elements. In this work we study such submodular expectations for negatively dependent distributions. We introduce a natural notion of negative dependence, which we call *Weak Negative Regression* (WNR), that generalizes both Negative Association and Negative Regression. We observe that WNR distributions satisfy *Submodular Dominance*, whereby the expected value of  $f$  under  $\mathcal{D}$  is at least the expected value of  $f$  under a product distribution with the same element-marginals.

Next, we give several applications of Submodular Dominance to submodular optimization. In particular, we improve the best known submodular prophet inequalities, we develop new rounding techniques for polytopes of set systems that admit negatively dependent distributions, and we prove existence of contention resolution schemes for WNR distributions.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis; Mathematics of computing → Probabilistic algorithms; Theory of computation → Algorithmic game theory

**Keywords and phrases** Submodular Optimization, Negative Dependence, Negative Association, Weak Negative Regression, Submodular Dominance, Submodular Prophet Inequality

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.44

**Category** APPROX

## 1 Introduction

A function  $f : 2^U \rightarrow \mathbb{R}$  on universe  $U = \{1, \dots, n\}$  is *submodular* if it satisfies  $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$  for all  $S, T \subseteq U$ . These functions capture the concept of diminishing returns, and are therefore useful in many fields such as machine learning, operations research, mechanism design, and combinatorial optimization; see books [20, 3, 36, 31].

Although  $f$  is a discrete function, for many applications it is useful to define a continuous relaxation  $f_{\text{cont}} : [0, 1]^n \rightarrow \mathbb{R}$  of  $f$ , since that allows us to use techniques from continuous optimization. Here, by a relaxation we mean that  $f_{\text{cont}}$  equals  $f$  at the indicator vectors of the sets, i.e.,  $f_{\text{cont}}(\mathbf{1}_S) = f(S)$  for all  $S \subseteq U$ . A standard way to define such continuous relaxations is to first define a probability distribution  $\mathcal{D}(\mathbf{x})$  over subsets of  $U$  with element-marginals  $\mathbf{x} \in [0, 1]^n$ , and then define  $f_{\text{cont}}(\mathbf{x})$  to be the expectation with respect to this distribution, i.e.,  $f_{\text{cont}}(\mathbf{x}) := \mathbb{E}_{S \sim \mathcal{D}(\mathbf{x})}[f(S)]$ , where  $S$  is a random set drawn from  $\mathcal{D}(\mathbf{x})$ . For example, the popular *multilinear relaxation*  $F(\mathbf{x})$  is defined by taking  $\mathcal{D}(\mathbf{x})$  to be the product distribution with marginals  $\mathbf{x}$ . Other examples include the convex closure relaxation  $f^-(\mathbf{x})$  (which is equivalent to the Lovász extension for submodular functions), the concave closure relaxation  $f^+(\mathbf{x})$ , and the relaxation  $f^*(\mathbf{x})$  [38]. Studying the properties of submodular expectations for these distributions has been a fruitful direction, which has led us to several optimal/approximation algorithms for submodular optimization [3, 39, 7, 18, 2, 16].

Given the success of the above continuous relaxations, it is natural to ask what other continuous relaxations, or equivalently, what other submodular expectations and distributions  $\mathcal{D}(\mathbf{x})$  could be defined that are useful for new or improved applications. In this work, we study



© Frederick Qiu and Sahil Singla;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 44; pp. 44:1–44:21



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



submodular expectations for negatively dependent distributions and use them to improve the best known submodular prophet inequalities, to develop new rounding techniques, and to design contention resolution schemes for negatively dependent distributions.

## 1.1 Submodular Dominance

Since the multilinear extension  $F$  is commonly employed in combinatorial optimization, one avenue to explore other continuous relaxations is by comparing them to  $F$ .

► **Definition 1** (Submodular Dominance). *A distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x} \in [0, 1]^n$  satisfies Submodular Dominance if for every submodular function  $f : 2^U \rightarrow \mathbb{R}$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}}[f(S)] \geq F(\mathbf{x}) .$$

Shao [37] studied a similar concept that he called a comparison theorem, which involved a subclass of submodular functions. Christofides and Vaggelatou [13] later studied what they called the supermodular ordering, which is essentially equivalent to Submodular Dominance. Both viewed the problem through the lens of probability theory, whereas we approach it from the standpoint of combinatorial optimization.

It is not difficult to see how one might apply Submodular Dominance, e.g., it immediately yields an algorithm to round multilinear extension subject to feasibility constraints. However, Submodular Dominance implies a much wider variety of results in stochastic settings, where most of our current understanding relies on the independence of random variables. By relating product distributions to more complex distributions, Submodular Dominance allows us to improve existing results and study more general problems.

## 1.2 Negative Dependence and Submodular Dominance

Positive correlations can only decrease the expectations of submodular functions due to their diminishing marginal returns, so we turn our attention to negatively dependent distributions. Pemantle initiated a systematic study of such distributions in [32]. In this work, we introduce the following generalization of Negative Association (NA) and Negative Regression (NR), two popular notions of negative dependence (details in Section 2).

► **Definition 2** (WNR). *A distribution  $\mathcal{D}$  over  $2^U$  satisfies Weak Negative Regression (WNR) if for any  $i \in U$  and any monotone function  $f : 2^U \rightarrow \mathbb{R}$ ,<sup>1</sup>*

$$\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid i \in S] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid i \notin S] . \quad (1)$$

Equivalently,  $\mathcal{D}$  is WNR if  $S \setminus i$  conditioned on  $i \notin S$  stochastically dominates  $S \setminus i$  conditioned on  $i \in S$  for all  $i \in U$ . This captures an intuitive notion of negative dependence where conditioning on including an element lowers the probability of other inclusion events. WNR distributions satisfy Submodular Dominance as well as many desirable closure properties.

**Submodular Dominance for Negatively Dependent Distributions.** Christofides and Vaggelatou [13] proved that NA distributions over continuous random variables satisfy Submodular Dominance for a continuous generalization of submodular functions. We strengthen their result in Section 3 in the setting of Bernoulli random variables from NA to WNR distributions, a strict superset of the union of NA and NR distributions.

<sup>1</sup> A function  $f$  is monotone if it satisfies  $f(S) \leq f(T)$  for all  $S \subseteq T$ . Elements should be taken as singleton sets depending on context, e.g.,  $S \setminus i$  means  $S \setminus \{i\}$ .



► **Theorem 3.** *WNR distributions satisfy Submodular Dominance.*

It turns out that there exist distributions that satisfy Submodular Dominance but are not WNR. This raises the question: what conditions are necessary for Submodular Dominance? We first recall the classic notion of Negative Cylinder Dependence (see, e.g., [21]).

► **Definition 4 (NCD).** *A distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x}$  satisfies Negative Cylinder Dependence (NCD) if for any  $T \subseteq U$ ,<sup>2</sup>*

$$\Pr_{S \sim \mathcal{D}}[T \subseteq S] \leq \Pr_{S \sim \mathbf{x}}[T \subseteq S] \quad \text{and} \quad \Pr_{S \sim \mathcal{D}}[T \subseteq S^c] \leq \Pr_{S \sim \mathbf{x}}[T \subseteq S^c] .$$

NCD can be interpreted as saying that any subset of elements are negatively correlated.

► **Theorem 5.** *All distributions that satisfy Submodular Dominance are NCD.*

This can be useful when Submodular Dominance is an easier property to prove. For example, the distribution arising from randomized swap rounding can be shown to satisfy Submodular Dominance via a straightforward convexity argument, but a direct proof that the distribution is NCD is more involved [10]; this theorem shows that such results follow due to a natural relationship between Submodular Dominance and negative dependence rather than any algorithm specific properties.

Although NCD is necessary for Submodular Dominance, it is insufficient on its own. While this insufficiency result was previously known [11],<sup>3</sup> we strengthen it by constructing an example of an NCD distribution which violates Submodular Dominance and is additionally homogeneous, meaning it is distributed only on sets of the same size. Such distributions occur often enough to be of interest, e.g., distributions over the bases of a matroid.

### 1.3 Applications

Besides being a natural question, Submodular Dominance has several applications.

**Submodular Prophet Inequalities.** The Prophet Inequality is a classical problem where a gambler sees the realizations of non-negative random variables one-by-one, choosing a random variable in an online fashion and attempting to maximize its value. The celebrated result of Krengel, Sucheston, and Garling [28, 29] demonstrates a  $1/2$  prophet inequality, meaning that just knowing the distributions in advance is enough to obtain  $1/2$  the expectation obtained by the prophet that knows all the realizations in advance.

Motivated by applications to mechanism design, several works extended the  $1/2$  prophet inequality to gamblers selecting multiple random variables subject to a packing constraint to maximize a linear objective function, e.g., [24, 8, 1, 27, 34]. The *Submodular Prophet Inequality* (SPI) was introduced by Rubinstein and Singla [35] as a further generalization to submodular objective functions to capture combinatorial applications.

One significant complication in SPI is that beyond simple Bernoulli settings, we deal with expectations that are no longer taken over product distributions. Chekuri and Livanos [9] obtain an efficient<sup>4</sup>  $c \cdot (1 - e^{-b}) \cdot (e^{-b} - \epsilon)$  SPI for set systems with solvable polytopes<sup>5</sup> and

<sup>2</sup>  $S \sim \mathbf{x}$  means  $S$  is sampled from the product distribution with marginals  $\mathbf{x}$ .

<sup>3</sup> Observing that certain randomized rounding algorithms give rise to distributions satisfying both Submodular Dominance and NCD, Chekuri, Vondrák, and Zenklusen [11] remarked that there exist NCD distributions which violate Submodular Dominance, so NCD was not sufficient for Submodular Dominance. Our Theorem 5 shows the other direction, that Submodular Dominance implies NCD.

<sup>4</sup> We use efficient to mean algorithms that run in probabilistic polynomial time.

<sup>5</sup> The polytope  $\mathcal{P}_{\mathcal{I}}$  of a set system  $\mathcal{I}$  is formed by taking the convex hull of the indicator vectors of maximal independent sets in  $\mathcal{I}$ , and is solvable if linear objective functions can be efficiently maximized over it.

#### 44:4 Submodular Dominance and Applications

an efficient  $(b, c)$ -selectable greedy online contention resolution scheme (OCRS) for product distributions (see formal definitions in Section 4). Crucially, their result loses a factor of  $e^{-b} - \epsilon$  to handle the non-product distributions of SPI. We use Submodular Dominance to re-analyze the performance of greedy OCRSs in Section 4.3, which allows us to save this factor of  $e^{-b} - \epsilon$  and improve the best known SPIs.

► **Theorem 6** (Submodular Prophet Inequalities). *For fixed  $\epsilon > 0$ , if a set system  $\mathcal{I} \subseteq 2^U$  has a solvable polytope and an efficient  $(b, c)$ -selectable greedy OCRS for product distributions:*

- *There is an efficient  $c \cdot (1 - e^{-b} - \epsilon)$  SPI for monotone non-negative submodular functions.*
- *There is an efficient  $c/4 \cdot (1 - e^{-b} - \epsilon)$  SPI for general non-negative submodular functions.*

*Combining with known greedy OCRSs, this implies efficient SPIs as given in Table 1.*

■ **Table 1** Submodular Prophet Inequalities for different feasibility constraints.

| Feasibility Constraint                         | Prior Best [9] |         | Our Results          |         |
|------------------------------------------------|----------------|---------|----------------------|---------|
|                                                | Monotone       | General | Monotone             | General |
| Uniform Matroid of rank $k \rightarrow \infty$ | 1/4.30         | 1/17.20 | $1 - 1/e - \epsilon$ | 1/6.33  |
| Matroid                                        | 1/7.39         | 1/29.54 | 1/5.02               | 1/20.07 |
| Matching                                       | 1/9.49         | 1/37.93 | 1/6.75               | 1/27.00 |
| Knapsack                                       | 1/17.41        | 1/69.64 | 1/13.40              | 1/53.60 |

It is known that even for offline monotone submodular maximization over uniform matroids, no efficient algorithm can do better than a  $(1 - 1/e)$ -approximation [30]. Thus, we obtain the first optimal efficient  $1 - 1/e - \epsilon$  monotone SPI over large rank uniform matroids.

**Submodular Maximization.** Another application is sampling from WNR distributions as a randomized rounding technique where the integral solution obtains at least the value of the fractional solution in expectation. A common method in submodular optimization is to first maximize the multilinear extension, which Vondrák [39] showed can be done for downward-closed set systems with solvable polytopes. For matroids, we know of methods which round the fractional solutions to sets without losing value [6, 10, 11], but set systems with solvable polytopes are far more general than matroids. Thus, the challenge in going beyond matroids is rounding the multilinear extension. By Submodular Dominance, an algorithm that efficiently generates a WNR distribution for a polytope automatically rounds the multilinear extension, which we show has immediate consequences for submodular maximization (details in Section 5.1).

► **Theorem 7** (Submodular Maximization). *Let  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$  be a monotone submodular function. If a downward-closed set system  $\mathcal{I} \subseteq 2^U$  has a solvable polytope and efficiently admits WNR distributions, there exists an efficient algorithm that returns  $T \in \mathcal{I}$  such that  $\mathbb{E}[f(T)] \geq (1 - 1/e - o(1)) \cdot \max_{S \in \mathcal{I}} f(S)$ .*

**Adaptivity Gaps for Stochastic Probing.** A natural generalization of submodular maximization is by adding stochasticity: replace elements by random variables called items. Such problems are often known as Stochastic Probing [22, 2, 23, 5, 17]. In addition to knowing the distributions of the items, we also allow algorithms to learn the realization of an item after selecting it. This opens up the concept of adaptive algorithms, which modify their behavior

conditioned on such realizations. Though adaptivity can result in better algorithms, it also introduces significant complexity; for example, a decision tree can be of exponential size. Therefore, non-adaptive algorithms may be preferable if their performance is comparable to that of the optimal adaptive algorithm, a concept known as the adaptivity gap. By sampling from WNR distributions to round the multilinear extension, we adapt the analysis of the adaptivity gap upper bound by Asadpour and Nazerzadeh [2] from matroids to any set system for which WNR distributions exist (details in Section 5.2).

► **Theorem 8** (Stochastic Probing). *For a downward-closed set system  $\mathcal{I}$  that admits WNR distributions, the adaptivity gap for Stochastic Probing is upper-bounded by  $\frac{e}{e-1}$ .*

**Contention Resolution Schemes.** Contention resolution schemes (CRS) are another randomized rounding technique, with the concept being formally introduced by [12] for submodular maximization. (Similar but less thoroughly explored notions appear in earlier works such as [4].) Since submodular maximization usually occurs via approximations of the multilinear extension, CRSs have generally been studied with respect to product distributions. Recently, Dughmi [14, 15] initiated the study of CRSs for non-product distributions because of their applications in settings such as the Matroid Secretary Problem. We extend the CRS of [12] for matroids from product distributions to WNR distributions, which gives possible directions to generalize our understanding of CRSs (details in Section 5.3).

► **Theorem 9** (Contention Resolution Schemes). *For a matroid  $\mathcal{M}$ , there exists a  $(1 - 1/e)$ -selectable CRS for any WNR distribution with marginals  $\mathbf{x} \in \mathcal{P}_{\mathcal{M}}$ .*

## 2 WNR and Other Negatively Dependent Distributions

In this section, we first discuss popular notions of negative dependence, and then introduce WNR and study its various properties. Lengthier proofs are deferred to Appendix A.1.

► **Definition 10** (NA). *A distribution  $\mathcal{D}$  over  $2^U$  satisfies Negative Association (NA) if for any monotone  $f, g : 2^U \rightarrow \mathbb{R}$  depending on disjoint sets of elements,  $\text{Cov}_{S \sim \mathcal{D}}[f(S), g(S)] \leq 0$ .*

This property is very similar to the Positive Association (PA) condition in the FKG inequality, with the main difference being the reversed inequality and disjoint sets. Although the FKG Inequality gives a straightforward condition to check for PA, no analogous result exists for NA, and in general, it is difficult to prove that a distribution is NA [26, 32].

Another way to define negative dependence is based on the idea that conditioning on “larger” inclusion events should reduce the probability of other inclusion events.

► **Definition 11** (NR). *A distribution  $\mathcal{D}$  over  $2^U$  satisfies Negative Regression (NR) if for any sets  $R_-, R_+, T \subseteq U$  such that  $R_- \subseteq R_+ \subseteq T$  and any monotone function  $f : 2^U \rightarrow \mathbb{R}$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus T) \mid (S \cap T) = R_+] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus T) \mid (S \cap T) = R_-] .$$

Equivalently,  $\mathcal{D}$  is NR if  $S \setminus T$  conditioned on  $S \cap T = R_-$  stochastically dominates  $S \setminus T$  conditioned on  $S \cap T = R_+$  for all  $R_- \subsetneq R_+ \subseteq T \subseteq U$ . It turns out that NR is also a difficult property to check.

Since NA and NR are both natural forms of negative dependence, it is surprising that the exact relationship between them is unknown. While it is known that NA does not imply NR, it is conjectured that NR implies NA [32]. One might then ask whether we can generalize

NA and NR to get the best of both worlds: a weaker notion of negative dependence that is easier to check while still satisfying many desirable properties. This is our motivations for defining WNR distributions.

We first reformulate the WNR condition in terms of covariance.

► **Claim 12.** The WNR condition (1) is equivalent to  $\text{Cov}_{S \sim \mathcal{D}}[f(S \setminus i), \mathbb{1}_{i \in S}] \leq 0$ .

*Proof.* Let  $\mathcal{D}$  have marginals  $\mathbf{x}$ . The covariance inequality is equivalent to  $\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \cdot \mathbb{1}_{i \in S}] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i)] \mathbb{E}_{S \sim \mathcal{D}}[\mathbb{1}_{i \in S}]$ . Then observe that if we expand LHS by conditioning on  $i$ , the expectation conditioned on  $i \notin S$  is 0 due to the indicator. We can also simplify RHS using  $\mathbb{E}_{S \sim \mathcal{D}}[\mathbb{1}_{i \in S}] = x_i$ , so the covariance inequality is equivalent to

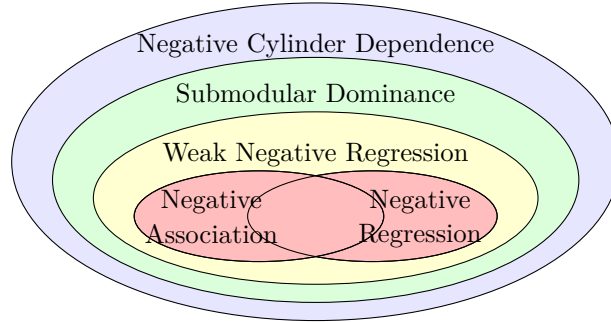
$$x_i \cdot \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \cdot \mathbb{1}_{i \in S} \mid i \in S] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i)] \cdot x_i.$$

Canceling the  $x_i$  and noting that the indicator on LHS is always 1, we have  $\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid i \in S] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i)]$ . This is equivalent to the WNR condition since the expectation is just a weighted sum of the conditional expectation on  $i \in S$  and  $i \notin S$ . ◀

Next, we prove that WNR distributions generalize NA and NR distributions.

► **Proposition 13.** *NA and NR imply WNR, and WNR implies NCD, but the reverse implications do not hold. In other words, the union of NA and NR distributions is a strict subset of WNR distributions, which is a strict subset of NCD distributions.*

*Proof.* The WNR condition is a special case of the NR condition when  $T := \{i\}$ , and by Claim 12, it is also a special case of the NA condition when  $g(S) := \mathbb{1}_{i \in S}$ , so both NA and NR imply WNR. For strict containment, we give a WNR distribution that is neither NA nor NR in Appendix A.1. Theorems 3 and 5 and the example distributions in Section 3 demonstrate that WNR distributions are a strict subset of NCD distributions. ◀



■ **Figure 1** Hierarchy of negative dependence and its relation to Submodular Dominance.

Finally, we observe that WNR satisfies two closure properties, proved in Appendix A.1.

► **Definition 14** (Projection). *Let  $\mathcal{D}$  be a distribution over  $2^U$ . Its projection onto  $U' \subseteq U$  is the distribution which samples  $S \sim \mathcal{D}$  and returns  $S \cap U'$ .*

► **Definition 15** (Products). *Let  $\mathcal{A}$  and  $\mathcal{B}$  be distributions over  $2^A$  and  $2^B$  for disjoint  $A, B$ . Their product distribution independently samples  $S \sim \mathcal{A}$  and  $T \sim \mathcal{B}$ , then returns  $S \cup T$ .*

► **Proposition 16.** *WNR is closed both under projection and under products.*

Joag-Dev and Proschan [26] showed that NA distributions are NCD, closed under projection, closed under products, and closed under taking monotone functions of disjoint subsets of variables.<sup>6</sup> Since WNR shares three of these properties with NA and requires a weaker condition while also generalizing NR, it appears to be a useful notion of negative dependence.

### 3 Towards a Characterization of Submodular Dominance

#### 3.1 WNR is a Sufficient Condition

► **Theorem 3.** *WNR distributions satisfy Submodular Dominance.*

**Proof.** We prove by induction on the number of elements. The base case of 1 element is trivial because the marginals fully specify the distribution.

We will show that any WNR distribution  $\mathcal{D}$  over  $2^{[k]}$  with marginals  $\mathbf{x}$  satisfies Submodular Dominance, assuming by induction that all WNR distributions over  $2^{[k-1]}$  satisfy Submodular Dominance. We assume that  $x_k \neq 0, 1$  because otherwise, we can interpret  $\mathcal{D}$  as a distribution over  $2^{[k-1]}$  and trivially be done.

Let  $\mathcal{D} \setminus k$  and  $\mathbf{x} \setminus k$  denote the projections of  $\mathcal{D}$  and  $\mathbf{x}$  onto  $[k-1]$ . Let  $\mathcal{D}_k$  denote the distribution which samples  $S \sim \mathcal{D} \setminus k$ , then returns  $S \cup k$  w.p.  $x_k$  and returns  $S$  otherwise, i.e.,  $\mathcal{D}_k$  is  $\mathcal{D}$  but with element  $k$  sampled independently.

Let  $f : 2^{[k]} \rightarrow \mathbb{R}$  be a submodular function. To prove Submodular Dominance, we will show the following inequalities hold:

$$\mathbb{E}_{S \sim \mathcal{D}} [f(S)] \stackrel{\text{Claim 18}}{\geq} \mathbb{E}_{S \sim \mathcal{D}_k} [f(S)] \stackrel{\text{Claim 17}}{\geq} \mathbb{E}_{S \sim \mathbf{x}} [f(S)] . \quad (2)$$

► **Claim 17.** The second inequality of (2) holds, i.e.,  $\mathbb{E}_{S \sim \mathcal{D}_k} [f(S)] \geq \mathbb{E}_{S \sim \mathbf{x}} [f(S)]$ .

**Proof.** Since  $k$  is independently sampled in both  $\mathcal{D}_k$  and  $\mathbf{x}$ , we can write

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_k} [f(S)] &= \mathbb{E}_{S \sim \mathcal{D} \setminus k} [x_k \cdot f(S \cup k) + (1 - x_k) \cdot f(S)] \quad \text{and} \\ \mathbb{E}_{S \sim \mathbf{x}} [f(S)] &= \mathbb{E}_{S \sim \mathbf{x} \setminus k} [x_k \cdot f(S \cup k) + (1 - x_k) \cdot f(S)] . \end{aligned}$$

Convex combinations of submodular functions are submodular,  $\mathcal{D} \setminus k$  is WNR by closure under projection (Proposition 16), and the marginals of  $\mathcal{D} \setminus k$  are equal to the marginals of  $\mathbf{x} \setminus k$ . Therefore, by the induction hypothesis,

$$\mathbb{E}_{S \sim \mathcal{D} \setminus k} [x_k \cdot f(S \cup k) + (1 - x_k) \cdot f(S)] \geq \mathbb{E}_{S \sim \mathbf{x} \setminus k} [x_k \cdot f(S \cup k) + (1 - x_k) \cdot f(S)] ,$$

which implies  $\mathbb{E}_{S \sim \mathcal{D}_k} [f(S)] \geq \mathbb{E}_{S \sim \mathbf{x}} [f(S)]$ . ◁

► **Claim 18.** The first inequality of (2) holds, i.e.,  $\mathbb{E}_{S \sim \mathcal{D}} [f(S)] \geq \mathbb{E}_{S \sim \mathcal{D}_k} [f(S)]$ .

**Proof.** Expanding expectations for  $\mathcal{D}_k$  (as in the proof of Claim 17) and moving it to LHS,

$$\mathbb{E}_{S \sim \mathcal{D}} [f(S) - x_k \cdot f(S \cup k) - (1 - x_k) \cdot f(S \setminus k)] \geq 0 .$$

<sup>6</sup> This last property is useful in the setting of continuous random variables studied in [26] because properties closed under convolutions are extremely powerful. However, it is not so relevant in the discrete settings we study.

Conditioning on  $k$  yields

$$\begin{aligned} & x_k \cdot \mathbb{E}_{S \sim \mathcal{D}} [f(S \cup k) - x_k \cdot f(S \cup k) - (1 - x_k) \cdot f(S \setminus k) \mid k \in S] \\ & + (1 - x_k) \cdot \mathbb{E}_{S \sim \mathcal{D}} [f(S \setminus k) - x_k \cdot f(S \cup k) - (1 - x_k) \cdot f(S \setminus k) \mid k \notin S] \geq 0, \end{aligned}$$

which simplifies to

$$x_k(1 - x_k) \left( \mathbb{E}_{S \sim \mathcal{D}} [f(S \cup k) - f(S \setminus k) \mid k \in S] + \mathbb{E}_{S \sim \mathcal{D}} [f(S \setminus k) - f(S \cup k) \mid k \notin S] \right) \geq 0.$$

Dividing out  $x_k(1 - x_k)$  and moving the second term to RHS gives

$$\mathbb{E}_{S \sim \mathcal{D}} [f(S \cup k) - f(S \setminus k) \mid k \in S] \geq \mathbb{E}_{S \sim \mathcal{D}} [f(S \cup k) - f(S \setminus k) \mid k \notin S]. \quad (3)$$

Let  $f_k(S) := f(S \cup k) - f(S \setminus k)$ .  $f_k$  does not depend on  $k$ , and by the submodularity of  $f$ ,  $-f_k$  is a monotone function. Thus, (3) is directly implied by the WNR condition (1).  $\triangleleft$

Claims 17 and 18 complete the proof of Theorem 3.  $\blacktriangleleft$

The following proposition, which we prove in Appendix A.2, shows that WNR is not a necessary condition for Submodular Dominance.

► **Proposition 19.** *The distribution  $\mathcal{D}$  which samples uniformly from  $\emptyset, \{1\}, \{2\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$  satisfies Submodular Dominance, but  $\mathcal{D}$  violates WNR for  $f(S) := \max(\mathbb{1}_{1 \in S}, \mathbb{1}_{2 \in S})$  and  $i = 3$ .*

### 3.2 NCD is a Necessary Condition

Since WNR is not equivalent to Submodular Dominance, we search for necessary conditions to better understand the relationship between negative dependence and Submodular Dominance.

► **Theorem 5.** *All distributions that satisfy Submodular Dominance are NCD.*

**Proof.** Let  $\mathcal{D}$  be a distribution over  $2^U$  with marginals  $\mathbf{x}$  which satisfies Submodular Dominance. For any  $T \subseteq U$ , consider the functions

$$f_T(S) := 1 - \mathbb{1}_{T \subseteq S^c} \quad \text{and} \quad g_T(S) := |S \cap T| - \mathbb{1}_{T \subseteq S}.$$

Equivalently,  $f_T, g_T$  are the rank functions of the uniform matroids of rank 1 and  $|T| - 1$  over the ground set  $T$ . The only fact about matroid rank functions we use here is that all matroid rank functions are submodular (though one can also easily check that  $f_T, g_T$  are submodular via the definition of submodularity). Since subtracting a linear function from a submodular function results in a submodular function,  $-\mathbb{1}_{T \subseteq S}$  and  $-\mathbb{1}_{T \subseteq S^c}$  are submodular functions. Thus, by Submodular Dominance we have

$$\begin{aligned} -\Pr_{S \sim \mathcal{D}} [T \subseteq S] &= \mathbb{E}_{S \sim \mathcal{D}} [-\mathbb{1}_{T \subseteq S}] \geq \mathbb{E}_{S \sim \mathbf{x}} [-\mathbb{1}_{T \subseteq S}] = -\Pr_{S \sim \mathbf{x}} [T \subseteq S] \quad \text{and} \\ -\Pr_{S \sim \mathcal{D}} [T \subseteq S^c] &= \mathbb{E}_{S \sim \mathcal{D}} [-\mathbb{1}_{T \subseteq S^c}] \geq \mathbb{E}_{S \sim \mathbf{x}} [-\mathbb{1}_{T \subseteq S^c}] = -\Pr_{S \sim \mathbf{x}} [T \subseteq S^c]. \end{aligned}$$

Multiplying both sides by  $-1$  yields the definition of NCD (Definition 4).  $\blacktriangleleft$

The following two propositions, which we prove in Appendix A.2, show that NCD is not a sufficient condition for Submodular Dominance.

► **Proposition 20.** *The distribution  $\mathcal{D}$  over  $2^{[4]}$  which chooses uniformly at random  $i \in [4]$ , then returns w.p.  $1/2$  either  $i$  or  $[4] \setminus i$ , is NCD. However,  $\mathcal{D}$  violates Submodular Dominance for the submodular function  $f(S) := \min(2, |S|)$ .*

► **Proposition 21.** *The distribution  $\mathcal{D}$  over  $2^{[8]}$  which chooses uniformly at random  $i \in A := \{1, 2, 3, 4\}$  and  $j \in B := \{5, 6, 7, 8\}$ , then returns w.p.  $1/2$  either  $i \cup (B \setminus j)$  or  $(A \setminus i) \cup j$ , is NCD. However,  $\mathcal{D}$  violates Submodular Dominance for the submodular function  $f(S) := \min(2, |S \cap A|)$ .*

Thus, the class of distributions which satisfy Submodular Dominance is a strict subset of NCD distributions and a strict superset of WNR distributions. It is unclear whether the “right” answer will turn out to be a useful notion of negative dependence.

## 4 Applications to Submodular Prophet Inequalities

In SPI, we have *items*  $U$ , which are discrete random variables with disjoint images and arbitrary probability mass functions. We denote realizations of items as *elements*. WLOG, let the image of  $i \in U$  be  $\{ij : j \in [m]\}$ , and let the realization of  $i$  be  $ij$  w.p.  $p_{ij}$ . Let  $E := [n] \times [m]$  denote the set of elements. The distributions of each item are independent and known to us in advance.

We are given a set system  $\mathcal{I} \subseteq 2^U$  and a submodular objective function  $f : 2^E \rightarrow \mathbb{R}_{\geq 0}$ . Notice that while the items are independent, the elements do not follow a product distribution. As we are optimizing over the element-space, this is a non-trivial complication.

Each item arrives one-by-one. When an item arrives, we learn its realization, and must choose whether to accept or reject it. The set of accepted items must be in  $\mathcal{I}$ , and the goal is to maximize  $f$  on the realizations of the accepted items. The arrival order is chosen by an *almighty adversary*, who knows in advance the outcomes of all randomness, such as the item realizations, the decisions of our algorithm, etc.

If there exists an  $\alpha$ -competitive algorithm compared to the prophet, we say there is an  $\alpha$  SPI. Rubinstein and Singla [35] proved  $\Omega(1)$  SPIs over matroids, and Chekuri and Livanos [9] refined their analysis to obtain better constants, as well as results for a broader range of set systems. We further improve upon their approach using Submodular Dominance, obtaining results such as the first tight SPI for large rank uniform matroids.

### 4.1 Core Approach: SPI for Bernoulli Items

Before tackling the full problem, it is helpful to first consider a simplified version, *Bernoulli SPI*, where each item  $i$  is only a Bernoulli random variable, taking value 1 w.p.  $p_i$  and taking value 0 otherwise. Here, there is no notion of elements (or rather, elements are effectively synonymous with items), so we consider a submodular objective function  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ . As in the full problem, we have a set system constraint  $\mathcal{I} \subseteq 2^U$ .

This is quite similar to the problem of submodular maximization from Section 5.1, but with a stochastic component (each item being usable only w.p.  $p_i$ ) and an online component (items are revealed one-by-one). Therefore, it makes sense to borrow the high level approach of optimizing the multilinear extension  $F$ , then rounding the fractional solution. Since  $\mathcal{I}$  is a discrete constraint and  $F$  is a continuous function, the following relaxation is useful in offline submodular maximization:

► **Definition 22.** *For any downward-closed set system  $\mathcal{I} \subseteq 2^U$ , its polytope  $\mathcal{P}_{\mathcal{I}} \subseteq [0, 1]^n$  is the convex hull of the indicator vectors representing the maximal sets of  $\mathcal{I}$ .*



For Bernoulli SPI, we consider a modified polytope  $\mathcal{P}'_{\mathcal{I}} := \{\mathbf{x} \in [0, 1]^n : \mathbf{x} \in \mathcal{P}_{\mathcal{I}}, x_i \leq p_i \forall i \in U\}$ . Since the fractional solution corresponds to a distribution over  $\mathcal{I}$ , the additional constraint  $x_i \leq p_i$  ensures that no item is included more often than it takes value 1. It turns out that we can efficiently optimize  $F$  over  $\mathcal{P}'_{\mathcal{I}}$  under mild conditions.

As for rounding, we can use *online contention resolution schemes* (OCRS). OCRSs function in the following setting: we have a set system  $\mathcal{I} \subseteq 2^U$  and a distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x}$ . Let items  $i \in S$  for some  $S \sim \mathcal{D}$  be called *active*. The items then arrive one-by-one in adversarial order. When item  $i$  arrives, we learn whether it is active, and if so, must decide to accept or reject it, subject to the set of accepted items being in  $\mathcal{I}$ . An OCRS  $\pi_{\mathcal{I}, \mathcal{D}}$  is an algorithm that plays this game. The following notion is a way to measure the performance of an OCRS.

► **Definition 23** ( $(b, c)$ -selectable OCRS). *For  $b, c \in [0, 1]$ , a set system  $\mathcal{I} \subseteq 2^U$ , and a distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x} \in b \cdot \mathcal{P}_{\mathcal{I}}$ , an OCRS  $\pi_{\mathcal{I}, \mathcal{D}}$  is  $(b, c)$ -selectable if the probability of  $\pi_{\mathcal{I}, \mathcal{D}}$  accepting  $i$  is at least  $c \cdot x_i$  for all  $i \in U$ . If  $b = 1$ , we say  $\pi_{\mathcal{I}, \mathcal{D}}$  is  $c$ -selectable.*

Feldman, Svensson, and Zenklusen [19] obtained the following approximation result for rounding via *greedy OCRSs* (we omit the definition as it is not relevant).

► **Proposition 24** ([19]). *For a set system  $\mathcal{I} \subseteq 2^U$ , a monotone submodular function  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ , and  $\mathbf{x} \in b \cdot \mathcal{P}_{\mathcal{I}}$ , applying a  $(b, c)$ -selectable greedy OCRS to  $S \sim \mathbf{x}$  obtains  $T \in \mathcal{I}$  such that  $\mathbb{E}_{S \sim \mathbf{x}}[f(T)] \geq c \cdot F(\mathbf{x})$ . Further, the greedy OCRS can be efficiently modified such that even for non-monotone  $f$ , the modified greedy OCRS obtains  $T \in \mathcal{I}$  such that  $\mathbb{E}_{S \sim \mathbf{x}}[f(T)] \geq c/4 \cdot F(\mathbf{x})$ .*

In Bernoulli SPI, there is no notion of elements and we optimize over items. Thus, the distribution of active items is already a product distribution, and simply applying greedy OCRSs already yields approximation results using Proposition 24.

## 4.2 Generalizing to Arbitrary Discrete Random Variables

We now return to the full version of SPI. Again, let  $U$  be the set of items,  $\mathcal{I} \subseteq 2^U$  be a downward-closed set system with a solvable polytope,  $E := [n] \times [m]$  be the set of elements,  $\mathbf{p} \in [0, 1]^{nm}$  be the element realization probabilities, and  $f : 2^E \rightarrow \mathbb{R}_{\geq 0}$  be a submodular function. The first step is to compute a fractional solution. Chekuri and Livanos [9] define the polytope

$$\mathcal{P}''_{\mathcal{I}} := \{\mathbf{x} \in [0, 1]^{nm} : \exists \mathbf{z} \in \mathcal{P}_{\mathcal{I}} \text{ satisfying } \sum_j x_{ij} = z_i \forall i \in U, x_{ij} \leq p_{ij} \forall ij \in E\}.$$

Here, the summation constraint is a natural relaxation of  $\mathcal{P}'_{\mathcal{I}}$  from the item-space to the element-space. Chekuri and Livanos prove a series of results<sup>7</sup> which culminates in the following (note that we do not require monotonicity of  $f$ ):

► **Proposition 25** ([9]). *Let OPT be the expectation obtained by the prophet,  $\mathcal{I} \subseteq 2^U$  be a set system with a solvable polytope, and  $f : 2^E \rightarrow \mathbb{R}_{\geq 0}$  be a submodular function. Then for any fixed  $\epsilon > 0$ , we can efficiently compute  $\mathbf{x} \in b \cdot \mathcal{P}''_{\mathcal{I}}$  such that  $F(\mathbf{x}) \geq (1 - e^{-b} - \epsilon) \cdot \text{OPT}$ .*

<sup>7</sup> See Section 3 of [9] for details, in particular, Claim 3.4, Theorem 1.3, and Remark 3.7.

It remains to round the fractional solution. While it is fairly straightforward to convert a  $(b, c)$ -selectable greedy OCRS for the item-space to a  $(b, c)$ -selectable greedy OCRS for the element-space (indeed, we do this in Algorithm 1), we cannot obtain approximation results directly from Proposition 24 like in the Bernoulli case because the distribution of elements is not a product distribution. Chekuri and Livanos handle this by incurring an additional loss of  $e^{-b} - \epsilon$  to “mask” the elements under a product distribution. We save this factor by re-analyzing a simpler algorithm.

### 4.3 Improved Analysis

Let  $\mathbf{x} \in b \cdot \mathcal{P}_{\mathcal{I}}''$  be the solution computed as per Proposition 25. Define  $x_i := \sum_j x_{ij}$ , and define  $\vec{x} := (x_i : i \in U)$ . By definition of  $\mathcal{P}_{\mathcal{I}}''$  and the fact that  $\mathbf{x} \in b \cdot \mathcal{P}_{\mathcal{I}}''$ , we have that  $\vec{x} \in b \cdot \mathcal{P}_{\mathcal{I}}$ , so let  $\pi_{\mathcal{I}, \vec{x}}$  be an efficient  $(b, c)$ -selectable greedy OCRS.

We first consider monotone  $f$ . Our rounding algorithm is almost identical to [9, Algorithm 1], removing some steps that our improved analysis demonstrates to be unnecessary.

■ **Algorithm 1** MONOTONE ROUNDING  $(U, E, \mathbf{p}, f, \mathbf{x}, \pi_{\mathcal{I}, \vec{x}})$ .

---

```

 $T_{\text{ALG}} = \emptyset$
for $t \leftarrow 1$ to n do
 Let $i \in U$ be the item that arrives on day t
 Let $ij \in E$ be the realization of i
 With probability x_{ij}/p_{ij} , reveal active i to $\pi_{\mathcal{I}, \vec{x}}$, otherwise reveal inactive i to $\pi_{\mathcal{I}, \vec{x}}$
 if $\pi_{\mathcal{I}, \vec{x}}$ accepts i then
 | $T_{\text{ALG}} \leftarrow T_{\text{ALG}} \cup \{ij\}$
 end
end
Return T_{ALG}

```

---

Denote element  $ij$  as *active* when  $ij$  is the realization of  $i$ , and Algorithm 1 reveals active  $i$  to  $\pi_{\mathcal{I}, \vec{x}}$ . Since the elements do not follow a product distribution, we cannot apply Proposition 24 even though the algorithm acts like a greedy OCRS. However, we provide a new analysis which states that a  $c$ -approximation for product distributions implies a  $c$ -approximation for the following wider class of distributions.

► **Definition 26.** A product of singletons distribution over  $2^E$  with marginals  $\mathbf{x} \in [0, 1]^{nm}$  such that  $\sum_j x_{ij} \leq 1$  for all  $i$  is a distribution which independently samples 0 or 1 elements from each set  $\{ij : j \in [m]\}$  according to the marginals  $\mathbf{x}$ .

It is not difficult to see that the active elements follow a product of singletons distribution with marginals  $\mathbf{x}$ . The following lemma, which we prove in Appendix A.3, draws a connection between product of singletons distributions and product distributions.

► **Lemma 27.** Let  $\mathcal{D}$  be a product of singletons distribution over  $2^E$  with marginals  $\mathbf{x} \in [0, 1]^{nm}$ . Let  $x_i := \sum_j x_{ij}$ , and let  $\vec{x} := (x_i : i \in U)$ . For any  $\mathbf{u} \in [m]^n$ , let  $E_{\mathbf{u}} := \{iu_i : i \in U\}$  and let  $\mathcal{D}_{\mathbf{u}}$  be a product distribution over  $2^{E_{\mathbf{u}}}$  with marginals  $\vec{x}$ . Then for any  $g : 2^E \rightarrow \mathbb{R}$ ,

$$\mathbb{E}_{S \sim \mathcal{D}}[g(S)] = \sum_{\mathbf{u} \in [m]^n} \left( \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}}[g(S)] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right). \quad (4)$$

In simpler terms,  $\mathcal{D}_{\mathbf{u}}$  is the distribution which samples  $S \sim \mathcal{D}$ , then replaces each element  $ij \in S$  with the element  $iu_i$ . Lemma 27 states that any product of singletons distribution with marginals  $\mathbf{x}$  can be written as a convex combination of product distributions with marginals  $\vec{x}$ .

► **Lemma 28.** *For monotone  $f$ , Algorithm 1 returns  $T_{\text{ALG}}$  such that  $\mathbb{E}[f(T_{\text{ALG}})] \geq c \cdot F(\mathbf{x})$ .*

**Proof.** Let  $\mathcal{D}$  be the distribution of active elements. While the adversary sees the item realizations and which items Algorithm 1 will reveal as active to  $\pi_{\mathcal{I}, \vec{x}}$ , the adversary cannot influence  $\mathcal{D}$  because the decisions to reveal active  $i$  do not depend on the item ordering.

Therefore, it is valid for us to “partition” the outcomes of randomness contributing to  $\mathcal{D}$ . Since  $\mathcal{D}$  is a product of singletons distribution with marginals  $\mathbf{x}$ , Lemma 27 tells us that there exists a partition such that each part is a product distribution  $\mathcal{D}_{\mathbf{u}}$  over  $2^{E_{\mathbf{u}}}$  with marginals  $\vec{x}$ . We fix some  $\mathbf{u} \in [m]^n$  and analyze the performance of Algorithm 1 on the subset of randomness corresponding to the distribution  $\mathcal{D}_{\mathbf{u}}$ .

Since  $\mathcal{D}_{\mathbf{u}}$  is a product distribution,  $\vec{x} \in b \cdot \mathcal{P}_{\mathcal{I}}$ , and the algorithm copies the acceptances of  $\pi_{\mathcal{I}, \vec{x}}$ , Algorithm 1 acts exactly like a  $(b, c)$ -selectable greedy OCRS over  $\mathcal{D}_{\mathbf{u}}$ . Most importantly,  $\mathcal{D}_{\mathbf{u}}$  being product distribution means we can apply Proposition 24 to get

$$\mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [f(T_{\text{ALG}})] \geq c \cdot \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [f(S)] .$$

$T_{\text{ALG}}$  is implicitly a randomized function of  $S$ , so we can rewrite LHS as

$$\mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} \left[ \mathbb{E}[f(T_{\text{ALG}}) \mid S = S'] \right] \geq c \cdot \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [f(S)] ,$$

where the inner expectation is taken over the possible randomization of the underlying greedy OCRS  $\pi_{\mathcal{I}, \vec{x}}$  and the adversarial ordering of the items. As this holds for any  $\mathbf{u}$ , weighting the inequality and summing over all  $\mathbf{u} \in [m]^n$  yields

$$\sum_{\mathbf{u} \in [m]^n} \left( \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} \left[ \mathbb{E}[f(T_{\text{ALG}}) \mid S = S'] \right] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right) \geq \sum_{\mathbf{u} \in [m]^n} \left( c \cdot \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [f(S)] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right) .$$

Factoring out the  $c$  on RHS, then applying Lemma 27 to the functions  $\mathbb{E}[f(T_{\text{ALG}}) \mid S = S']$  and  $f(S)$  simplifies to  $\mathbb{E}_{S \sim \mathcal{D}} [f(T_{\text{ALG}})] \geq c \cdot \mathbb{E}_{S \sim \mathcal{D}} [f(S)]$ .

A distribution which samples only sets of size 0 or 1 is WNR because conditioning on inclusion of an element excludes all other elements. Further, products of WNR distributions are WNR (Proposition 16). Thus, product of singletons distributions are WNR, and applying Submodular Dominance (Theorem 3) on  $\mathcal{D}$  gives the following and completes the proof:

$$\mathbb{E}_{S \sim \mathcal{D}} [f(T_{\text{ALG}})] \geq c \cdot \mathbb{E}_{S \sim \mathcal{D}} [f(S)] \geq c \cdot F(\mathbf{x}) . \quad \blacktriangleleft$$

► **Remark 29.** For general  $f$ , we can replace the greedy OCRS  $\pi_{\mathcal{I}, \vec{x}}$  by its efficient modification mentioned in Proposition 24, then just repeat the proof of Lemma 28. We lose an additional factor of  $1/4$  when we invoke Proposition 24 on the modified greedy OCRS, which gives us a  $c/4$ -approximation algorithm when  $f$  is not monotone.

► **Theorem 6 (Submodular Prophet Inequalities).** *For fixed  $\epsilon > 0$ , if a set system  $\mathcal{I} \subseteq 2^U$  has a solvable polytope and an efficient  $(b, c)$ -selectable greedy OCRS for product distributions:*

- *There is an efficient  $c \cdot (1 - e^{-b} - \epsilon)$  SPI for monotone non-negative submodular functions.*
  - *There is an efficient  $c/4 \cdot (1 - e^{-b} - \epsilon)$  SPI for general non-negative submodular functions.*
- Combining with known greedy OCRSs, this implies efficient SPIs as given in Table 1.*

**Proof.** Combining Proposition 25 and Lemma 28 gives us a  $c \cdot (1 - e^{-b} - \epsilon)$  SPI for monotone non-negative submodular functions, and, as noted in Remark 29, a similar argument gives us a  $c/4 \cdot (1 - e^{-b} - \epsilon)$  SPI for general non-negative submodular functions. From [19], we can efficiently construct greedy OCRSs satisfying the following properties:

- $(b, 1 - b)$ -selectable over matroids, for  $b \in [0, 1]$ .
- $(b, e^{-2b})$ -selectable over matchings, for  $b \in [0, 1]$ .
- $(b, \frac{1-2b}{2-2b})$ -selectable over knapsacks, for  $b \in [0, 1/2]$ .
- $(1 - o(1))$ -selectable over uniform matroids of rank  $k \rightarrow \infty$  [9].

To obtain the results in Table 1, we simply choose  $b$  which maximizes  $c \cdot (1 - e^{-b} - \epsilon)$ . ◀

## 5 Applications to Rounding

A common problem setting is optimization constrained to some feasible set system  $\mathcal{I}$ .

► **Definition 30.** For a downward-closed set system  $\mathcal{I} \subseteq 2^U$ , its polytope  $\mathcal{P}_{\mathcal{I}} \subseteq [0, 1]^n$  is the convex hull of the indicator vectors representing the maximal sets of  $\mathcal{I}$ .

Under mild conditions, we can efficiently optimize over the polytope, then round the fractional solution  $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$  to an integral solution  $S \in \mathcal{I}$ . It is natural to think of  $\mathbf{x}$  as a distribution over  $\mathcal{I}$  with those marginals. If these distributions exhibit certain properties, then sampling can be an effective rounding technique. We give results for set systems which satisfy the following property:

► **Definition 31.** A set system  $\mathcal{I} \subseteq 2^U$  admits WNR distributions if for any  $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$ , there exists a WNR distribution over  $\mathcal{I}$  with marginals  $\mathbf{x}$ . If we can efficiently sample from these distributions, we say  $\mathcal{I}$  efficiently admits WNR distributions.

### 5.1 Submodular Maximization

For a set system  $\mathcal{I} \subseteq 2^U$  and a monotone submodular function  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ , a classical optimization problem is to efficiently find  $T \in \mathcal{I}$  such that  $f(T)$  is a good approximation of  $\max_{S \in \mathcal{I}} f(S)$ . We start by optimizing over  $\mathcal{P}_{\mathcal{I}}$ .

► **Proposition 32** ([39]). For any set system  $\mathcal{I}$  with a solvable polytope, we can efficiently compute  $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$  such that  $F(\mathbf{x}) \geq (1 - 1/e - o(1)) \cdot \max_{S \in \mathcal{I}} f(S)$ .

Now, we want to round  $\mathbf{x}$  to an integral solution with at least value  $F(\mathbf{x})$ . Pipage [6] and randomized swap rounding [10] achieve this for matroid polytopes, but it is unclear how to extend it. Submodular Dominance gives new approaches for submodular maximization.

► **Theorem 7** (Submodular Maximization). Let  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$  be a monotone submodular function. If a downward-closed set system  $\mathcal{I} \subseteq 2^U$  has a solvable polytope and efficiently admits WNR distributions, there exists an efficient algorithm that returns  $T \in \mathcal{I}$  such that  $\mathbb{E}[f(T)] \geq (1 - 1/e - o(1)) \cdot \max_{S \in \mathcal{I}} f(S)$ .

**Proof.** We compute  $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$  as per Proposition 32, then sample from a WNR distribution over  $\mathcal{I}$  with marginals  $\mathbf{x}$ . By Theorem 3, this returns  $T \in \mathcal{I}$  such that  $\mathbb{E}[f(T)] \geq F(\mathbf{x}) \geq (1 - 1/e - o(1)) \cdot \max_{S \in \mathcal{I}} f(S)$ . ◀

### 5.2 Adaptivity Gaps for Stochastic Probing

*Stochastic Probing* is a generalization of submodular maximization with randomized inputs. Elements are replaced by *items*, and we *probe* items to learn their realizations. The goal is to maximize the expectation of a function over the realizations of probed items. We consider a simple version of the problem where we have a monotone submodular function  $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$  and each item  $X_i$  contains element  $i$  independently w.p.  $p_i$  and is empty otherwise.

As probing reveals information, we differentiate between *adaptive algorithms*, which behave differently conditioned on the realizations of probed items, and *non-adaptive algorithms*.

► **Definition 33.** *The adaptivity gap is the ratio between the expectations obtained by the optimal adaptive algorithm and optimal non-adaptive algorithm.*

Asadpour and Nazerzadeh [2] give a tight result that the adaptivity gap for stochastic probing subject to a matroid constraint is  $\frac{e}{e-1}$ . The approach is to first define an auxiliary function  $f'$ , where  $f'(S)$  is the expectation of  $f$  upon probing items  $\{X_i : i \in S\}$ . It turns out that the multilinear extension  $F'$  of  $f'$  satisfies the property that  $\max_{\mathbf{x} \in \mathcal{P}_{\mathcal{I}}} F'(\mathbf{x})$  is a  $(1 - 1/e)$ -approximation of the expectation obtained by the optimal adaptive algorithm.<sup>8</sup>

With this approximation result, the idea is to use  $\mathbf{x}$  to design a non-adaptive algorithm. Simply probing each item w.p.  $x_i$  may violate the matroid constraint, so Asadpour and Nazerzadeh design non-adaptive algorithms using pipage rounding. We go beyond matroids by designing non-adaptive algorithms using WNR distributions.

► **Theorem 8 (Stochastic Probing).** *For a downward-closed set system  $\mathcal{I}$  that admits WNR distributions, the adaptivity gap for Stochastic Probing is upper-bounded by  $\frac{e}{e-1}$ .*

**Proof.** Our analysis follows that of [2] until we obtain  $\arg\max_{\mathbf{x} \in \mathcal{P}_{\mathcal{I}}} F'(\mathbf{x})$ . As  $\mathcal{I}$  admits WNR distributions, there exists a WNR distribution  $\mathcal{D}$  over  $\mathcal{I}$  with marginals  $\mathbf{x}$ . By Theorem 3, we have  $\mathbb{E}_{S \sim \mathcal{D}}[f'(S)] \geq F'(\mathbf{x})$ . Therefore, the non-adaptive algorithm which samples  $S \sim \mathcal{D}$  and probes  $\{X_i : i \in S\}$  obtains at least  $F'(\mathbf{x})$  in expectation. No adaptive algorithm can obtain expectation greater than  $\frac{e}{e-1} \cdot F'(\mathbf{x})$ , so  $\frac{e}{e-1}$  upper-bounds the adaptivity gap. ◀

### 5.3 Contention Resolution Schemes

► **Definition 34 (CRS).** *A contention resolution scheme (CRS) for a set system  $\mathcal{I} \subseteq 2^U$  and a distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x}$  is a (possibly randomized) mapping  $\pi_{\mathcal{I}, \mathcal{D}} : 2^U \rightarrow \mathcal{I}$  such that for all  $S \subseteq U$ , we have  $\pi_{\mathcal{I}, \mathcal{D}}(S) \subseteq S$ .*

Contention resolution schemes have applications to submodular maximization as a rounding technique. The following is the simplest measure of performance for a CRS.

► **Definition 35 (c-selectable CRS).** *For  $c \in [0, 1]$ , a set system  $\mathcal{I} \subseteq 2^U$ , and a distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$ , a CRS  $\pi_{\mathcal{I}, \mathcal{D}}$  is c-selectable if  $\Pr_{S \sim \mathcal{D}}[i \in \pi_{\mathcal{I}, \mathcal{D}}(S)] \geq c \cdot x_i$  for all  $i \in U$ .*

In submodular maximization, rounding fractional solutions is closely related to the multilinear extension, so study of CRSs is primarily centered around product distributions. However, as Dughmi [14, 15] recently showed, CRSs over non-product distributions have applications in settings such as the Matroid Secretary Problem. We use Submodular Dominance to extend a selectability result to WNR distributions, which provides a direction by which other CRS results may be generalized to correlated distributions.

► **Theorem 9 (Contention Resolution Schemes).** *For a matroid  $\mathcal{M}$ , there exists a  $(1 - 1/e)$ -selectable CRS for any WNR distribution with marginals  $\mathbf{x} \in \mathcal{P}_{\mathcal{M}}$ .*

**Proof.** [12] demonstrated this result for product distributions via strong LP duality. Following the same idea, Dughmi [14] reduced this result to proving Submodular Dominance:

<sup>8</sup> We omit many of the finer details because our result does not alter this part of the analysis. Section 3 of [2] covers this in depth.

► **Proposition 36** ([14]). *For a matroid  $\mathcal{M}$  and a distribution  $\mathcal{D}$  over  $2^U$  with marginals  $\mathbf{x} \in \mathcal{P}_{\mathcal{M}}$ , there exists a  $(1 - 1/e)$ -selectable CRS if every submodular function  $f : 2^U \rightarrow \mathbb{R}$  satisfies  $\mathbb{E}_{S \sim \mathcal{D}}[f(S)] \geq \mathbb{E}_{S \sim \mathbf{x}}[f(S)]$ .*

Theorem 9 follows directly from Proposition 36 and Theorem 3. ◀

## 6 Conclusion and Open Questions

In this paper, we explore Submodular Dominance and its applications. In the process, we introduce a notion of negative dependence that we refer to as Weak Negative Regression (WNR), which is a natural generalization of both Negative Association (NA) and Negative Regression (NR) and may be of use in other applications. We prove that WNR distributions satisfy Submodular Dominance, and that all distributions satisfying Submodular Dominance also satisfy Negative Cylinder Dependence (NCD). Finally, we give a variety of applications for Submodular Dominance, improving the best known submodular prophet inequalities, developing new rounding techniques, and generalizing results for contention resolution schemes to negatively dependent distributions.

**Sampling for More General Set Systems.** Although our results for negatively distributions satisfying Submodular Dominance already have several applications, their usage could be broadened further by finding new techniques to generate negatively dependent distributions. An interesting future direction is to design algorithms to sample from negatively dependent distributions for more general set systems. For example, can we efficiently sample from a WNR/NA/NR distribution for any marginals in a given matroid polytope? We remark that [33] claimed such a result for NA distributions, but later, a gap in their proof was found. Max-entropy distributions over matroids are also not negatively dependent in general, as it is known that there exist matroids for which the uniform distribution (which is entropy-maximizing without constrained marginals) exhibits positive correlations [25].

**Approximate Submodular Dominance.** While we showed that NCD distributions do not always satisfy Submodular Dominance, one question is whether these weaker notions of negative dependence obtain constant-factor approximation variants of Submodular Dominance; that is, for a non-negative submodular function  $f$ , what distributions satisfy  $\mathbb{E}_{S \sim \mathcal{D}}[f(S)] \geq O(1) \cdot F(\mathbf{x})$ ? What about for monotone  $f$ ? Another direction is to generalize Submodular Dominance to a larger class of functions. XOS functions are functions that can be expressed as the maximum of a collection of linear functions, and are a strict superset of submodular functions. While no non-product distribution satisfies “XOS Dominance” (consider  $\max(X_1 + X_2, 1)$  and  $\max(X_1, X_2)$ , which are both XOS; the former decreases in expectation if  $X_1$  and  $X_2$  are negatively correlated, the latter if  $X_1$  and  $X_2$  are positively correlated), we might similarly ask if approximate versions hold for XOS functions.

**Concentration Inequalities.** Another important direction is understanding concentration inequalities for negatively dependent distributions. Submodular Dominance demonstrates that the expectation of negatively dependent distributions behaves favorably compared to product distributions, but we may also be interested in whether these distributions are concentrated around their mean. We know dimension-dependent concentration inequalities for arbitrary Lipschitz functions over NR distributions [21]. Proving dimension-independent concentration inequalities for submodular functions is an interesting future direction.



## References

- 1 Saeed Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 512–521, 2011.
- 2 Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Manag. Sci.*, 62(8):2374–2391, 2016. doi:10.1287/mnsc.2015.2254.
- 3 Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Found. Trends Mach. Learn.*, 6(2-3):145–373, 2013.
- 4 Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory Comput.*, 8(1):533–565, 2012. doi:10.4086/toc.2012.v008a024.
- 5 Domagoj Bradac, Sahil Singla, and Goran Zuzic. (Near) optimal adaptivity gaps for stochastic multi-value probing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, pages 49:1–49:21, 2019.
- 6 Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, pages 182–196, 2007.
- 7 Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
- 8 Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 311–320, 2010.
- 9 Chandra Chekuri and Vasilis Livanos. On submodular prophet inequalities and correlation gap. In Ioannis Caragiannis and Kristoffer Arnsfelt Hansen, editors, *Algorithmic Game Theory - 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21-24, 2021, Proceedings*, volume 12885 of *Lecture Notes in Computer Science*, page 410. Springer, 2021. URL: <https://link.springer.com/content/pdf/bbm%3A978-3-030-85947-3%2F1.pdf>.
- 10 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding for matroid polytopes and applications. *arXiv preprint arXiv:0909.4348*, 2009.
- 11 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 575–584, 2010.
- 12 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 43(6):1831–1879, 2014.
- 13 Tasos Christofides and Eutichia Vaggelatou. A connection between supermodular ordering and positive/negative association. *Journal of Multivariate Analysis*, 88:138–151, January 2004. doi:10.1016/S0047-259X(03)00064-2.
- 14 Shaddin Dughmi. The outer limits of contention resolution on matroids and connections to the secretary problem. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 42:1–42:18, 2020.
- 15 Shaddin Dughmi. Matroid secretary is equivalent to contention resolution. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, pages 58:1–58:23, 2022.



- 16 Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond  $1/e$ . In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 248–257, 2016.
- 17 Hossein Esfandiari, Amin Karbasi, and Vahab S. Mirrokni. Adaptivity in adaptive submodularity. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 1823–1846. PMLR, 2021. URL: <http://proceedings.mlr.press/v134/esfandiari21a.html>.
- 18 Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 570–579, 2011.
- 19 Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1014–1033. SIAM, 2016. doi:10.1137/1.9781611974331.ch72.
- 20 Satoru Fujishige. *Submodular functions and optimization*. Elsevier, 2005.
- 21 Kevin Garbe and Jan Vondrak. Concentration of lipschitz functions of negatively dependent variables. *arXiv preprint arXiv:1804.10084*, 2018.
- 22 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, pages 205–216, 2013.
- 23 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1731–1747, 2016.
- 24 Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 58–65, 2007.
- 25 Mark Jerrum. Two remarks concerning balanced matroids. *Comb.*, 26(6):733–742, 2006. doi:10.1007/s00493-006-0039-5.
- 26 Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, 11(1):286–295, 1983. doi:10.1214/aos/1176346079.
- 27 Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 123–136, 2012.
- 28 Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bull. Am. Math. Soc.*, 1977.
- 29 Ulrich Krengel and Louis Sucheston. On semiamarts, amarts, and processes with finite value. *Advances in Prob.*, 4:197–266, 1978.
- 30 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.
- 31 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- 32 Robin Pemantle. Towards a theory of negative dependence. *Journal of Mathematical Physics*, 41(3):1371–1390, 2000. doi:10.1063/1.533200.
- 33 Yuval Peres, Mohit Singh, and Nisheeth K. Vishnoi. Random walks in polytopes and negative dependence. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 50:1–50:10, 2017.
- 34 Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 324–332, 2016.

- 35 Aviad Rubinstein and Sahil Singla. Combinatorial prophet inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1671–1687, 2017.
- 36 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
- 37 Qi-Man Shao. A comparison theorem on moment inequalities between negatively associated and independent random variables. *Journal of Theoretical Probability*, 13(2):343–356, 2000.
- 38 Jan Vondrák. *Submodularity in combinatorial optimization*. PhD thesis, Univerzita Karlova, Matematicko-fyzikální fakulta, 2007.
- 39 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.

## A

 Missing Proofs

### A.1 Weak Negative Regression Proofs

► **Proposition 13.** *NA and NR imply WNR, and WNR implies NCD, but the reverse implications do not hold. In other words, the union of NA and NR distributions is a strict subset of WNR distributions, which is a strict subset of NCD distributions.*

**Proof.** We proved in Section 2 that the union of NA and NR distributions is a subset of WNR, and that WNR distributions are a subset of NCD distributions. Appendix A.2 provides example distributions demonstrating strict containment of WNR in NCD.

To show strict containment of NA and NR in WNR, we consider the following distribution  $\mathcal{D}$ , which was given by Joag-Dev and Proschan [26].<sup>9</sup> We treat  $\mathcal{D}$  as a distribution over Bernoulli random variables  $\mathbf{X} = (X_1, X_2, X_3, X_4)$  to simplify notation.

■ **Table 2** A distribution which is WNR, but not NA or NR.

| $\mathcal{D}$ | $(X_1, X_2)$ |        |        |        |        |          |
|---------------|--------------|--------|--------|--------|--------|----------|
| $(X_3, X_4)$  |              | (0, 0) | (0, 1) | (1, 0) | (1, 1) | Marginal |
|               | (0, 0)       | 0.0577 | 0.0623 | 0.0623 | 0.0577 | 0.24     |
|               | (0, 1)       | 0.0623 | 0.0677 | 0.0677 | 0.0623 | 0.26     |
|               | (1, 0)       | 0.0623 | 0.0677 | 0.0677 | 0.0623 | 0.26     |
|               | (1, 1)       | 0.0577 | 0.0623 | 0.0623 | 0.0577 | 0.24     |
|               | Marginal     | 0.24   | 0.26   | 0.26   | 0.24   |          |

$\mathcal{D}$  violates NA because  $\text{Cov}_{\mathbf{X} \sim \mathcal{D}}[X_1 X_2, X_3 X_4] > 0$ , and  $\mathcal{D}$  violates NR because the conditional expectation  $\mathbb{E}_{\mathbf{X} \sim \mathcal{D}}[X_1 X_2 \mid X_3 = 1, X_4]$  is increasing in  $X_4$ .

By observing that the value in column 2 is larger than in column 4 for any row of Table 2, we see that for any  $x_3, x_4 \in \{0, 1\}$ , the conditional expectation  $\mathbb{E}_{\mathbf{X} \sim \mathcal{D}}[X_2 \mid X_3 = x_3, X_4 = x_4, X_1]$  is decreasing in  $X_1$ . Therefore, we can convert the distribution  $\mathcal{D}$  conditioned on  $X_1 = 0$  into the distribution  $\mathcal{D}$  conditioned on  $X_1 = 1$  by only transferring probability mass “downwards,” which cannot increase the expectation of a monotone function  $f : \{0, 1\}^4 \rightarrow \mathbb{R}$ . Thus, for any such function which does not depend on  $X_1$ ,

<sup>9</sup> They studied NA distributions over continuous random variables, and gave this distribution as an example that Negative Orthant Dependence (this is equivalent to NCD for Bernoulli random variables) does not imply NA.

$$\mathbb{E}_{\mathbf{X} \sim \mathcal{D}}[f(\mathbf{X}) \mid X_1 = 1] \leq \mathbb{E}_{\mathbf{X} \sim \mathcal{D}}[f(\mathbf{X}) \mid X_1 = 0] ,$$

which is the WNR condition. Since  $X_1, X_2$  and  $(X_1, X_2), (X_3, X_4)$  are both exchangeable, we can repeat this analysis for all  $X_i$ . Thus,  $\mathcal{D}$  is WNR, but neither NA nor NR. ◀

► **Proposition 16.** *WNR is closed both under projection and under products.*

**Proof.** Closure under projection follows trivially because the WNR condition (1) is satisfied for all monotone functions, and monotone functions restricted to a subset of elements are still monotone.

For closure under products, let  $\mathcal{A}$  and  $\mathcal{B}$  be WNR distributions over  $2^A$  and  $2^B$  for disjoint  $A, B$ , and let  $\mathcal{D}$  be their product. WLOG, fix  $i \in A$  and a monotone function  $f : 2^{A \cup B} \rightarrow \mathbb{R}$ . Since  $\mathcal{A}$  and  $\mathcal{B}$  are independent,  $\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid S \cap B = T] = \mathbb{E}_{S \sim \mathcal{A}}[f((S \setminus i) \cup T)]$ . Since  $\mathcal{A}$  is WNR and  $f(S \cup T)$  is still a monotone function,

$$\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid S \cap B = T, i \in S] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid S \cap B = T, i \notin S] .$$

Taking expectations over  $S \cap B$  gives  $\mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid i \in S] \leq \mathbb{E}_{S \sim \mathcal{D}}[f(S \setminus i) \mid i \notin S]$ , completing the proof. ◀

## A.2 Submodular Dominance Example Distributions

► **Proposition 19.** *The distribution  $\mathcal{D}$  which samples uniformly from  $\emptyset, \{1\}, \{2\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$  satisfies Submodular Dominance, but  $\mathcal{D}$  violates WNR for  $f(S) := \max(\mathbb{1}_{1 \in S}, \mathbb{1}_{2 \in S})$  and  $i = 3$ .*

**Proof.** Using the definition of  $\mathcal{D}_k$  from the proof of Theorem 3, notice that  $\mathcal{D}_1$  is a product distribution. Therefore, we only need the analysis in Claim 18 to follow, which only requires that the WNR condition (1) holds for  $i = 1$ .  $\mathcal{D}$  conditioned on  $1 \in S$  samples  $\emptyset, \{2\}$ , and  $\{3\}$  w.p.  $1/3$ , and  $\mathcal{D}$  conditioned on  $1 \notin S$  samples  $\emptyset, \{2\}$ , and  $\{2, 3\}$  w.p.  $1/3$ , which cannot obtain lower expectation for a monotone function, so the WNR condition holds for  $i = 1$  and  $\mathcal{D}$  satisfies Submodular Dominance.

$\mathcal{D}$  violates WNR for  $f$  and  $i = 3$  because  $\mathcal{D}$  conditioned on  $3 \in S$  always samples either 1 or 2, whereas  $\mathcal{D}$  conditioned on  $3 \notin S$  can sample  $\emptyset$ . Thus, there exist distributions which satisfy Submodular Dominance but violate WNR. ◀

► **Proposition 20.** *The distribution  $\mathcal{D}$  over  $2^{[4]}$  which chooses uniformly at random  $i \in [4]$ , then returns w.p.  $1/2$  either  $i$  or  $[4] \setminus i$ , is NCD. However,  $\mathcal{D}$  violates Submodular Dominance for the submodular function  $f(S) := \min(2, |S|)$ .*

**Proof.** Notice that  $\mathcal{D}$  is identical under permutations of elements. Further, because  $i$  or  $[4] \setminus i$  is returned with equal probability, we have the property that for any  $T \subseteq [4]$ ,

$$\Pr_{S \sim \mathcal{D}}[T \subseteq S] = \Pr_{S \sim \mathcal{D}}[T \subseteq S^c] .$$

Therefore, it is sufficient to show that  $\Pr_{S \sim \mathcal{D}}[1, 2 \in S] \leq 1/4$ ,  $\Pr_{S \sim \mathcal{D}}[1, 2, 3 \in S] \leq 1/8$ , and  $\Pr_{S \sim \mathcal{D}}[1, 2, 3, 4 \in S] \leq 1/16$  to prove  $\mathcal{D}$  is NCD.

- For  $1, 2 \in S$ , we need to choose  $i = 3, 4$ , then return  $[4] \setminus i$ . This occurs w.p.  $1/2 \cdot 1/2 = 1/4$ .
- For  $1, 2, 3 \in S$ , we need to choose  $i = 4$ , then return  $[4] \setminus i$ . This occurs w.p.  $1/4 \cdot 1/2 = 1/8$ .
- There is no way for  $1, 2, 3, 4 \in S$ .

Thus,  $\mathcal{D}$  is NCD.  $f$  is a matroid rank function, so it is submodular. Letting  $\mathbf{x}$  be the marginals of  $\mathcal{D}$ , a simple expected value computation shows that  $\mathbb{E}_{S \sim \mathcal{D}}[f(S)] = 12/8 < 13/8 = F(\mathbf{x})$ , so  $\mathcal{D}$  violates Submodular Dominance.  $\blacktriangleleft$

► **Proposition 21.** *The distribution  $\mathcal{D}$  over  $2^{[8]}$  which chooses uniformly at random  $i \in A := \{1, 2, 3, 4\}$  and  $j \in B := \{5, 6, 7, 8\}$ , then returns w.p.  $1/2$  either  $i \cup (B \setminus j)$  or  $(A \setminus i) \cup j$ , is NCD. However,  $\mathcal{D}$  violates Submodular Dominance for the submodular function  $f(S) := \min(2, |S \cap A|)$ .*

**Proof.** This example extends the previous example to a homogeneous distribution. Similar to the previous example,  $\mathcal{D}$  is closed under permutations of  $A$ , permutations of  $B$ , and swaps of  $A$  with  $B$ . Since we already showed any  $T \subseteq A$  satisfies the NCD condition, it is sufficient to show that  $\Pr_{S \sim \mathcal{D}}[1, 5 \in S] \leq 1/4$ ,  $\Pr_{S \sim \mathcal{D}}[1, 2, 5 \in S] \leq 1/8$ ,  $\Pr_{S \sim \mathcal{D}}[1, 2, 5, 6 \in S] \leq 1/16$ , and  $\Pr_{S \sim \mathcal{D}}[1, 2, 3, 5 \in S] \leq 1/16$  (since only sets of size 4 are drawn, if  $|T| > 4$  it automatically satisfies the NCD condition).

- For  $1, 5 \in S$ , we need to choose  $i = 1$  and  $j = 6, 7, 8$ , then return  $i \cup (B \setminus j)$ , or choose  $i = 2, 3, 4$  and  $j = 5$ , then return  $(A \setminus i) \cup j$ . This occurs w.p.  $2 \cdot 1/4 \cdot 3/4 \cdot 1/2 = 3/16 \leq 1/4$ .
- For  $1, 2, 5 \in S$ , we need to choose  $i = 3, 4$  and  $j = 5$ , then return  $(A \setminus i) \cup j$ . This occurs w.p.  $1/2 \cdot 1/4 \cdot 1/2 = 1/16 \leq 1/8$ .
- There is no way for  $1, 2, 5, 6 \in S$ .
- For  $1, 2, 3, 5 \in S$ , we need to choose  $i = 4$  and  $j = 5$ , then return  $(A \setminus i) \cup j$ . This occurs w.p.  $1/4 \cdot 1/4 \cdot 1/2 = 1/32 \leq 1/16$ .

Thus,  $\mathcal{D}$  is NCD, and we again have  $\mathbb{E}_{S \sim \mathcal{D}}[f(S)] = 12/8 < 13/8 = \mathbb{E}_{S \sim \mathbf{x}}[f(S)]$ , so  $\mathcal{D}$  violates Submodular Dominance.  $\blacktriangleleft$

### A.3 Product of Singletons is a Convex Combination of Product Distributions

► **Lemma 27.** *Let  $\mathcal{D}$  be a product of singletons distribution over  $2^E$  with marginals  $\mathbf{x} \in [0, 1]^{nm}$ . Let  $x_i := \sum_j x_{ij}$ , and let  $\vec{x} := (x_i : i \in U)$ . For any  $\mathbf{u} \in [m]^n$ , let  $E_{\mathbf{u}} := \{iu_i : i \in U\}$  and let  $\mathcal{D}_{\mathbf{u}}$  be a product distribution over  $2^{E_{\mathbf{u}}}$  with marginals  $\vec{x}$ . Then for any  $g : 2^E \rightarrow \mathbb{R}$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}}[g(S)] = \sum_{\mathbf{u} \in [m]^n} \left( \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [g(S)] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right). \quad (4)$$

**Proof.** For some weights  $p_S$ , we can rewrite RHS of (4) as

$$\sum_{S \subseteq E} g(S) \cdot p_S.$$

Our approach is to show that  $p_T = \Pr_{S \sim \mathcal{D}}[S = T]$  for any  $T \subseteq E$ . Then the summation is simply the expectation of  $g$  over  $\mathcal{D}$  and we are finished.

Fix some set  $T \subseteq E$ . The distributions for which  $T$  is in the image of  $\mathcal{D}_{\mathbf{u}}$  are those where for all  $ij \in T$ ,  $u_i = j$ . Therefore,

$$p_T = \sum_{\substack{\mathbf{u} \in [m]^n \\ u_i = j \forall ij \in T}} \left( \Pr_{S \sim \mathcal{D}_{\mathbf{u}}} [S = T] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right).$$

Let  $T^* \subseteq U$  be a set where  $i \in T^*$  if there exists some  $j$  for which  $ij \in T$ . Then,

$$p_T = \sum_{\substack{\mathbf{u} \in [m]^n \\ u_i = j \forall ij \in T}} \left( \prod_{i \in T^*} x_i \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{i \in T^*} \frac{x_{iu_i}}{x_i} \prod_{i \notin T^*} \frac{x_{iu_i}}{x_i} \right).$$

We combine the products over  $i \in T^*$ , and move the first product over  $i \notin T^*$  outside the summation as the inner term does not depend on  $\mathbf{u}$ .

$$p_T = \prod_{i \notin T^*} (1 - x_i) \cdot \sum_{\substack{\mathbf{u} \in [m]^n \\ u_i = j \forall ij \in T}} \left( \prod_{i \in T^*} x_{iu_i} \prod_{i \notin T^*} \frac{x_{iu_i}}{x_i} \right).$$

Because the summation is restricted to  $\mathbf{u}$  where  $u_i = j$  for all  $ij \in T$ , the coordinates of  $\mathbf{u}$  for  $i \in T^*$  can only take one value. Thus, the product over  $i \in T^*$  is always the same, and can be factored out of the summation.

$$p_T = \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{ij \in T} x_{ij} \cdot \sum_{\substack{\mathbf{u} \in [m]^n \\ u_i = j \forall ij \in T}} \left( \prod_{i \notin T^*} \frac{x_{iu_i}}{x_i} \right).$$

As we just observed, the summation only enforces a condition on  $i \in T^*$ , so we sum up over all possible  $u_i \in [m]$  for  $i \notin T^*$ . We can rewrite this as

$$\begin{aligned} p_T &= \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{ij \in T} x_{ij} \cdot \prod_{i \notin T^*} \left( \sum_{j \in [m]} \frac{x_{ij}}{x_i} \right) \\ &= \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{ij \in T} x_{ij} \cdot \prod_{i \notin T^*} \left( \frac{1}{x_i} \cdot \sum_{j \in [m]} x_{ij} \right) \\ &= \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{ij \in T} x_{ij} \cdot \prod_{i \notin T^*} \left( \frac{1}{x_i} \cdot x_i \right) \\ &= \prod_{i \notin T^*} (1 - x_i) \cdot \prod_{ij \in T} x_{ij} \\ &= \Pr_{S \sim \mathcal{D}} [S = T]. \end{aligned}$$

Since these computations follow for any  $T \subseteq E$ , we have

$$\mathbb{E}_{S \sim \mathcal{D}} [g(S)] = \sum_{S \subseteq E} g(S) \cdot p_S = \sum_{\mathbf{u} \in [m]^n} \left( \mathbb{E}_{S \sim \mathcal{D}_{\mathbf{u}}} [g(S)] \cdot \prod_{i \in U} \frac{x_{iu_i}}{x_i} \right),$$

which completes the proof. ◀



# Online Facility Location with Linear Delay

Marcin Bienkowski 

Institute of Computer Science, University of Wrocław, Poland

Martin Böhm 

Institute of Computer Science, University of Wrocław, Poland

Jarosław Byrka 

Institute of Computer Science, University of Wrocław, Poland

Jan Marcinkowski 

Institute of Computer Science, University of Wrocław, Poland

---

## Abstract

In the problem of online facility location with delay, a sequence of  $n$  clients appear in the metric space, and they need to be eventually connected to some open facility. The clients do not have to be connected immediately, but such a choice comes with a certain penalty: each client incurs a waiting cost (equal to the difference between its arrival and its connection time). At any point in time, an algorithm may decide to open a facility and connect any subset of clients to it. That is, an algorithm needs to balance three types of costs: cost of opening facilities, costs of connecting clients, and the waiting costs of clients. We study a natural variant of this problem, where clients may be connected also to an *already open* facility, but such action incurs an extra cost: an algorithm pays for waiting of the facility (a cost incurred separately for each such “late” connection). This is reminiscent of online matching with delays, where both sides of the connection incur a waiting cost. We call this variant *two-sided delay* to differentiate it from the previously studied *one-sided delay*, where clients may connect to a facility only at its opening time.

We present an  $O(1)$ -competitive deterministic algorithm for the two-sided delay variant. Our approach is an extension of the approach used by Jain, Mahdian and Saberi [STOC 2002] for analyzing the performance of *offline* algorithms for facility location. To this end, we substantially simplify the part of the original argument in which a bound on the sequence of factor-revealing LPs is derived. We then show how to transform our  $O(1)$ -competitive algorithm for the two-sided delay variant to  $O(\log n / \log \log n)$ -competitive deterministic algorithm for one-sided delays. This improves the known  $O(\log n)$  bound by Azar and Touitou [FOCS 2020]. We note that all previous online algorithms for problems with delays in general metrics have at least logarithmic ratios.

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** online facility location, network design problems, facility location with delay, JMS algorithm, competitive analysis, factor revealing LP

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.45

**Category** APPROX

**Supplementary Material** *Software*: <https://github.com/bohm/fl-double-sided-waiting>

**Funding** *Marcin Bienkowski*: Supported by Polish National Science Centre grant 2016/22/E/ST6/00499.

*Jarosław Byrka*: Supported by Polish National Science Centre grant 2020/39/B/ST6/01641.

## 1 Introduction

The facility location problem [1] is one of the best-known examples of network design problems, extensively studied both in operations research and in computer science. The problem is defined in a metric space  $\mathcal{X}$ . An algorithm is given a set of  $n$  clients and its goal is to open



© Marcin Bienkowski, Martin Böhm, Jarosław Byrka, and Jan Marcinkowski;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 45; pp. 45:1–45:17



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



a set of facilities (chosen points of  $\mathcal{X}$ ) minimizing the total cost, defined as the sum of costs of opening facilities plus the costs of connecting clients. Our focus is on the *non-uniform* case, where the opening cost of a facility may depend on its position in  $\mathcal{X}$ . The connection cost of a given client is simply its distance to the nearest open facility. This simple statement hides a surprisingly rich combinatorial structure and gave rise to series of algorithms and extensions. In particular, the problem is NP-complete and APX-hard [31] and its approximation ratio has been studied in a long sequence of improvements [41, 36, 34, 33, 25, 23, 39, 21], with the current record of 1.488 proved by Li [37].

In the online scenario, the set of clients is not known up-front, but it is revealed to an (online) algorithm one element at a time. Once a client becomes known, an algorithm has to make an irrevocable and immediate decision whether to open additional facilities and to which facility the current client should be connected.<sup>1</sup> As in the offline scenario, the algorithm is compared to the best *offline* solution, and in online scenarios, we use the name *competitive ratio* instead of approximation ratio. This scenario has been fully resolved: tight asymptotic lower and upper bounds of  $\Theta(\log n / \log \log n)$  are known both for randomized and deterministic algorithms [40, 2, 29, 30].

In the last few years, many online problems have been considered in scenarios *with delays*. In the case of online facility location, first studied by Azar and Touitou [9], the clients arrive in time, and while each of them has to be connected eventually, such action does not have to be executed immediately. This additional degree of freedom comes, however, with a price: each waiting client incurs an extra cost that (in the basic setting studied in this paper) is equal to its total waiting time (time between its arrival and its connection). We note that in terms of achievable competitive ratios, the classic models and models with delays are rarely comparable as the possibility of delaying actions is allowed also in the benchmark offline solution.

### Facility location with one-sided delay

This variant has been introduced and studied in [9, 10]. There each facility is *ephemeral*: it is opened only momentarily at time  $t$  chosen by an algorithm, and all connections to this facility must be made at time  $t$ . The algorithm can open another facility at the same location at a different time  $t'$ , but the opening cost must be paid again. The waiting costs of clients are as described above.

The best known algorithm for this problem variant is  $O(\log n)$ -competitive [10]. (Interestingly, it is not known whether this particular variant admits constant-factor approximation in the offline setting when all client arrivals are known up-front.)

### Facility location with two-sided delay

We propose the following slight deviation from the one-sided variant described above: once a facility becomes open at time  $t$ , it remains open forever and can be connected to in the future. However, any client that connects to such facility at time  $t' > t$  needs to pay an *additional* waiting cost of  $t' - t$ . We call this amount *facility-side waiting cost*, which needs to be paid on top of the “standard” (*client-side*) *waiting cost*. We emphasize that such connections, dubbed *late connections*, can be made both by clients that arrived before facility opening and also after this time. Similarly to the one-sided delay model, we allow opening of multiple facilities in the same location at different points in time.

---

<sup>1</sup> The best option is clearly to connect a given client to the closest facility, but some naturally defined algorithms may not have this property.

The two-sided model can be seen as an approximation of the *early-late adopter behavior* in crowdfunding models. In crowdfunding platforms for technology products [42] such as Kickstarter, any specific project is started by gathering initial contributions by enthusiasts up to a certain threshold, which should (in an ideal case) imply opening of a production line for the specified technology product. Contributors in this pre-production phase are called *early adopters*.

As we move forward in time, while the production may already begin, it may still be possible for so-called *late adopters* to join the crowdfunding project on the Kickstarter website and receive the final product. As early adoption is more beneficial for the producer of the technology product, late adoption is sometimes penalized with an increased cost of the same product compared to the early adoption.

In the framework of crowdfunding models, we can see online facility location with two-sided delay as facility location in an early-late adopter setting, where a technology product can be manufactured at multiple factories and late adopters may join into the crowdfunding scheme and thus contribute towards offsetting the cost of the production while it is in progress. However, the clients who join late need to deal with the missed-opportunity cost (the client-side cost) as well as the late adopter increase in price (the facility-side waiting cost).

## 1.1 Our Results and Techniques

Our first positive contribution is showing that facility location with two-sided delay admits a constant-competitive online algorithm, which is an important open problem for the one-sided case. Namely, we show:

► **Theorem 1.** *There exists an 3.869-competitive deterministic algorithm for the online facility location problem with two-sided delay, where all waiting costs are equal to the waiting times.*

We analyze a natural greedy algorithm, which grows budgets with increasing waiting delays and opens facilities for subsets of clients once sums of these budgets reach certain thresholds. To analyze this algorithm, we use dual fitting methods. Our analysis is a substantial extension of the approach used by Jain et al. [34, 33] for analyzing the performance of *offline* algorithms for (non-delayed) facility location.

The central part of the analysis is a linear program (LP), parameterized by an integer  $k$ , whose objective value is an upper bound on the competitive ratio of our algorithm *provided the number of clients in the input is at most  $k$* . As the objective function of this LP grows with  $k$ , simply solving the LP would yield the correct upper bound only for instances of limited size. As a replacement for the technical original argument in [33] we propose a much more intuitive one that is based on an upper bound to this sequence by the value of a finite linear program.

We would like to stress that we see our novel approach to the competitive analysis of facility location dual-fitting LP as an important contribution of this paper to the area of facility location with delays. Our approach has a significant computer-assisted component (Section 4) and it can thus be quickly deployed to give provable estimates on the potential competitive ratio or an approximation ratio of factor-revealing linear programs in other settings. Our code for the algorithmic part is publicly available [13].

Our second result is showing that  $O(1)$ -competitive algorithm for the two-sided variant yields an improved guarantee also for the one-sided variant.

► **Theorem 2.** *There exists an  $O(\log n / \log \log n)$ -competitive deterministic algorithm for the online facility location problem with one-sided delay, where all waiting costs are equal to the waiting times.*

We prove this result via a reduction that can be applied to any algorithm solving the two-sided variant provided it satisfies a certain technical condition that we call *sensibility*. Informally speaking this property means that the waiting costs associated with late connections are not very large; we defer the precise definition to Section 3.

Theorem 2 improves the known  $O(\log n)$ -bound by Azar and Touitou [10]. While the improvement is small, we note that all previous online algorithms for problems with delays have at least logarithmic ratios (in the number of used points of the metric space), so ours is the first to break this natural barrier.

## 1.2 Related work

Recently many online graph problems have been considered in a variant that allows requests to be delayed. Apart from the facility location problem studied in this paper, examples include the Steiner tree problem [9, 10], multi-level aggregation [15, 24, 19, 9, 11, 12], Steiner forest/network [10], directed Steiner tree [10], multi-cut [10], online matching [27, 3, 4, 17, 16, 38, 28, 6, 8], set cover [22, 5] and  $k$ -server (known in this setting as online service with delay) [7, 18, 9].

That said, the concept of delaying requests itself is not new. Famous studied problems include the TCP acknowledgement problem [26, 35] and joint replenishment problem [20, 14] (that are equivalent to the recently studied multi-level aggregation problem on one-level or two-level trees).

### General waiting costs

The waiting costs considered in this paper are equal to waiting times. Another studied case are deadlines, where waiting costs are zero till a request-specific time (the deadline) and infinite afterwards. For some problems, easier algorithms or better bounds are known when the waiting costs are in the deadline form (see, e.g., [14, 19]).

Many of the results listed above can be extended to waiting costs being arbitrary non-decreasing left-continuous functions of waiting times. These extensions are straightforward if an algorithm is defined by simple thresholds on (sums of) waiting costs; when these thresholds are reached, they trigger an appropriate action of the algorithm. For instance, algorithms for the TCP acknowledgement problem were constructed for linear waiting costs, but they can be trivially extended to general costs.

There are however a few cases where general waiting costs are more problematic. Most notably, the online matching problem was studied for linear waiting costs [27, 3, 4, 17, 16, 28, 6], then shown to be more difficult (in terms of achievable competitive ratios) for convex waiting costs [38], and only recently competitive algorithms were shown for concave waiting costs [8].

The algorithms presented in this paper also fall into the latter category: our LP-based analysis heavily depends on the linearity of the waiting functions, and thus our algorithms cannot be easily extended to general waiting costs. (We note that the  $O(\log n)$ -competitive algorithm by Azar and Touitou [10] can handle arbitrary waiting costs.)

### 1.3 Remark about Facility-Side Waiting Costs

Recall that in the two-sided variant that we study in our paper, we assume that each late connection incurs an additional waiting cost at the facility side (equal to the time that passes between facility opening and client connection). Below, we argue that setting this cost to zero would cause the optimal competitive ratio to be  $\Theta(\log n / \log \log n)$ . This shows that for the  $O(1)$ -competitive result of Theorem 1, some assumptions about waiting cost functions are necessary.

► **Observation 3.** *In the two-sided variant of the facility location problem with delays, setting facility-side waiting costs to zero causes the optimal competitive ratio (both deterministic and randomized) to become asymptotically equal to  $\Theta(\log n / \log \log n)$ .*

**Proof.** Assume no penalty for facility-side waiting. For any input instance, without an increase of the cost, OPT may open all its facilities at time 0 and connect all clients immediately when they arrive. Thus, the optimal solution incurs no waiting cost at all.

As the waiting cost can be avoided also for an online algorithm (by serving all clients immediately upon their arrival), the desired competitive ratio can be attained by running an  $O(\log n / \log \log n)$ -competitive algorithm (deterministic or randomized) for the online facility location problem (without delays) [40, 30],

For showing a lower bound, we may simply use an adversarial strategy for the online facility location problem (without delays) [30]; however, now the next request is presented only after an algorithm serves the previous one. This way, waiting becomes useless for an online algorithm, and the lower bound of  $\Omega(\log n / \log \log n)$  [30] applies also for the waiting model. ◀

### 1.4 Preliminaries

We use the following notions throughout the paper.

The pair  $(\mathcal{X}, \text{dist})$  denotes the underlying metric space with its distance function, and  $\mathcal{Y} \subseteq \mathcal{X}$  denotes the positions of potential facilities. The function  $\text{open} : \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$  defines the cost of opening a facility at a specific location.

The clients are numbered from 1 to  $n$  and arrive in time; each is associated with a point in  $\mathcal{X}$ . Their total number is not known up-front to an online algorithm. For a client  $j$ , we use  $x_j$  to denote its position and  $t_j$  to denote the time of its arrival. We say that a client is *active* from the time of its arrival until it gets connected to a facility by an online algorithm, and it is *inactive* after the connection.

At any time, an algorithm may open a facility at any point  $y \in \mathcal{Y}$ , paying *opening cost*  $\text{open}(y)$ . An active client  $j$  may be connected by an algorithm at time  $t_j^c \geq t_j$  to a facility that was open at time  $\tau$ , such that

- $\tau = t_j^c$ , for the one-sided delay variant;
- $\tau \leq t_j^c$ , for the two-sided delay variant.

Such a connection incurs a *connection cost*  $\text{dist}(x_j, y)$  and two types of waiting costs: a *client-side waiting cost*  $t_j^c - t_j$  and a *facility-side waiting cost*  $t_j^c - \tau$ . Note that the latter waiting cost is always zero for the one-sided delay variant.

The goal of an algorithm is to minimize the total cost, defined as the sum of opening costs, connection costs, and waiting costs.

## 2 Algorithm for the Two-Sided Variant

We will now describe the algorithm for the online facility location with two-sided linear waiting cost. Our algorithm is parameterized with a constant  $\gamma > 1$  that will be fixed later. We say that an active client  $j$  at time  $t \geq t_j$ , after experiencing waiting cost of  $t - t_j$ , has a *connectivity budget*

$$\alpha_j(t) = \gamma \cdot (t - t_j).$$

In the analysis, we use  $\alpha_j$  to denote the connectivity budget of client  $j$  at time  $t_j^c$ , when it becomes connected to a facility.

At any time  $t$ , for each potential facility location  $y \in \mathcal{Y}$ , an active client  $j$  has an offer of a contribution towards the opening of a facility at  $y$  equal to  $\beta_j^t(y) = \max\{0, \alpha_j(t) - \text{dist}(x_j, y)\}$ . When client  $j$  becomes connected (and inactive), it no longer offers any contribution.

The algorithm follows the natural continuous flow of time of the online sequence and reacts to the events occurring throughout its runtime.

► **Algorithm 1.** *At any time  $t$ , do the following.*

- (a) *If a new client  $j$  arrives at point  $x_j$  (client  $j$  becomes active): From this time onward start growing its connectivity budget.*
- (b) *If for any location  $y \in \mathcal{Y}$ , the sum of offered contributions  $\beta_j^t(y)$  towards this location from all the currently active clients reaches  $\text{open}(y)$  (the facility opening cost at  $y$ ): Let  $A^t(y)$  be the set of active clients  $j$  for which  $\alpha_j(t) \geq \text{dist}(x_j, y)$ , i.e., those that can afford the distance. Open a facility at  $y$ , and connect every client  $j \in A^t(y)$  to it.*
- (c) *If for a facility that has already been open at time  $\tau \leq t$  at location  $y$ , and for an active client  $j$ , it holds that  $t - \tau = \alpha_j(t) - \text{dist}(x_j, y)$ : Connect client  $j$  to this facility. We call this action late connection.*

*In Case b and Case c, all clients that get connected become inactive.*

### 2.1 Basic observations

One may observe that Algorithm 1 is a generalization of (the simpler version of) the algorithm by Jain et al. [34]. Furthermore, if it is run on an instance where all clients appear at the same time, it produces the same solution (the same facility locations and the same connections) as the original offline approximation algorithm.

It is convenient to think that the connectivity budget of a client is first spent on connection cost, and the remaining part either contributes to the opening of a new facility or pays for the facility-side waiting cost of an already open facility.

► **Observation 4.** *The total cost of the solution produced by Algorithm 1 is  $(1 + 1/\gamma) \cdot \sum_j \alpha_j$ .*

**Proof.** The sum of opening costs, connection costs and facility-side waiting costs in the produced solution equals the sum of final connectivity budgets  $\sum_j \alpha_j$ . The total client-side waiting cost is  $(1/\gamma) \cdot \sum_j \alpha_j$ . ◀

To estimate the competitive ratio of the algorithm, it thus suffices to compare the cost of the optimal solution to  $(1 + 1/\gamma) \cdot \sum_j \alpha_j$ , which we do in Section 3.

## 2.2 Sensibility

Our later construction for the one-sided waiting variant requires that the used online algorithm for the two-sided variant has the following property, which bounds the number of clients that connect late to an already open facility.

► **Definition 5** (sensibility). *Fix any  $\lambda > 1$  and  $\xi > 1$ . An algorithm solving the two-sided variant is called  $(\lambda, \xi)$ -sensible if it satisfies the following property for any facility  $f$  opened at time  $\tau$  and location  $y$ : for any  $w > 0$ , the number of clients connected to  $f$  within the interval  $(\tau + w, \tau + \lambda \cdot w]$  is at most  $\xi \cdot \text{open}(y)/w$ .*

We show that for some constants  $\lambda$  and  $\xi$ , our algorithm is  $(\lambda, \xi)$ -sensible: if clients connecting late to an open facility incurred large (facility-side) waiting costs, then our algorithm would rather create a new copy of this facility, and connect these clients to the copy.

► **Lemma 6.** *Fix a parameter  $\gamma > 1$  of Algorithm 1. For any  $\lambda \in (1, 1 + 1/(\gamma - 1))$ , Algorithm 1 is  $(\lambda, (\gamma - (\gamma - 1) \cdot \lambda)^{-1})$ -sensible.*

**Proof.** Fix a facility  $f$  opened at time  $\tau$  at location  $y$ . Let  $C_w$  be the set of clients that are connected late to  $f$  within interval  $(\tau + w, \tau + \lambda \cdot w]$ . Take any client  $j \in C_w$  and let  $t_j^c = \tau + h$  be its connection time. We define the *residual budget* of client  $j$  at time  $t$  as  $r_j(t) = \alpha_j(t) - \text{dist}(x_j, y)$ .

We now estimate the residual budget of client  $j$  at time  $\tau + w$ . Let  $\tau + g$  be the time when its residual budget becomes zero, i.e.,  $r_j(\tau + g) = 0$ . Its residual budget at time  $\tau + h$  is then  $r_j(\tau + h) = \alpha_j(\tau + h) - \alpha_j(\tau + g) = \gamma \cdot (h - g)$ . On the other hand,  $r_j(\tau + h) = (\tau + h) - \tau = h$ , as  $j$  forms a late connection to  $f$  at time  $\tau + h$ . Hence,  $\gamma \cdot g = (\gamma - 1) \cdot h \leq (\gamma - 1) \cdot \lambda \cdot w$ , which implies  $r_j(\tau + w) = \gamma \cdot (w - g) \geq (\gamma - (\gamma - 1) \cdot \lambda) \cdot w$ . (Note that for our choice of  $\lambda$ , this amount is positive.)

By the definition of Algorithm 1, the residual budgets are spent either on opening new facilities or on facility-side waiting of an already opened facility. Hence, at time  $\tau + w$ , the sum of residual budgets of all clients from  $C_w$  cannot be larger than  $\text{open}(y)$ , as in such case Algorithm 1 would open another copy of the facility at  $y$ . This argument implies that  $|C_w| \cdot (\gamma - (\gamma - 1) \cdot \lambda) \cdot w \leq \text{open}(y)$ , which concludes the proof. ◀

For example, by Lemma 6, Algorithm 1 with  $\gamma = 2$  is  $(3/2, 2)$ -sensible.

## 3 Competitive Analysis via Factor Revealing LP

Fix an optimal offline solution. We will heavily use the structure of this solution being a collection of stars, each of them composed of a single open facility and a set of clients connected in the optimal solution to this facility. Just as in the analysis of the JMS algorithm [34, 33], we will focus on a single star  $S$  of OPT, and compare  $\sum_{j \in S} \alpha_j$  to the cost of this star.

Let  $\tau$  be the time of opening the facility  $f$  in the considered star  $S$  of the optimal solution. Note that our online algorithm could connect clients  $j \in S$  to facilities opened by the online algorithm both before and after  $\tau$ . For a client  $j \in S$  that arrived at time  $t_j$  and got connected by the algorithm at time  $t_j^c$ , we set  $a_j = t_j - \tau$  and  $s_j = t_j^c - \tau$  to denote the arrival time and the *service time* of  $j$  (time when  $j$  is connected to a facility by the algorithm) relative to  $\tau$ . Note that our algorithm grows  $\alpha_j$  until it reaches value  $\gamma \cdot (s_j - a_j)$ . Variable  $d_j$  will denote the distance between the locations of the client  $j$  and the facility  $f$ . Let  $\text{open}(f)$  denote the cost of opening the facility  $f$ .

Consider the following factor revealing LP:

► **Linear Program 1.**

$$z_k(\gamma) = \max \frac{(1 + \gamma) \cdot \sum_{i=0}^{k-1} (s_i - a_i)}{\text{open}(f) + \sum_{i=0}^{k-1} (d_i + |a_i|)} \quad (1a)$$

$$s_i \leq s_{i+1} \quad \forall 0 \leq i < k-1 \quad (1b)$$

$$(\gamma - 1) \cdot s_i - \gamma \cdot a_i \leq d_i + d_j + (\gamma - 1) \cdot s_j - \gamma \cdot a_j \quad \forall 0 \leq j < i < k \quad (1c)$$

$$\sum_{i=\ell}^{k-1} \max \{ \gamma \cdot (s_\ell - a_i) - d_i, 0 \} \leq \text{open}(f) \quad \forall \ell < k \quad (1d)$$

$$d_i \geq 0, s_i \geq a_i \quad \forall 0 \leq i < k \quad (1e)$$

► **Lemma 7.**  $z_k(\gamma)$  is an upper bound on the competitive ratio of the algorithm for a fixed value of parameter  $\gamma$  on a star of OPT with  $k$  clients.

**Proof.** It suffices to argue that the constraints (1b-1e) are satisfied for any values  $d_i, a_i, s_i$  representing the situation of clients from a single star of OPT in an actual run of the online algorithm.

We consider the set of clients in the order of them being connected by the online algorithm. If two clients are connected at the same time, we first take the one that arrived first. Constraints (1b) are trivially satisfied by this ordering.

Constraints (1d) reflect the situation just before time  $t_\ell^c$  when client  $\ell$  gets connected by Algorithm 1. The left-hand side is a sum of the contributions of still active clients towards opening a facility in the very same spot as OPT has a facility for this star. Obviously, these contributions cannot exceed the cost of opening a facility.

Finally, to argue for Constraints (1c) being satisfied we consider two cases. If  $s_i = s_j$ , then  $a_j \leq a_i$ , and the constraint is trivially satisfied. Otherwise,  $s_i > s_j$ . Consider the moment just before client  $i$  gets connected. Client  $i$  cannot have a budget that would be more than sufficient to connect to the facility where client  $j$  is connected. The connectivity budget of  $i$  is then  $\gamma \cdot (s_i - a_i)$  and the distance to the facility serving  $j$  plus waiting at this facility for  $i$  can be upper bounded by  $d_i + d_j + (s_i - s_j) + \gamma \cdot (s_j - a_j)$ , see Figure 1. ◀

### 3.1 Bounding the LP value

Here we will show an alternative (an in our opinion conceptually much simpler than the original one from Sec 5.2 of [33]) method to upper-bound a sequence of factor-revealing linear programs.

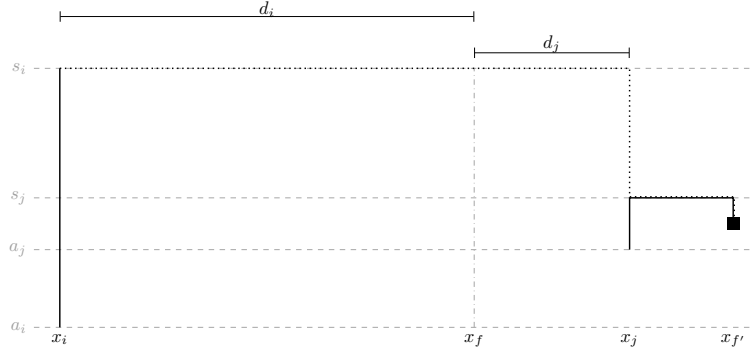
Our task now is to bound the value of  $z_k(\gamma)$  for every  $k$ . It is insufficient to compute  $z_k(\gamma)$  for some fixed  $k$  as it is monotonically increasing with  $k$  (a fact which we prove in a slightly weaker form). In this section, we will however show that it converges to a constant as  $k$  goes to infinity.

► **Proposition 8.** For any  $\gamma > 0$  and  $k, m \in \mathbb{N}_+$ , it holds that  $z_k(\gamma) \leq z_{k \cdot m}(\gamma)$ .

**Proof.** Let  $\langle d, a, s \rangle \in \mathbb{R}^k \times \mathbb{R}^k \times \mathbb{R}^k$  be a solution to LP1 optimizing  $z_k(\gamma)$ . We will construct a solution  $\langle d', a', s' \rangle \in \mathbb{R}^{k \cdot m} \times \mathbb{R}^{k \cdot m} \times \mathbb{R}^{k \cdot m}$  of the same value.

For  $i \in [k]$  and  $r \in [m]$ , let  $d'_{m \cdot i + r} = d_i/m$ ,  $a'_{m \cdot i + r} = a_i/m$ , and  $s'_{m \cdot i + r} = s_i/m$ . The inequalities (1b), (1c), and (1e) are satisfied by the new solution, since the same numbers appear in these inequalities in the solution  $\langle d, a, s \rangle$ . To show feasibility of  $\langle d', a', s' \rangle$  we need to argue that (1d) is satisfied for  $\ell$  not divisible by  $m$ :





■ **Figure 1** Illustration for Constraint (1c) (the *triangle inequality*). Before time  $s_i$ , the budget of client  $i$  must not be sufficient to connect  $i$  to the facility  $f'$  currently serving  $j$  – otherwise the algorithm would have already connected it. The cost of such a connection can be bounded by a sum of  $d_i + d_j$  (upper bounds  $\text{dist}(x_i, x_j)$ ),  $s_i - s_j$  (the waiting cost on the facility side since  $s_j$ ), and  $\gamma \cdot (s_j - a_j)$  (upper-bounds  $\text{dist}(x_j, y_{f'})$  and the remainder of waiting of  $f'$ ).

$$\begin{aligned}
& \sum_{i=\ell}^{m \cdot k - 1} \max \{ \gamma \cdot (s'_\ell - a'_i) - d'_i, 0 \} \\
& \leq \sum_{i=m \cdot \lfloor \ell/m \rfloor}^{m \cdot k - 1} \max \{ \gamma \cdot (s'_\ell - a'_i) - d'_i, 0 \} \quad (\text{more summands}) \\
& = \sum_{i=m \cdot \lfloor \ell/m \rfloor}^{m \cdot k - 1} \max \{ \gamma \cdot (s'_{m \cdot \lfloor \ell/m \rfloor} - a'_i) - d'_i, 0 \} \quad (\text{since } s'_\ell = s'_{m \cdot \lfloor \ell/m \rfloor}) \\
& = \sum_{i=\lfloor \ell/m \rfloor}^{k-1} \max \{ \gamma \cdot (s_{\lfloor \ell/m \rfloor} - a_i) - d_i, 0 \} \quad (\text{by definition of } \langle d', a', s' \rangle) \\
& \leq \text{open}(f).
\end{aligned}$$

To determine how  $z(\gamma)$  converges, we will define another LP, whose optimal value will always upper-bound LP1.

► **Linear Program 2.** This program with optimal value equal  $y_k(\gamma)$  is defined to be the same as LP1 except for the following inequality swapped for (1d):

$$\sum_{i=\ell+1}^{k-1} \max \{ \gamma \cdot (s_\ell - a_i) - d_i, 0 \} \leq \text{open}(f) \quad \forall \ell < k. \quad (2d)$$

► **Proposition 9.** For any  $\gamma > 0$  and  $k, m \in \mathbb{N}_+$  it holds that  $y_k(\gamma) \geq y_{k \cdot m}(\gamma)$ .

**Proof.** Let  $\langle d, a, s \rangle \in \mathbb{R}^{k \cdot m} \times \mathbb{R}^{k \cdot m} \times \mathbb{R}^{k \cdot m}$  be the solution to LP2 maximizing  $y_{k \cdot m}(\gamma)$ . We define a solution  $\langle d', a', s' \rangle \in \mathbb{R}^k \times \mathbb{R}^k \times \mathbb{R}^k$  of the same value as:

$$d'_i = \sum_{r=0}^{m-1} d_{m \cdot i + r}, \quad a'_i = \sum_{r=0}^{m-1} a_{m \cdot i + r}, \quad s'_i = \sum_{r=0}^{m-1} s_{m \cdot i + r}, \quad \forall 0 \leq i < k.$$

Again, we only need to focus on the inequality (2d):

$$\begin{aligned}
 & \sum_{i=\ell+1}^{k-1} \max\{\gamma \cdot (s'_\ell - a'_i) - d'_i, 0\} \\
 &= \sum_{i=\ell+1}^{k-1} \max\left\{ \sum_{r=0}^{m-1} \gamma \cdot (s_{m \cdot \ell + r} - a_{m \cdot i + r}) - d_{m \cdot i + r}, 0 \right\} \quad (\text{def. of } \langle d', a', s' \rangle) \\
 &\leq \sum_{i=\ell+1}^{k-1} \max\left\{ \sum_{r=0}^{m-1} \gamma \cdot (s_{m \cdot (\ell+1) - 1} - a_{m \cdot i + r}) - d_{m \cdot i + r}, 0 \right\} \quad (\text{monotonicity of } s_i) \\
 &\leq \sum_{i=\ell+1}^{k-1} \sum_{r=0}^{m-1} \max\{\gamma \cdot (s_{m \cdot (\ell+1) - 1} - a_{m \cdot i + r}) - d_{m \cdot i + r}, 0\} \quad (\text{Jensen's inequality}) \\
 &= \sum_{i=m \cdot (\ell+1)}^{m \cdot k - 1} \max\{\gamma \cdot (s_{m \cdot (\ell+1) - 1} - a_i) - d_i, 0\} \\
 &\leq \text{open}(f). \quad \blacktriangleleft
 \end{aligned}$$

Finally we show the relation between the corresponding  $z$  and  $y$  values.

► **Proposition 10.** *For any  $\gamma > 0$  and  $k \in \mathbb{N}_+$  it holds that  $y_k(\gamma) \geq z_k(\gamma)$ .*

**Proof.** Let  $\langle d, w, s \rangle \in \mathbb{R}^k \times \mathbb{R}^k \times \mathbb{R}^k$  be the solution to LP1 maximizing  $z_k(\gamma)$ . The same solution is feasible for LP2, as (2d) is weaker (has strictly fewer summands on the left side) than (1d). ◀

Proposition 10 completes the picture and lets us compare  $z(\gamma)$  with  $y(\gamma)$  for any  $k$ , even with the weak monotonicity we show in Propositions 8 and 9.

► **Corollary 11.** *For any  $\gamma > 0$  and for any  $k_1, k_2 \in \mathbb{N}_+$ , we have*

$$z_{k_1}(\gamma) \stackrel{P8}{\leq} z_{k_1 \cdot k_2}(\gamma) \stackrel{P10}{\leq} y_{k_1 \cdot k_2}(\gamma) \stackrel{P9}{\leq} y_{k_2}(\gamma).$$

We can now solve  $y_k(\gamma)$  for some  $k$  and obtain a bound on all  $\{z_\ell(\gamma)\}_{\ell \in \mathbb{N}_+}$ . The higher  $k$  we choose, the more precise bound we get in return.

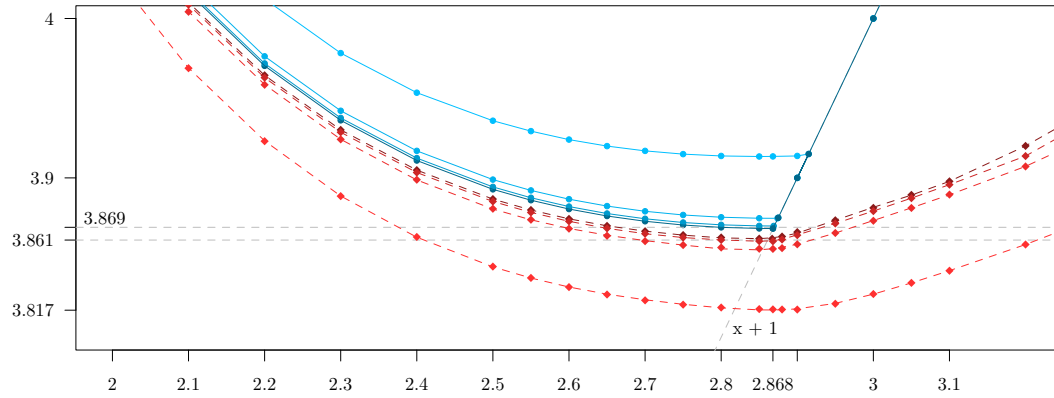
## 4 Computation of the competitive bounds

Having Corollary 11 ready to use, we now can lean on a major strength of our proof strategy: We can now simply use a linear programming solver to compute a valid bound the competitive ratio.

Our computational results are summarized in Figure 2. We have been able to solve LPs with up to 1500 clients, which allows us to make the following two claims, one formally proved and one empirical:

▷ **Claim 12.** There exists an optimal solution for the linear program LP2 with parameters  $k = 1500$  and  $\gamma = 2.868$  of objective value  $y_{1500}(2.868) \approx 3.869$ .

**Proof.** We provide the proof in the form of feasible LP solution and a feasible dual solution of the same objective value. The solutions are available along with the rest of our data at [13]. ◀



■ **Figure 2** Visual representation of the computed bounds on the competitive ratio as functions of  $\gamma$ . Each data point corresponds to a single optimal solution of LP1 or LP2, the curves are interpolated. The blue (solid) curves correspond to the upper bound LP2 for  $k = 100, 500, 1000$  and  $1500$  respectively. The red (dashed) curves represent the solutions of LP1 for the same steps of  $k$ . We obtain that our algorithm is  $3.869$ -competitive for  $\gamma = 2.868$  (Claim 12), and it is not better than  $3.861$ -competitive (Empirical Claim 13). Note that in line with results of Section 3, for a fixed value of  $\gamma$ , the value of  $z_k(\gamma)$  increases with increasing  $k$  while  $y_k(\gamma)$ , its upper bound, decreases with  $k$ .

▷ **Empirical Claim 13.** The objective function  $z_{1500}(\gamma)$  of LP1, viewed as a function of  $\gamma$ , is minimized for  $\gamma \approx 2.867$  with objective value  $z_{1500}(2.867) \approx 3.861$ .

With Claim 12, we can complete the proof of Theorem 1:

**Proof of Theorem 1.** We partition the cost of OPT into costs associated with a single facility that OPT opens, forming a star with the facility in the center. Lemma 7 gives us that  $z_k(\gamma)$  is an upper bound on the competitive ratio of Algorithm 1 for a single star, and Corollary 11 implies that computing a single optimal solution of the upper bound LP2 is sufficient for the bound on competitive ratio. Finally, Claim 12 tells us that for  $\gamma = 2.868$ , the competitive ratio on a single star – and thus, on all stars – is, after rounding, at most  $3.869$ . ◀

The complementary Empirical Claim 13 provides a lower bound on the efficiency of our method, as it suggests that Algorithm 1 is not better than  $3.861$ -competitive, and thus our analysis is almost tight.

In Figure 2, we can see that all curves for LP2 begin to follow the line  $y = x + 1$  once they intersect it, which is not the case for the dashed curves of LP1. The following observation explains the structure of feasible solutions once the  $y = x + 1$  line is crossed.

► **Observation 14.** Consider the linear program LP2 with parameters  $k \geq 2$  and  $\gamma$ , and let us set  $\text{open}(f) + \sum_{i=0}^{k-1} (d_i + |a_i|) = 1$ . Then, there is a feasible solution to LP2 with objective value  $\gamma + 1$ .

**Proof.** Recall that LP2 is designed for the case the optimal solution opens a facility at time  $\tau$ , which we can think of as  $0$ . For our feasible solution, we set all distances to be identically zero and we release the first client at time  $a_0 = -1$ , with  $a_1 = a_2 = \dots = a_{k-1} = 0$ . The cost of opening a facility is also zero. For the service times of the algorithm, we set  $s_0 = s_1 = \dots = s_{k-1} = 0$ .

We first double check that indeed, setting these values gives us  $\text{open}(f) + \sum_{i=0}^{k-1} (d_i + |a_i|) = 1$ ; we now proceed to check that the solution is feasible. As all distances and service times are zero, all constraints except the type (2d) are trivially satisfied.

Let us restate the last set of constraints, (2d):

$$\sum_{i=\ell+1}^{k-1} \max \{ \gamma \cdot (s_\ell - a_i) - d_i, 0 \} \leq \text{open}(f) \quad \forall \ell < k.$$

Observe that  $a_0$  does not appear in any constraint of this type, and so it is never checked that  $\gamma(s_0 - a_0) \leq \text{open}(f) = 0$ , which would be false for any  $s_0 > -1$ . All constraints of this type only check variables  $a_1, \dots, a_{k-1}$  and  $s_1, \dots, s_{k-1}$ , which are all set to zero, causing the constraints to be indeed satisfied.  $\blacktriangleleft$

The quirk from Observation 14 does not cause any issues for us, as the minimum of  $z_k(\gamma)$ , which serves as a lower bound of the competitive ratio of Algorithm 1, is attained before the curve  $z_k(\gamma)$  intersects  $y = x + 1$  (see Figure 2).

For our computations, we use the LP solver Gurobi Optimizer version 9 [32]. The source code of our generator and selected few solutions (that can be verified) can be found online at [13].

## 5 From Two-Sided to One-Sided Delay

We will now show how an online algorithm that solves the two-sided variant of the facility location problem can be transformed into an algorithm for the one-sided variant. We require that the former algorithm is  $(\lambda, \xi)$ -sensible for some constants  $\lambda > 1$  and  $\xi > 1$  (cf. Definition 5). Recall that by Lemma 6, Algorithm 1 is  $(3/2, 2)$ -sensible for  $\gamma = 2$ .

### 5.1 Algorithm definition

On the basis of an arbitrary online  $(\lambda, \xi)$ -sensible algorithm ATS for the two-sided variant, we construct an online algorithm for the one-sided variant in the following way.

► **Algorithm 2.** *Our algorithm simulates the execution of ATS on the input instance, and when ATS opens a facility  $f$  at point  $y$  at time  $\tau$ , and connects a subset of clients to this facility, our algorithm does the same at time  $\tau$ .*

*From this point on, our algorithm tracks, in an online manner, all (late) connections to facility  $f$ . Clients that are already connected by ATS but not yet connected by our algorithm are called pending for  $f$ . At some times (defined below), our algorithm opens a new facility at  $y$  (called a copy of  $f$ ), and connects all clients pending for  $f$  to this copy.*

- *Let  $\tau + b$  be the earliest time when  $p \cdot b \geq \text{open}(y)$ , where  $p$  denotes the number of pending clients. (The inequality may be strict if it becomes true because of the increment of  $p$ .) If there are no pending clients at time  $\tau + \text{open}(y)$ , we set  $b = \text{open}(y)$ . In either case, the first copy of  $f$  is opened at time  $\tau + b$ .*

■ *Let*

$$\tilde{n} = \frac{\lambda \cdot \xi}{\lambda - 1} \cdot \frac{\text{open}(y)}{b},$$

*Note that  $\tilde{n} > 1$  by the choice of  $b$ . Let  $q = \sqrt{\log \tilde{n}}$  and let  $\ell$  be the smallest integer such that  $q^\ell > \tilde{n}$ . Subsequent facility copies are opened at times  $\tau + b \cdot q^i$  for all integers  $i \in \{1, \dots, \ell\}$ .*

In total, Algorithm 2 opens a facility at  $y$  at time  $\tau$ , and  $\ell + 1$  of its copies at times  $\tau + b \cdot q^i$ , for  $i \in \{0, \dots, \ell\}$ .

## 5.2 Analysis

In the following, we assume that ATS is  $(\lambda, \xi)$ -sensible for some fixed constants  $\lambda > 1$  and  $\xi > 1$ . We show that Algorithm 2 is a valid algorithm (i.e., it eventually connects all clients), and we upper-bound its total cost in comparison to the cost paid by ATS.

We perform the cost comparison for each facility  $f$  opened by ATS; in the following, we use  $y$  to denote its location and  $\tau$  to denote its opening time in the solution of ATS. We also use values of  $b$ ,  $\tilde{n}$ ,  $q$  and  $\ell$  as computed by Algorithm 2 when handling clients connected to  $f$  by ATS.

### Correctness

We start with the following helper claim.

► **Lemma 15.** *It holds that  $\tilde{n} \leq n \cdot \lambda \cdot \xi / (\lambda - 1)$ .*

**Proof.** Let  $p'$  be the number of clients ATS connected to  $f$  within the interval  $(\tau, \tau + b]$ . If  $p' = 0$ , then  $b = \text{open}(y)$  and thus  $\tilde{n} = (\lambda \cdot \xi) / (\lambda - 1)$ . The lemma follows trivially as  $n \geq 1$ . Otherwise,  $p' > 0$ , and then  $p' \cdot b \geq \text{open}(y)$ . In this case,  $\tilde{n} = (\lambda \cdot \xi) \cdot (\lambda - 1)^{-1} \cdot \text{open}(y) / b \leq (\lambda \cdot \xi) \cdot (\lambda - 1)^{-1} \cdot p'$ . As  $p' \leq n$ , the lemma follows. ◀

► **Lemma 16.** *Algorithm 2 connects all clients.*

**Proof.** The last copy of the facility  $f$  is opened by Algorithm 2 at time  $\tau + b \cdot q^\ell$ . Thus, it suffices to show that ATS connects no clients to  $f$  after this time.

Fix any  $t > 0$ , any integer  $i \geq 0$ , and consider time interval

$$I_i^t = \left( \tau + \lambda^i \cdot t, \tau + \lambda^{i+1} \cdot t \right].$$

As ATS is  $(\lambda, \xi)$ -sensible, it connects at most  $\xi \cdot \text{open}(y) / (\lambda^i \cdot t)$  clients within interval  $I_i^t$ .

Note that  $\biguplus_{i=0}^{\infty} I_i^t = (\tau + \lambda^0 \cdot t, \infty) = (\tau + t, \infty)$ . Thus, for an arbitrary  $t$ , by summing over all  $i \geq 0$ , the number of clients connected after time  $\tau + t$  is at most

$$\sum_{i=0}^{\infty} \frac{\xi \cdot \text{open}(y)}{\lambda^i \cdot t} = \frac{\lambda \cdot \xi}{\lambda - 1} \cdot \frac{\text{open}(y)}{t} = \frac{\tilde{n} \cdot b}{t}.$$

Hence, the number of clients connected after time  $\tau + b \cdot q^\ell$  is at most  $\tilde{n} \cdot b / (b \cdot q^\ell) = \tilde{n} / q^\ell < 1$ . As the number of clients is integral, it must be zero. ◀

### Bounding the waiting time

Now we focus on bounding the total waiting time of Algorithm 2. Let  $C$  be the set of clients connected by ATS to the facility  $f$  at  $y$ . We split  $C$  into four disjoint parts: the clients connected by ATS at time  $\tau$ , within interval  $(\tau, \tau + b)$ , at time  $\tau + b$ , and after  $\tau + b$ . We denote these parts  $C_{=\tau}$ ,  $C_{(\tau, \tau+b)}$ ,  $C_{=\tau+b}$  and  $C_{>\tau+b}$ , respectively. For any client  $j \in C$ , let  $w_j^{\text{ATS}}$  and  $w_j$  denote its waiting cost in the solutions of ATS and Algorithm 2, respectively.

► **Lemma 17.** *For any client  $j \in C \setminus C_{(\tau, \tau+b)}$ , it holds that  $w_j \leq q \cdot w_j^{\text{ATS}}$ .*

**Proof.** For any client  $j$ , let  $t_j$  be the time of its arrival and  $t_j^c$  the time ATS connects it to facility  $f$ .

In the case  $j \in C_{=\tau}$ , in both solutions of ATS and Algorithm 2, client  $j$  waits till  $\tau$ , and thus  $w_j = w_j^{\text{ATS}}$ . The case  $j \in C_{=\tau+b}$  is possible only if  $t_j = \tau + b$ , and then  $w_j = 0 \leq q \cdot w_j^{\text{ATS}}$ .

It remains to consider the case  $j \in C_{>\tau+b}$ , i.e.,  $t_j^c > \tau + b$ . Let  $w_j^{\text{init}} = t_j^c - t_j$ ; this amount represents the inevitable waiting time that  $j$  incurs in both solutions. Note that  $w_j^{\text{ATS}} - w_j^{\text{init}} = t_j^c - \tau$  corresponds to the facility-side waiting cost of  $j$  in the solution of ATS. By Lemma 16,  $t_j^c \leq \tau + b \cdot q^\ell$ . Thus, there exists an integer  $i \in \{1, \dots, \ell\}$ , such that  $\tau + b \cdot q^{i-1} < t_j^c \leq \tau + b \cdot q^i$ . Algorithm 2 connects  $j$  at time  $\tau + b \cdot q^i$ , and therefore

$$w_j - w_j^{\text{init}} = \tau + b \cdot q^i - t_j^c \leq b \cdot q^i = q \cdot (b \cdot q^{i-1}) < q \cdot (t_j^c - \tau) = q \cdot (w_j^{\text{ATS}} - w_j^{\text{init}}).$$

The proof is concluded by adding  $w_j^{\text{init}}$  to both sides.  $\blacktriangleleft$

► **Lemma 18.** *It holds that  $\sum_{c \in C} w(c) \leq \text{open}(y) + q \cdot \sum_{c \in C} w^{\text{ATS}}(c)$ .*

**Proof.** We use the same notions of  $t_j$ ,  $t_j^c$ ,  $w_j$ ,  $w_j^{\text{ATS}}$  and  $w_j^{\text{init}}$  as in the previous proof.

Fix any client  $j \in C_{(\tau, \tau+b)}$ . Clearly,  $t_j^c > \tau$ . Furthermore,  $j$  is served by Algorithm 2 at time  $\tau + b$ , and thus

$$\sum_{j \in C_{(\tau, \tau+b)}} (w_j - w_j^{\text{init}}) = \sum_{j \in C_{(\tau, \tau+b)}} (\tau + b - t_j^c) \leq |C_{(\tau, \tau+b)}| \cdot b < \text{open}(y).$$

The last inequality follows by the definition of  $b$  in Algorithm 2. By adding  $\sum_{c \in C_{(\tau, \tau+b)}} w_j^{\text{init}}$  to both sides, we obtain

$$\sum_{j \in C_{(\tau, \tau+b)}} w_j \leq \text{open}(y) + \sum_{j \in C_{(\tau, \tau+b)}} w_j^{\text{init}} \leq \text{open}(y) + \sum_{j \in C_{(\tau, \tau+b)}} w_j^{\text{ATS}}.$$

By combining the inequality above with Lemma 17 applied to all  $C \setminus C_{(\tau, \tau+b)}$ , we obtain the lemma statement.  $\blacktriangleleft$

## Competitive ratio

Finally, we use our bounds to prove Theorem 2, i.e., show that the competitive ratio of Algorithm 2.

**Proof of Theorem 2.** We fix any  $(\lambda, \xi)$ -sensible  $O(1)$ -competitive algorithm ATS for the two-sided variant. By Lemma 6, such algorithm exists for  $\lambda = 3/2$  and  $\xi = 2$ .

We fix any facility opened by ATS at location  $y$ ; let  $C$  denote the set of clients that are connected to this facility in the solution of ATS. Below we show that the total cost pertaining to clients from  $C$  in the solution of Algorithm 2 is at most  $O(\log n / \log \log n)$  times larger than the cost pertaining to these clients in the solution of ATS.

The theorem will then follow by summing this relation over all facilities opened by ATS, and observing that the value of OPT for the one-sided variant can be only more expensive than OPT for the two-sided variant (as the two-sided variant is a relaxation of the one-sided variant).

We relate parts of cost of solution produced by Algorithm 2 to the corresponding costs of ATS.

- The cost of connecting clients from  $C$  by Algorithm 2 is trivially equal to the connection cost in the solution of ATS.

- To bound the waiting cost of clients from  $C$ , we apply Lemma 18 obtaining that  $\sum_{c \in C} w(c) \leq \text{open}(y) + q \cdot \sum_{c \in C} w^{\text{ATS}}(c)$ . As  $q = \sqrt{\log \tilde{n}} = O(\sqrt{\log n}) = O(\log n / \log \log n)$ , the total waiting cost of Algorithm 2 is at most  $\text{open}(y) + O(\log n / \log \log n) \cdot \sum_{c \in C} w^{\text{ATS}}(c)$ .
- Algorithm 2 opens a facility at location  $y$  at time  $\tau$  and then  $\ell + 1$  of its copies at times  $\tau + b \cdot q^i$  for  $i \in \{0, \dots, \ell\}$ . Thus, its overall opening cost is  $(\ell + 2) \cdot \text{open}(y)$ . Recall that  $\ell = \lceil \log \tilde{n} / \log q \rceil = O(\log \tilde{n} / \log \log \tilde{n})$ . By Lemma 15, the latter amount is  $O(\log n / \log \log n)$ . Thus, the opening cost of Algorithm 2 is  $O(\log n / \log \log n) \cdot \text{open}(y)$  while that of ATS is  $\text{open}(y)$ .

The proof follows by adding guarantees of all the cases above. ◀

## References

- 1 Karen Aardal, Jarosław Byrka, and Mohammad Mahdian. Facility location. In *Encyclopedia of Algorithms*, pages 717–724. Springer, 2016. doi:10.1007/978-1-4939-2864-4\_139.
- 2 Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Information and Computation*, 194(2):175–202, 2004. doi:10.1016/j.ic.2004.06.002.
- 3 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul M. Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. In *Proc. 20th Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX/RANDOM)*, pages 1:1–1:20, 2017.
- 4 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1051–1061, 2017.
- 5 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay — clairvoyance is not required. In *Proc. 28th European Symp. on Algorithms (ESA)*, pages 8:1–8:21, 2020. doi:10.4230/LIPIcs.ESA.2020.8.
- 6 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. *Theory of Computing Systems*, 64(4):572–592, 2020. doi:10.1007/s00224-019-09963-7.
- 7 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalaya Panigrahi. Online service with delay. In *Proc. 49th ACM Symp. on Theory of Computing (STOC)*, pages 551–563, 2017.
- 8 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In *Proc. 2021 ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 301–320, 2021. doi:10.1137/1.9781611976465.20.
- 9 Yossi Azar and Noam Touitou. General framework for metric optimization problems with delay or with deadlines. In *Proc. 60th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 60–71, 2019. doi:10.1109/FOCS.2019.00013.
- 10 Yossi Azar and Noam Touitou. Beyond tree embeddings - a deterministic framework for network design with deadlines or delay. In *Proc. 61st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 1368–1379, 2020. doi:10.1109/FOCS46700.2020.00129.
- 11 Marcin Bienkowski, Martin Böhm, Jarosław Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Nguyen Kim Thang, and Pavel Veselý. Online algorithms for multilevel aggregation. *Operations Research*, 68(1):214–232, 2020. doi:10.1287/opre.2019.1847.
- 12 Marcin Bienkowski, Martin Böhm, Jarosław Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Nguyen Kim Thang, and Pavel Veselý. New results on multi-level aggregation. *Theoretical Computer Science*, 861:133–143, 2021. doi:10.1016/j.tcs.2021.02.016.
- 13 Marcin Bienkowski, Martin Böhm, Jarosław Byrka, and Jan Marcinkowski. Data and computations for online facility location with linear delay. <https://github.com/bohmf1-double-sided-waiting>, 2021.



- 14 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Łukasz Jeż, Dorian Nogneng, and Jiri Sgall. Better approximation bounds for the joint replenishment problem. In *Proc. 25th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 42–54, 2014.
- 15 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Łukasz Jeż, Jiri Sgall, and Grzegorz Stachowiak. Online control message aggregation in chain networks. In *Proc. 13th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 133–145, 2013.
- 16 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In *Proc. 16th Workshop on Approximation and Online Algorithms (WAOA)*, pages 51–68, 2018. doi:10.1007/978-3-030-04693-4\_4.
- 17 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. A match in time saves nine: Deterministic online matching with delays. In *Proc. 15th Workshop on Approximation and Online Algorithms (WAOA)*, pages 132–146, 2017.
- 18 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. Online service with delay on a line. In *Proc. 25th Int. Colloq. on Structural Information and Communication Complexity (SIROCCO)*, pages 237–248, 2018. doi:10.1007/978-3-030-01325-7\_22.
- 19 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon.  $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1235–1244, 2017.
- 20 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Operations Research*, 61(4):1014–1029, 2013. doi:10.1287/opre.2013.1188.
- 21 Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010. doi:10.1137/070708901.
- 22 Rodrigo A. Carrasco, Kirk Pruhs, Cliff Stein, and José Verschae. The online set aggregation problem. In *Proc. 13th Latin American Theoretical Informatics Symposium (LATIN)*, pages 245–259, 2018. doi:10.1007/978-3-319-77404-6\_19.
- 23 Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005. doi:10.1137/S0097539701398594.
- 24 Marek Chrobak. Online aggregation problems. *SIGACT News*, 45(1):91–102, 2014.
- 25 Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003. doi:10.1137/S0097539703405754.
- 26 Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. On-line analysis of the TCP acknowledgment delay problem. *Journal of the ACM*, 48(2):243–273, 2001.
- 27 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 333–344, 2016.
- 28 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. *Theoretical Computer Science*, 754:122–129, 2019. doi:10.1016/j.tcs.2018.07.004.
- 29 Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007. doi:10.1016/j.jda.2006.03.001.
- 30 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008. doi:10.1007/s00453-007-9049-y.
- 31 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999. doi:10.1006/jagm.1998.0993.
- 32 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL: <https://www.gurobi.com>.
- 33 Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003. doi:10.1145/950620.950621.

- 34 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proc. 34th ACM Symp. on Theory of Computing (STOC)*, pages 731–740, 2002. doi:10.1145/509907.510012.
- 35 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgement and other stories about  $e/(e - 1)$ . *Algorithmica*, 36(3):209–224, 2003.
- 36 Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000. doi:10.1006/jagm.2000.1100.
- 37 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. doi:10.1016/j.ic.2012.01.007.
- 38 Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Impatient online matching. In *Proc. 29th Int. Symp. on Algorithms and Computation (ISAAC)*, pages 62:1–62:12, 2018. doi:10.4230/LIPIcs.ISAAC.2018.62.
- 39 Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006. doi:10.1137/S0097539703435716.
- 40 Adam Meyerson. Online facility location. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 426–431, 2001. doi:10.1109/SFCS.2001.959917.
- 41 David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proc. 29th ACM Symp. on Theory of Computing (STOC)*, pages 265–274, 1997. doi:10.1145/258533.258600.
- 42 Michael A. Stanko and David H. Henard. How crowdfunding influences innovation. *MIT Sloan Management Review*, 57(3):15, 2016.



# Prophet Matching in the Probe-Commit Model

Allan Borodin  

Department of Computer Science, University of Toronto, Canada

Calum MacRury  

Department of Computer Science, University of Toronto, Canada

Akash Rakheja  

Department of Computer Science, University of Toronto, Canada

---

## Abstract

We consider the online bipartite stochastic matching problem with known i.d. (independently distributed) online vertex arrivals. In this problem, when an online vertex arrives, its weighted edges must be probed (queried) to determine if they exist, based on known edge probabilities. Our algorithms operate in the probe-commit model, in that if a probed edge exists, it must be used in the matching. Additionally, each online node has a downward-closed probing constraint on its adjacent edges which indicates which sequences of edge probes are allowable. Our setting generalizes the commonly studied patience (or time-out) constraint which limits the number of probes that can be made to an online node's adjacent edges. Most notably, this includes non-uniform edge probing costs (specified by knapsack/budget constraint). We extend a recently introduced configuration LP to the known i.d. setting, and also provide the first proof that it is a relaxation of an optimal offline probing algorithm (the offline adaptive benchmark). Using this LP, we establish the following competitive ratio results against the offline adaptive benchmark:

1. A tight  $\frac{1}{2}$  ratio when the arrival ordering  $\pi$  is chosen adversarially.
2. A  $1 - 1/e$  ratio when the arrival ordering  $\pi$  is chosen u.a.r. (uniformly at random).

If  $\pi$  is generated adversarially, we generalize the prophet inequality matching problem. If  $\pi$  is u.a.r., we generalize the prophet secretary matching problem. Both results improve upon the previous best competitive ratio of 0.46 in the more restricted known i.i.d. (independent and identically distributed) arrival model against the standard offline adaptive benchmark due to Brubach et al. We are the first to study the prophet secretary matching problem in the context of probing, and our  $1 - 1/e$  ratio matches the best known result without probing due to Ehsani et al. This result also applies to the unconstrained bipartite matching probe-commit problem, where we match the best known result due to Gamlath et al.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** Stochastic probing, Online algorithms, Bipartite matching, Optimization under uncertainty

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.46

**Category** APPROX

**Related Version** Full Version: <https://arxiv.org/abs/2102.04325> [9]

**Acknowledgements** We would like to thank Brendan Lucier and David Wajc for their constructive comments on an early version of this paper.

## 1 Introduction

Stochastic probing problems are part of the larger area of decision making under uncertainty and more specifically, stochastic optimization. Unlike more standard forms of stochastic optimization, it is not just that there is some stochastic uncertainty in the set of inputs, stochastic probing problems involve inputs that cannot be determined without probing (at some cost and/or within some constraint). Applications of stochastic probing occur naturally



© Allan Borodin, Calum MacRury, and Akash Rakheja;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 46; pp. 46:1–46:24



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in many settings, such as in matching problems where compatibility cannot be determined without some trial or investigation (for example, in online dating, online advertising, and kidney exchange applications). There is by now an extensive literature for stochastic probing problems.

Although we are only considering “one-sided online bipartite matching”, stochastic matching was first considered in the context of a general graph by Chen et al. [18]. In this problem, the algorithm is presented an adversarially generated *stochastic graph*  $G = (V, E)$  as input, which has a probability  $p_e$  associated with each edge  $e$  and a *patience* (or time-out) parameter  $\ell_v$  associated with each vertex  $v$ . An algorithm probes edges in  $E$  in some adaptive order within the constraint that at most  $\ell_v$  edges are probed incident to any particular vertex  $v$ . The patience parameter can be viewed as a simple budgetary constraint, where each probe has unit cost and the patience parameter is the budget. When an edge  $e$  is probed, it is guaranteed to exist with probability exactly  $p_e$ . If an edge  $(u, v)$  is found to exist, then the algorithm must *commit* to the edge – that is, it must be added to the current matching. The goal is to maximize the expected size of a matching constructed in this way.

In addition to generalizing the results of Chen et al. to edge weights, Bansal et al. [6] introduced the *online bipartite stochastic matching problem*. In this problem, a single seller wishes to match their offline (indivisible) *items* to (unit-demand) *buyers* which arrive online one by one. The seller knows the possible *type/profile* of each online buyer, which is specified by edge probabilities, edge weights and a patience parameter. Here an edge probability models the likelihood a buyer type will purchase an item if the seller presents it to them, and an edge weight represents the revenue the seller will gain from making such a sale successfully. The patience of a buyer type indicates the maximum number of items they are willing to be shown. The online buyers are drawn i.i.d. from a known distribution, where the type of each online buyer is presented to the seller upon its arrival. The (potential) sale of an item to an online buyer must be made before the next online buyer arrives, and the seller’s goal is to maximize their expected revenue. As in the Chen et al. model, the seller must *commit* to the first sale to which an online buyer agrees. Fata et al. observed that this problem is closely related to the multi-customer *assortment optimization problem*, which has numerous practical applications in revenue management (see [24] for details).

We study the online bipartite stochastic matching problem in the more general known i.i.d. setting. Specifically, each online buyer is drawn from a (potentially) distinct distribution, and the draws are done independently. When online buyers arrive adversarially, we generalize the *prophet inequality matching problem* of Alaei et al. [4]. When online buyers arrive in random order, we generalize the *prophet secretary matching problem* of Ehsani et al. [22]. We note that prophet inequalities give rise to (and in some sense are equivalent to) order oblivious posted price mechanisms, as first studied in Hajiaghayi et al. [31] and further developed for multi-parameter settings in Chawla et al. [17] and recently in Correa et al. [19]. There have been a number of very recent works studying prophet matching problems with limited distributional sample access [16, 33]. In these works, a main emphasis has been towards understanding whether a few samples is sufficient to obtain the best known competitive ratios when one is instead given full access to the distributions. Our work is motivated by analogous questions for competitive ratios in the probe-commit model, and we provide a positive answer for adversarial arrivals as well as for random order arrivals.

The online bipartite stochastic matching problem models the altruistic kidney exchange problem where the offline nodes correspond to donors and the online nodes correspond to recipients (or vice versa). A trial (probe) must be performed to determine whether a donor/recipient pair may exchange kidneys, and the edge probability corresponds to the

likelihood of a permissible exchange. The online arrival setting models the restriction that the algorithm must process the recipients (or donors) in an order of which it cannot control. Our generalization of patience to downward-closed probing constraints is motivated by this application. Specifically, our framework includes knapsack/budget constraints, which allows us to model non-uniform trial costs. Another application is to online advertising, where an advertiser presents ads to consumers, and the edge probabilities represent the likelihood they will “click” on the presented ad. Basically, any matching problem in which there is uncertainty in whether the matches will succeed is a relevant application.

## 2 Preliminaries and Our Results

An input to the **online stochastic matching problem with known i.d. arrivals** firstly includes a **type graph**  $H_{\text{typ}} = (U, B, F)$ , which is a bipartite graph with edge weights  $(w_f)_{f \in F}$  and edge probabilities  $(p_f)_{f \in F}$  where  $F := U \times B$ . We refer to  $U$  as the **offline nodes** of  $H_{\text{typ}}$  and  $B$  as its **type nodes**. An **online probing algorithm** is given access to  $H_{\text{typ}}$ , and for each  $u \in U$ , and  $b \in B$ ,  $p_{u,b}$  indicates the probability that an **active** edge between  $u$  and  $b$  exists, and  $w_{u,b} \geq 0$  indicates the reward for matching  $u$  to  $b$ . Given an arbitrary set  $S$ , let  $S^{(*)}$  denote the set of all tuples (strings) formed from  $S$ , whose entries (characters) are all distinct. Note that we use tuple/string notation and terminology interchangeably. Each  $b \in B$  has its own **(online) probing constraint**  $\mathcal{C}_b \subseteq \partial(b)^{(*)}$ , where  $\partial(b) := U \times \{b\}$ . This probing constraint indicates whether the edges of  $e = (e_1, \dots, e_k) \in \partial(b)^{(*)}$  may be probed by the algorithm in the order of its indices. Here a **probe** of an edge informs the algorithm whether or not the edge is active. We make the minimal assumption that  $\mathcal{C}_b$  is **downward-closed**; that is, if  $e \in \mathcal{C}_b$ , then any substring or permutation of  $e$  is also in  $\mathcal{C}_b$ . This includes matroid constraints, as well when  $b \in B$  has a **budget**  $L_b \geq 0$ , and **(edge) probing costs**  $(c_{u,b})_{u \in U}$ , such that  $e = (e_1, \dots, e_k) \in \mathcal{C}_b$  provided  $\sum_{i=1}^k c_{e_i} \leq L_b$ . Observe that if  $b$  has uniform probing costs, then this corresponds to the previously discussed case of an integer **patience parameter**  $\ell_b \geq 1$ .

The input additionally consists of a sequence of distributions  $(\mathcal{D}_i)_{i=1}^n$  supported on  $B$ , where  $n \geq 1$  indicates the number of **online** vertices to be presented to the algorithm. Specifically, for  $i = 1, \dots, n$ , vertex  $v_i$  is drawn independently from  $\mathcal{D}_i$ , and we define  $V$  to be the multiset including  $v_1, \dots, v_n$ . The online probing algorithm executes on the **stochastic graph**  $G = (U, V, E)$  where  $E := U \times V$ , and we denote  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$  to indicate  $G$  is drawn from  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ . We assume that each  $e \in E$  is active independently with probability  $p_e$ , where the **edge state**  $\text{st}(e) \sim \text{Ber}(p_e)$  indicates this event.

Initially, the online algorithm is only given access to  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ , yet its goal is to build a matching of active edges of  $G$  of largest possible expected weight. In the **adversarial order arrival model (AOM)**, a permutation  $\pi$  is generated by an **oblivious adversary**, in which case  $\pi$  is a function of  $H_{\text{typ}}$  and  $(\mathcal{D}_i)_{i=1}^n$ . In the **random order arrival model (ROM)**,  $\pi$  is generated u.a.r., independent of all other randomization. In either setting,  $\pi$  is unknown to the algorithm. For each  $t = 1, \dots, n$ , vertex  $v_{\pi(t)}$  is presented to the algorithm, along with its edge weights, probabilities, and online probing constraint. Note that the algorithm is also presented the value  $\pi(t)$ , and thus learns from which distribution  $v_{\pi(t)}$  was drawn. However, the edge states  $(\text{st}(e))_{e \in \partial(v_{\pi(t)})}$  initially remain hidden to the algorithm. Instead, using all past available information regarding  $v_{\pi(1)}, \dots, v_{\pi(t-1)}$ , the algorithm must **probe** the edges of  $\partial(v_{\pi(t)})$  to reveal their states, while adhering to  $\mathcal{C}_{v_{\pi(t)}}$ . The algorithm operates in the **probe-commit model**, in which there is a **commitment** requirement upon probing an edge. Specifically, if an edge  $e = (u, v)$  is probed and turns out to be

active, then the online probing algorithm must make an irrevocable decision as to whether or not to include  $e$  in its matching, prior to probing any subsequent edges. This definition of commitment is the one considered by Gupta et al. [30], and is slightly different but equivalent to the Chen et al. [18] model in which an active edge must be immediately accepted into the matching. The algorithm always has the option to pass on  $v_{\pi(t)}$ , yet its (potential) match must be made before the next online vertex arrives.

In general, it is easy to see that even when the edges are unweighted and the algorithms initially knows the stochastic graph we cannot hope to obtain a non-trivial competitive ratio against the expected size of an optimal matching of the stochastic graph. Consider a stochastic graph with a single online vertex with patience 1, and  $k \geq 1$  offline (unweighted) vertices where each edge  $e$  has probability  $\frac{1}{k}$  of being active. The expectation of an online probing algorithm will be at most  $\frac{1}{k}$  while the expected size of an optimal matching will be  $1 - (1 - \frac{1}{k})^k \rightarrow 1 - \frac{1}{e}$  as  $k \rightarrow \infty$ . The standard approach in the literature is to instead consider the **offline stochastic matching problem** and benchmark against an *optimal offline probing algorithm* [6, 2, 14, 15]. An **offline probing algorithm** knows  $G = (U, V, E)$ , but initially the edge states  $(\text{st}(e))_{e \in E}$  are hidden. Its goal is to construct a matching of active edges of  $G$  with weight as large as possible in expectation. It can adaptively probe the edges of  $E$  in any order, but must satisfy the probing constraints  $(\mathcal{C}_v)_{v \in V}$  at each step of its execution. That is, edges  $e \in E^{(*)}$  may be probed in order, provided  $e^v \in \mathcal{C}_v$  for each  $v \in V$ , where  $e^v$  is the substring of  $e$  restricted to edges of  $\partial(v)$ . It must also operate in the same probe-commit model as an online probing algorithm. We define the **(offline) adaptive benchmark** as an optimal offline probing algorithm, and denote  $\text{OPT}(G)$  as the expected weight of its matching when executing on  $G$ . An alternative weaker benchmark used by Brubach et al. [11, 12] is the **online adaptive benchmark**. This is defined as an optimal offline probing algorithm which executes on  $G$  and whose edge probes respect some adaptively chosen vertex ordering on  $V$ . Equivalently, the edge probes involving each  $v \in V$  occur contiguously: if  $e' = (u, v') \in E$  is probed after  $e = (u, v)$  for  $v' \neq v$ , then no edge of  $\partial(v)$  is probed following  $e'$ . We benchmark against  $\mathbb{E}[\text{OPT}(G)]$ , where the expectation is over the randomness in  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ . For clarity, we denote  $\mathbb{E}[\text{OPT}(G)]$  by  $\text{OPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .

Observe that if  $p_e \in \{0, 1\}$  for each  $e \in F$  of  $H_{\text{typ}} = (U, B, F)$ , then probing is unnecessary, and the offline adaptive benchmark and the online adaptive benchmark both correspond to the expected weight of the maximum matching of  $G$ . In this special case, the online algorithm also does not need to probe edges, and so no matter which benchmark is chosen, the problem generalizes either the **prophet inequality matching problem** or the **prophet secretary matching problem**, depending on whether  $\pi$  is adversarial or u.a.r., respectively.

► **Theorem 1.** *If  $\mathcal{M}(\pi)$  is the matching returned by Algorithm 8 when presented the online vertices of  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$  in an adversarial order  $\pi : [n] \rightarrow [n]$ , then  $\mathbb{E}[w(\mathcal{M}(\pi))] \geq \frac{1}{2} \text{OPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .*

► **Remark 2.** We say that Algorithm 8 attains a  $1/2$  competitive ratio or is  $1/2$ -competitive (against adversarial arrivals). This is a tight bound since the problem generalizes the classic single item prophet inequality for which  $\frac{1}{2}$  is an optimal competitive ratio. Recently, Brubach et al. [11, 12] independently proved the same competitive ratio against the online adaptive benchmark when  $G$  has patience values and the arrival order is adversarial yet known to the algorithm. Our results are incomparable, as their results can be applied to an *unknown* patience framework (at a loss in competitive ratio), whereas our results apply to *known* downward-closed online probing constraints, and hold against a stronger benchmark.



► **Theorem 3.** *If  $\mathcal{M}$  is the matching returned by Algorithm 9 when presented the online vertices of  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$  in random order, then  $\mathbb{E}[w(\mathcal{M})] \geq (1 - \frac{1}{e}) \text{OPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .*

► **Remark 4.** In part due to its applications to multi-customer assortment optimization, the special case of identical distributions with one-sided patience values has been studied in multiple works [6, 2, 14, 15], beginning with the 0.12 competitive ratio of Bansal et al. [6]. The previously best known competitive ratio of 0.46 for arbitrary patiences is due to Brubach et al. [15]. Fata et al. [24] improved this competitive ratio to 0.51 for the special case of **unbounded patience**. In an early 2020 arXiv version of this paper [8], we proved a competitive ratio of  $1 - 1/e$  for arbitrary patience values. All these previous competitive ratios (including ours) are proven against the offline adaptive benchmark. Theorem 3 generalizes this result, as it is the first to apply to non-identical distributions, as well as to more general probing constraints. Brubach et al. [11, 12] independently achieved a  $1 - 1/e$  competitive ratio for arbitrary patience values in the known i.i.d. setting, however their ratio is against the weaker online adaptive benchmark, and so is incomparable with previous results in the literature. Interestingly,  $1 - 1/e$  remains the best known competitive ratio in the prophet secretary matching problem due to Ehsani et al. [22], despite significant progress in the case of a single offline node (see [5, 20]). Huang et al. [32] very recently proved a 0.703 hardness result for multiple offline nodes and known i.i.d. arrivals.

In order to discuss the efficiency of our algorithms in the generality of our probing constraints, we work in the **membership oracle model**. An online probing algorithm may make a **membership query** to any string  $e \in \partial(b)^{(*)}$  for  $b \in B$ , thus determining in a single operation whether or not  $e \in \partial(b)^{(*)}$  is in  $\mathcal{C}_b$ . All our algorithms are implementable in polynomial time, as we prove in the full version of the paper (hereby denoted [9]).

A well studied special case of the online stochastic matching problem with known i.d. online arrivals is the case of a **known stochastic graph** (see [18, 1, 6, 2, 7, 27, 13, 35]). In this setting, the input  $H_{\text{typ}} = (V, B, F)$  satisfies  $n = |B|$ , and the distributions  $(\mathcal{D}_i)_{i=1}^n$  are all point-mass on *distinct* vertices of  $B$ . Thus, the online vertices of  $G$  are not randomly drawn, and  $G$  is instead equal to  $H_{\text{typ}}$ . The online probing algorithm thus knows the stochastic graph  $G$  in advance, but remains unaware of the edge states  $(\text{st}(e))_{e \in E}$ , and so it still must sequentially probe the edges to reveal their states. Again, it must operate in the probe-commit model, and respect the probing constraints  $(\mathcal{C}_v)_{v \in V}$  as well as the arrival order  $\pi$  on  $V$ .

► **Corollary 5** (of Theorem 3). *If  $\mathcal{M}$  is the matching returned by Algorithm 6 when presented the online vertices of  $G$  in random order, then  $\mathbb{E}[w(\mathcal{M})] \geq (1 - \frac{1}{e}) \text{OPT}(G)$ .*

► **Remark 6.** When Algorithm 9 executes in the known graph setting, it is **non-adaptive** in that its probes are a (randomized) function of  $G$ . In [9], we complement Corollary 5 with a  $1 - 1/e$  hardness result which applies to *all* non-adaptive probing algorithms (even probing algorithms which execute offline, and thus do not respect the arrival order  $\pi$  of  $V$ ).

► **Remark 7.** Gamblath et al. [27] consider an online probing algorithm when  $G$  is **unconstrained** – i.e.,  $\mathcal{C}_v = \partial(v)^{(*)}$  for all  $v \in V$  – and known to the algorithm. Both our algorithm and theirs attain a performance guarantee of  $1 - 1/e$  against very different non-standard LPs – LP-config and LP-QC, respectively. Note that LP-QC has exponentially many constraints and polynomially many variables, whereas LP-config has polynomially many constraints and exponentially many variables (see Appendix B for a statement of LP-QC). To the best of our knowledge, LP-QC does not seem to have an extension even to arbitrary patience values, as it is unclear how to generalize its constraints while maintaining polynomial time solvability. Despite having such different forms, in the unconstrained setting the LPs take

on the same value, as we prove in Proposition 28 of Appendix B. Thus, Theorem 3 can be viewed as a generalization of their work to downward-closed online probing constraints and known i.i.d. random order arrivals. Very recently, Pollner et al. [35] proved a 0.426 competitive ratio against the offline adaptive benchmark in the special case of a bipartite graph with (one-sided) patience values. Our results are incomparable, as their algorithm works for random order *edge arrivals*, whereas ours requires one-sided random order vertex arrivals, yet has a better competitive ratio and works for more general probing constraints.

## 2.1 An Overview of Our Techniques

For simplicity, we first describe our techniques in the known stochastic graph setting. Afterwards, we explain how our techniques extend to the known i.i.d. setting. Let us suppose that we are presented a stochastic graph  $G = (U, V, E)$ . For the case of patience values  $(\ell_v)_{v \in V}$ , a natural solution is to solve an LP introduced by Bansal et al. [6] to obtain fractional values for the edges of  $G$ , say  $(x_e)_{e \in E}$ , such that  $x_e$  upper bounds the probability  $e$  is probed by the offline adaptive benchmark. Clearly,  $\sum_{e \in \partial(v)} x_e \leq \ell_v$  is a constraint for each  $v \in V$ , and so by applying a dependent rounding algorithm (such as the GKSP algorithm of Gandhi et al. [28]), one can round the values  $(x_e)_{e \in \partial(v)}$  to determine  $\ell_v$  edges of  $\partial(v)$  to probe. By probing these edges in a carefully chosen order, and matching  $v$  to the first edge revealed to be active, one can guarantee that each  $e \in \partial(v)$  is matched with probability reasonably close to  $p_e x_e$ . This is the high-level approach used in many stochastic matching algorithms (for example [6, 2, 7, 15, 13, 35]). However, even for a single online node, this LP overestimates the value of the offline adaptive benchmark, and so any algorithm designed in this way will match certain edges with probability strictly less than  $p_e x_e$ . This is problematic, for the value of the match made to  $v$  is ultimately compared to  $\sum_{e \in \partial(v)} p_e w_e x_e$ , the contribution of the variables  $(x_e)_{e \in \partial(v)}$  to the LP solution. In fact, Fata et al. [24] showed that the ratio between  $\text{OPT}(G)$  and an optimum solution to this LP can be as small as 0.51, so the  $1 - 1/e$  competitive ratio of Theorem 3 cannot be achieved via a comparison to this LP, even for the special case of patience values.

**Defining LP-config.** Our approach is to work with a configuration LP (LP-config) which we initially called LP-new in our 2020 arXiv paper [8] and used in our companion paper [10] to attain an (optimal)  $1/e$  competitive ratio for the edge-weighted secretary matching problem in the probe-commit model. This LP has exponentially many variables which accounts for the many probing strategies available to an arriving vertex  $v$  with probing constraint  $\mathcal{C}_v$ . For each  $e \in E^{(*)}$ , define  $q(e) = \prod_{f \in e} (1 - p_f)$ , to be the probability that all the edges of  $e$  are inactive, where  $q(\lambda) := 1$  for the empty string/character  $\lambda$ . For  $f \in e$ , we denote  $e_{<f}$  to be the substring of  $e$  from its first edge up to, but not including,  $f$ . Observe then that  $\text{val}(e) := \sum_{f \in e} w_f \cdot p_f \cdot q(e_{<f})$  corresponds to the expected weight of the first active edge revealed if  $e$  is probed in order of its entries. For each  $v \in V$ , we introduce a decision variable  $x_v(e)$  and write the following LP:

$$\begin{aligned} &\text{maximize} && \sum_{v \in V} \sum_{e \in \mathcal{C}_v} \text{val}(e) \cdot x_v(e) && \text{(LP-config)} \end{aligned}$$

$$\begin{aligned} &\text{subject to} && \sum_{v \in V} \sum_{\substack{e \in \mathcal{C}_v: \\ (u,v) \in e}} p_{u,v} \cdot q(e_{<(u,v)}) \cdot x_v(e) \leq 1 && \forall u \in U \end{aligned} \quad (1)$$

$$\sum_{e \in \mathcal{C}_v} x_v(e) = 1 \quad \forall v \in V, \quad (2)$$

$$x_v(e) \geq 0 \quad \forall v \in V, e \in \mathcal{C}_v \quad (3)$$

In this work, we provide the first proof that LP-config is a **relaxation** of the offline adaptive benchmark. This result is stated but not proven in our companion paper, instead crediting the full arXiv version [9] of this paper. Unlike previous LPs used in the literature, we are not aware of an easy proof of this fact, and so we consider our proof to be a technical contribution.

► **Theorem 8.**  $OPT(G) \leq LPOPT(G)$ .

► **Remark 9.** For the case of patience values, a closely related LP was independently introduced by Brubach et al. [11, 12] to design probing algorithms for known i.i.d. arrivals and known i.d. adversarial arrivals. Their competitive ratios are proven against an optimal solution to this LP, which they argue relaxes the *online* adaptive benchmark.

When each  $\mathcal{C}_v$  is downward-closed, LP-config can be solved efficiently by using a deterministic separation oracle for the dual of LP-config, in conjunction with the ellipsoid algorithm [36, 29]. In [10], we introduce a greedy probing algorithm for offline vertex weights which attains  $1/2$  and  $1 - 1/e$  competitive ratios for adversarial and random order arrivals, respectively. These ratios are proven by applying the primal-dual method to a non-standard LP (distinct from LP-config). We also showed that this greedy probing algorithm can be used as a separation oracle for the dual of LP-config, as this ensures our  $1/e$ -competitive edge weights algorithm is efficient. For completeness, we provide the details for extending to the known i.d. case in [9], as well as a buyer/seller interpretation of the separation oracle problem.

**Proving Theorem 8.** In order to prove Theorem 8, the natural approach is to view  $x_v(e)$  as the probability that the offline adaptive benchmark probes the edges of  $e$  in order, where  $v \in V$  and  $e \in \mathcal{C}_v$ . Let us suppose that hypothetically we could make the following restrictive assumptions regarding the offline adaptive benchmark:

- $P_1$  If  $e = (u, v)$  is probed and  $st(e) = 1$ , then  $e$  is included in the matching, provided  $v$  is currently unmatched.
- $P_2$  For each  $v \in V$ , the edge probes involving  $\partial(v)$  are made independently of the edge states  $(st(e))_{e \in \partial(v)}$ .

Observe then that  $P_1$  and  $P_2$  would imply that the expected weight of the edge assigned to  $v$  is  $\sum_{e \in \mathcal{C}_v} \text{val}(e) \cdot x_v(e)$ . Moreover, the left-hand side of (1) would correspond to the probability  $u \in U$  is matched, so  $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$  would be a feasible solution to LP-config, and so we could upper bound  $OPT(G)$  by  $LPOPT(G)$ . Now, if we were working with the *online* adaptive benchmark, then it is clear that we could assume  $P_1$  and  $P_2$  simultaneously<sup>1</sup> w.l.o.g. On the other hand, if a probing algorithm does *not* respect an adaptive vertex ordering on  $V$  (i.e., does not probe edges in  $\partial(v)$  consecutively), then the probes involving  $v \in V$  will in general depend on  $(st(e))_{e \in \partial(v)}$ . For instance, if  $e \in \partial(v)$  is probed and inactive, then perhaps the offline adaptive benchmark next probes  $e' = (u, v') \in \partial(v')$  for some  $v' \neq v$ . If  $e'$  is active and thus added to the matching by  $P_1$ , then the offline adaptive benchmark can never subsequently probe  $(u, v)$  without violating  $P_1$ , as  $u$  is now unavailable to be matched to  $v$ . Thus, the natural interpretation of the decision variables of LP-config does not seem to easily lend itself to a proof of Theorem 8.

Our solution is to consider a **combinatorial relaxation** of the offline stochastic matching problem, which we define to be a new stochastic probing problem on  $G$  whose optimal value  $OPT_{\text{rel}}(G)$  satisfies  $OPT(G) \leq OPT_{\text{rel}}(G)$ . We refer to this problem as the **relaxed**

<sup>1</sup> It is clear that we may assume the offline adaptive benchmark satisfies  $P_1$  w.l.o.g., but not  $P_2$ .

**stochastic matching problem**, a solution to which is a **relaxed probing algorithm**. Roughly speaking, a relaxed probing algorithm operates in the same framework as an offline probing algorithm, yet it returns a one-sided matching of the online vertices which matches each offline node at most once *in expectation*. We provide a precise definition in Section 3. Crucially, there exists an *optimal* relaxed probing algorithm which is **non-adaptive** – that is, a (randomized) function of  $G$  – and which satisfies  $P_1$ . Non-adaptivity is a much stronger property than  $P_2$ , and so by the above discussion we are able to conclude that  $\text{OPT}_{\text{rel}}(G) \leq \text{LPOPT}(G)$ . Since  $\text{OPT}(G) \leq \text{OPT}_{\text{rel}}(G)$  by construction, this implies Theorem 8. Proving the existence of an optimal relaxed probing algorithm which is non-adaptive is one of the most technically challenging parts of the paper, and is the main content of Lemma 14 of Section 3. Note that there may be a simpler proof of Theorem 8, however our relaxed stochastic matching problem exactly characterizes LP-config (i.e.,  $\text{OPT}_{\text{rel}}(G) = \text{LPOPT}(G)$ ), and so it helps us understand LP-config. For instance, in Appendix B, we show that in the unconstrained patience setting, LP-QC of [27] is also characterized by our relaxed matching problem. This implies that the LPs take on the same value, despite having very different formulations in this special setting.

**Defining the probing algorithms:** After proving that LP-config is a relaxation of the offline adaptive benchmark, we use it to design online probing algorithms. Suppose that we are presented a feasible solution, say  $(x_v(\mathbf{e}))_{v \in V, \mathbf{e} \in \mathcal{C}_v}$ , to LP-config for  $G$ . For each  $e \in E$ , define

$$\tilde{x}_e := \sum_{\substack{\mathbf{e}' \in \mathcal{C}_v: \\ e \in \mathbf{e}'}} q(\mathbf{e}'_{<e}) \cdot x_v(\mathbf{e}'). \quad (4)$$

We refer to the values  $(\tilde{x}_e)_{e \in E}$  as the **edge variables** of the solution  $(x_v(\mathbf{e}))_{v \in V, \mathbf{e} \in \mathcal{C}_v}$ . If we now fix  $s \in V$ , then we can easily leverage constraint (2) to design a simple *fixed vertex* probing algorithm which matches each edge of  $e \in \partial(s)$  with probability *exactly* equal to  $p_e \tilde{x}_e$ . Specifically, draw  $\mathbf{e}' \in \mathcal{C}_s$  with probability  $x_s(\mathbf{e}')$ . If  $\mathbf{e}' = \lambda$ , then return the empty set. Otherwise, set  $\mathbf{e}' = (e'_1, \dots, e'_k)$  for  $k := |\mathbf{e}'| \geq 1$ , and probe the edges of  $\mathbf{e}'$  in order. Return the first edge which is revealed to be active, if such an edge exists. Otherwise, return the empty set. We refer to this algorithm as **VertexProbe**, and denote its output on the input  $(s, \partial(s), (x_s(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_s})$  by **VertexProbe** $(s, \partial(s), (x_s(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_s})$ .

► **Lemma 10.** *For each  $e \in \partial(s)$ ,  $\mathbb{P}[\text{VertexProbe}(s, \partial(s), (x_s(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_s}) = e] = p_e \tilde{x}_e$ .*

► **Remark 11.** We can view Lemma 10 as an **exact rounding guarantee**. The fact that such a guarantee exists, no matter the choice of  $\mathcal{C}_s$ , is one of the main benefits of working with LP-config, opposed to LP-std or LP-QC. As discussed, a solution to LP-std provably cannot be rounded exactly in this way. There *does* exist an exact rounding guarantee for LP-QC, however it only applies to the unconstrained setting of  $\mathcal{C}_v = \partial(s)^{(*)}$ , and the procedure is much more complicated than ours (see Theorem 29 of Appendix B for details).

► **Definition 12.** *We say that **VertexProbe** **commits** to the edge  $e = (u, s) \in \partial(s)$ , or equivalently the vertex  $u \in N(s)$ , provided the algorithm outputs  $e$  when executing on the fixed node  $s \in V$ . When it is clear that **VertexProbe** is being executed on  $s$ , we say that  $s$  commits to  $e$  (equivalently the vertex  $u$ ).*

Consider now the following online probing algorithm, where  $\pi$  is either u.a.r. or adversarial.

---

**Algorithm 1** Known Stochastic Graph.
 

---

**Require:** a stochastic graph  $G = (U, V, E)$ .

**Ensure:** a matching  $\mathcal{M}$  of active edges of  $G$ .

```

1: $\mathcal{M} \leftarrow \emptyset$.
2: Compute an optimal solution of LP-config for G , say $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$.
3: for $s \in V$ in order based on π do
4: Set $e \leftarrow \text{VertexProbe}(s, \partial(s), (x_s(e))_{e \in \mathcal{C}_s})$.
5: if $e = (u, s)$ for some $u \in U$, and u is unmatched then ▷ this line ensures $e \neq \emptyset$
6: Add e to \mathcal{M} .
7: end if
8: end for
9: return \mathcal{M} .

```

---

► **Remark 13.** Technically, line (6) should occur within the **VertexProbe** subroutine to adhere to the probe-commit model, however we express our algorithms in this way for conciseness.

**Improvement via online contention resolution.** Algorithm 1 does not attain a constant competitive ratio for adversarial arrivals, and its competitive ratio is only  $1/2$  in the random order arrivals. Thus, we must modify the algorithm to prove Theorems 1 and 3, even in the known stochastic graph setting. Our modification involves concurrently applying an appropriate rank one matroid **contention resolution scheme (CRS)** to each offline vertex of  $G$ , a concept formalized much more generally in the seminal paper by Chekuri, Vondrak, and Zenklusen [38]. Contention resolution has become a fundamental tool for stochastic optimization problems, and we illustrate its versatility by applying it to a non-standard LP.

Fix  $u \in U$ , and observe that constraint (1) ensures that  $\sum_{e \in \partial(u)} p_e \tilde{x}_e \leq 1$ . Moreover, if we set  $z_e := p_e \tilde{x}_e$ , then observe that as **VertexProbe** executes on  $v$ , each edge  $e = (u, v) \in \partial(u)$  is committed to  $u$  independently with probability  $z_e$ . On the other hand, there may be many edges which commit to  $u$  so we must resolve which one to take. In Algorithm 1,  $u$  is matched greedily to the first online vertex which commits to it, regardless of how  $\pi$  is generated. We apply existing **online** and **random order** contention resolution schemes to ensure that  $e$  is matched to  $u$  with probability  $1/2 \cdot z_e$  when  $\pi$  is generated by an adversary, and  $(1 - 1/e) \cdot z_e$  when  $\pi$  is generated u.a.r. These lower bounds on the edge variables allow us to conclude the desired competitive ratios, as  $\sum_{e \in E} w_e p_e \tilde{x}_e$  upper bounds  $\text{OPT}(G)$  by Theorem 8. We provide the specific schemes used for adversarial arrivals and random order arrivals in Section 4. In the latter setting, the CRS based approach simplifies the pricing based approach Gamlath et al. [27] used to attain a competitive ratio of  $1 - 1/e$  in the special unconstrained setting (see Remark 7). This simplified approach was also observed by Fu et al. [26] in the context of the Gamlath et al. LP (LP-QC). They focus on the unconstrained probe-commit model, and design a  $8/15$ -competitive algorithm for general graph random order vertex arrivals. It remains open whether their results can be extended to general patience values and random order edge arrivals. For context,  $0.395$  is the best known competitive ratio when allowing for arbitrary patience values and random order edge arrivals [35]. We focus on the bipartite graphs with one-sided arrivals, as the main goal of this paper was to fully resolve the complications posed by one-sided probing constraints in this arrival model.

**Extending to known i.d. arrivals.** In Appendix A, we prove Theorems 1 and 3 in their full generality when  $G$  is unknown and drawn from  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ . We do so by first generalizing LP-config to a new LP called LP-config-id. This LP departs from previous ones used in the

probing literature, as it depends both on the type graph as well as the distributions. For each  $i \in [n]$ , we introduce a collection of variables  $(x_i(e || b))_{e \in \mathcal{C}_b, b \in B}$  associated with the distribution  $\mathcal{D}_i$ . We again apply known contention resolution schemes, however the additional variables associated with the possible types of  $v_i \sim \mathcal{D}_i$  introduce correlated events which must be treated delicately in the context of **CRS selectibility**. Crucially, the schemes we employ do *not* make use of the type of vertex  $v_i$ , and so we are able to argue that analogous edge variable lower bounds hold as in the known stochastic graph setting.

### 3 Relaxing the Offline Adaptive Benchmark via LP-config

Given a stochastic graph  $G = (U, V, E)$ , we define the **relaxed stochastic matching problem**. A solution to this problem is a **relaxed probing algorithm**  $\mathcal{A}$ , which operates in the previously described framework of an (offline) probing algorithm. That is,  $\mathcal{A}$  is firstly given access to a stochastic graph  $G = (U, V, E)$ . Initially, the edge states  $(st(e))_{e \in E}$  are unknown to  $\mathcal{A}$ , and  $\mathcal{A}$  must adaptively probe these edges to reveal their states, while respecting the downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . As in the offline problem,  $\mathcal{A}$  returns a subset  $\mathcal{M}$  of its active edge probes, and its goal is to maximize  $\mathbb{E}[w(\mathcal{M})]$ , where  $w(\mathcal{M}) := \sum_{e \in \mathcal{M}} w_e$ . However, unlike before where  $\mathcal{M}$  was required to be a matching of  $G$ , we relax the required properties of  $\mathcal{M}$ :

1. Each  $v \in V$  appears in at most one edge of  $\mathcal{M}$ .
2. If  $N_u$  counts the number of edges of  $\partial(u)$  which are included in  $\mathcal{M}$ , then  $\mathbb{E}[N_u] \leq 1$  for each  $u \in U$ .

We refer to  $\mathcal{M}$  as a **one-sided matching** of the online nodes, and abuse terminology slightly and say that  $e \in E$  is matched by  $\mathcal{A}$  if  $e \in \mathcal{M}$ . In constructing  $\mathcal{M}$ ,  $\mathcal{A}$  must operate in the previously described probe-commit model. We define the **relaxed benchmark** as an optimal relaxed probing algorithm, and denote its expected value when executing on  $G$  by  $\text{OPT}_{\text{rel}}(G)$ . Observe that since any offline probing algorithm is a relaxed probing algorithm, we have that

$$\text{OPT}(G) \leq \text{OPT}_{\text{rel}}(G). \quad (5)$$

We say that  $\mathcal{A}$  is **non-adaptive**, provided the probes are a (randomized) function of  $G$ . Equivalently,  $\mathcal{A}$  is non-adaptive if the probes of  $\mathcal{A}$  are statistically independent from  $(st(e))_{e \in E}$ . Unlike for the offline stochastic matching problem, there exists a relaxed probing algorithm which is both optimal *and* non-adaptive:

► **Lemma 14.** *For any stochastic graph  $G = (U, V, E)$  with downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ , there exists an optimum relaxed probing algorithm  $\mathcal{B}$  which satisfies the following properties:*

$Q_1$  *If  $e = (u, v)$  is probed,  $st(e) = 1$ , and  $v$  was previously unmatched, then  $\mathcal{B}$  matches  $e$ .*

$Q_2$   *$\mathcal{B}$  is non-adaptive on  $G$ .*

► **Remark 15.** Note that  $Q_2$  implies the hypothetical property  $P_2$ , yet is much stronger. Let us assume that Lemma 14 holds for now.

**Proof of Theorem 8.** Consider  $\mathcal{B}$  of Lemma 14, and define  $x_v(e)$  to be the probability that  $\mathcal{B}$  probes the edges of  $e$  in order for  $v \in V$  and  $e \in \mathcal{C}_v$ . Since  $\mathcal{B}$  is a relaxed probing algorithm, we can apply properties  $Q_1$  and  $Q_2$  to show that  $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$  is a feasible solution to LP-config. Moreover, if  $\mathcal{N}$  is returned when  $\mathcal{B}$  executes on  $G$ , then

$$\mathbb{E}[w(\mathcal{N})] = \sum_{v \in V} \sum_{e \in \mathcal{C}_v} \text{val}(e) \cdot x_v(e).$$

Thus, the optimality of  $\mathcal{B}$  implies that  $\text{OPT}_{\text{rel}}(G) \leq \text{LPOPT}(G)$ , and so together with (5), Theorem 8 follows. ◀

► **Remark 16.** As mentioned, LP-config is an exact LP formulation of the relaxed stochastic matching problem, as we prove in Theorem 27 of Appendix B.

### 3.1 Proving Lemma 14

Let us suppose that  $G = (U, V, E)$  is a stochastic graph with downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . In order to prove Lemma 14, we must show that there exists an optimal relaxed probing algorithm which is non-adaptive and satisfies  $Q_1$ . Our high level approach is to consider an optimal relaxed probing algorithm  $\mathcal{A}$  which satisfies  $Q_1$ , and then to construct a new non-adaptive algorithm  $\mathcal{B}$  by *stealing* the strategy of  $\mathcal{A}$ , without any loss in performance. More specifically, we construct  $\mathcal{B}$  by writing down for each  $v \in V$  and  $e \in \mathcal{C}_v$  the probability that  $\mathcal{A}$  probes the edges of  $e$  in order. These probabilities necessarily satisfy certain inequalities which we make use of in designing  $\mathcal{B}$ . In order to do so, we need a technical randomized rounding procedure whose precise relevance will become clear in the proof of Lemma 14.

Suppose that  $e \in E^{(*)}$ , and recall that  $\lambda$  is the empty string/character. Let us now assume that  $(y_v(e))_{e \in \mathcal{C}_v}$  is a collection of non-negative values which satisfy  $y_v(\lambda) = 1$ , and

$$\sum_{\substack{e \in \partial(v): \\ (e', e) \in \mathcal{C}_v}} y_v(e', e) \leq y_v(e'), \quad (6)$$

for each  $e' \in \mathcal{C}_v$ . For space considerations, we defer the proof of the below proposition to [9].

► **Proposition 17.** *Given a collection of values  $(y_v(e))_{e \in \mathcal{C}_v}$  which satisfy  $y_v(\lambda) = 1$  and (6), there exists a distribution  $\mathcal{D}^v$  supported on  $\mathcal{C}_v$ , such that if  $\mathbf{Y} \sim \mathcal{D}^v$ , then for each  $e = (e_1, \dots, e_k) \in \mathcal{C}_v$  with  $k := |e| \geq 1$ , it holds that*

$$\mathbb{P}[(\mathbf{Y}_1, \dots, \mathbf{Y}_k) = (e_1, \dots, e_k)] = y_v(e), \quad (7)$$

where  $\mathbf{Y}_1, \dots, \mathbf{Y}_k$  are the first  $k$  characters of  $\mathbf{Y}$  (where  $\mathbf{Y}_i := \lambda$  if  $\mathbf{Y}$  has no  $i^{\text{th}}$  character).

**Proof of Lemma 14.** Suppose that  $\mathcal{A}$  is an optimal relaxed probing algorithm which returns the one-sided matching  $\mathcal{M}$  after executing on the stochastic graph  $G = (U, V, E)$ . In a slight abuse of terminology, we say that  $e$  is matched by  $\mathcal{A}$ , provided  $e$  is included in  $\mathcal{M}$ . We shall also make the simplifying assumption that  $p_e < 1$  for each  $e \in E$ , as the proof can be clearly adapted to handle the case when certain edges have  $p_e = 1$  by restricting which strings of each  $\mathcal{C}_v$  are considered.

Observe that since  $\mathcal{A}$  is optimal, it is clear that we may assume the following properties hold w.l.o.g. for each  $e \in E$ :

1.  $e$  is probed only if  $e$  can be added to the currently constructed one-sided matching.
2. If  $e$  is probed and  $\text{st}(e) = 1$ , then  $e$  is included in  $\mathcal{M}$ .

Thus, in order to prove the lemma, we must find an alternative algorithm  $\mathcal{B}$  which is non-adaptive, yet continues to be optimal. To this end, we shall first express  $\mathbb{E}[w(\mathcal{M}(v))]$  in a convenient form for each  $v \in V$ , where  $w(\mathcal{M}(v))$  is the weight of the edge matched to  $v$  (which is 0 if no match occurs).

Given  $v \in V$  and  $1 \leq i \leq |U|$ , we define  $X_i^v$  to be the  $i^{\text{th}}$  edge adjacent to  $v$  that is probed by  $\mathcal{A}$ . This is set equal to  $\lambda$  by convention, provided no such edge exists. We may then define  $\mathbf{X}^v := (X_1^v, \dots, X_{|U|}^v)$ , and  $\mathbf{X}_{\leq k}^v := (X_1^v, \dots, X_k^v)$  for each  $1 \leq k \leq |U|$ . Moreover, given  $e = (e_1, \dots, e_k) \in E^{(*)}$  with  $k \geq 1$ , define  $S(e)$  to be the event in which  $e_k$  is the only active edge amongst  $e_1, \dots, e_k$ . Observe then that



## 46:12 Prophet Matching in the Probe-Commit Model

$$\mathbb{E}[w(\mathcal{M}(v))] = \sum_{\substack{\mathbf{e}=(e_1,\dots,e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} \mathbb{P}[S(\mathbf{e}) \cap \{\mathbf{X}_{\leq k}^v = \mathbf{e}\}],$$

as (1) and (2) ensure  $v$  is matched to the first probed edge which is revealed to be active. Moreover, if  $\mathbf{e} = (e_1, \dots, e_k) \in \mathcal{C}_v$  for  $k \geq 2$ , then

$$\mathbb{P}[S(\mathbf{e}) \cap \{\mathbf{X}_{\leq k}^v = \mathbf{e}\}] = \mathbb{P}[\{\text{st}(e_k) = 1\} \cap \{\mathbf{X}_{\leq k}^v = \mathbf{e}\}], \quad (8)$$

as (1) and (2) ensure  $\mathbf{X}_{\leq k}^v = \mathbf{e}$  only if  $e_1, \dots, e_{k-1}$  are inactive. Thus,

$$\begin{aligned} \mathbb{E}[w(\mathcal{M}(v))] &= \sum_{\substack{\mathbf{e}=(e_1,\dots,e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} \mathbb{P}[S(\mathbf{e}) \cap \{\mathbf{X}_{\leq k}^v = \mathbf{e}\}] \\ &= \sum_{\substack{\mathbf{e}=(e_1,\dots,e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} \mathbb{P}[\{\text{st}(e_k) = 1\} \cap \{\mathbf{X}_{\leq k}^v = \mathbf{e}\}] \\ &= \sum_{\substack{\mathbf{e}=(e_1,\dots,e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} p_{e_k} \mathbb{P}[\mathbf{X}_{\leq k}^v = \mathbf{e}], \end{aligned}$$

where the final equality holds since  $\mathcal{A}$  must decide on whether to probe  $e_k$  prior to revealing  $\text{st}(e_k)$ . As a result, after summing over  $v \in V$ ,

$$\mathbb{E}[w(\mathcal{M})] = \sum_{v \in V} \sum_{\substack{\mathbf{e}=(e_1,\dots,e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} p_{e_k} \mathbb{P}[\mathbf{X}_{\leq k}^v = \mathbf{e}]. \quad (9)$$

Our goal is to find a non-adaptive relaxed probing algorithm which matches the value of (9). Thus, for each  $v \in V$  and  $\mathbf{e} = (e_1, \dots, e_k) \in \mathcal{C}_v$  with  $k \geq 1$ , define  $x_v(\mathbf{e}) := \mathbb{P}[\mathbf{X}_{\leq k}^v = \mathbf{e}]$ , where  $x_v(\lambda) := 1$ . Observe now that for each  $\mathbf{e}' = (e'_1, \dots, e'_k) \in \mathcal{C}_v$ ,

$$\sum_{\substack{\mathbf{e} \in \partial(v): \\ (\mathbf{e}', \mathbf{e}) \in \mathcal{C}_v}} \mathbb{P}[\mathbf{X}_{\leq k+1}^v = (\mathbf{e}', \mathbf{e}) \mid \mathbf{X}_{\leq k}^v = \mathbf{e}'] \leq 1 - p_{e'_k}. \quad (10)$$

To see (10), observe that the left-hand side corresponds to the probability  $\mathcal{A}$  probes some edge  $\mathbf{e} \in \partial(v)$ , given it already probed  $\mathbf{e}'$  in order. On the other hand, if a subsequent edge is probed, then (1) and (2) imply that  $e'_k$  must have been inactive, which occurs independently of the event  $\mathbf{X}_{\leq k}^v = \mathbf{e}'$ . This explains the right-hand side of (10). Using (10), the values  $(x_v(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_v}$  satisfy

$$\sum_{\substack{\mathbf{e} \in \partial(v): \\ (\mathbf{e}', \mathbf{e}) \in \mathcal{C}_v}} x_v(\mathbf{e}', \mathbf{e}) \leq (1 - p_{e'_k}) \cdot x_v(\mathbf{e}'), \quad (11)$$

for each  $\mathbf{e}' = (e'_1, \dots, e'_k) \in \mathcal{C}_v$  with  $k \geq 1$ . Moreover, clearly  $\sum_{\mathbf{e} \in \partial(v)} x_v(\mathbf{e}) \leq 1$ .

Given  $\mathbf{e} = (e_1, \dots, e_k) \in \mathcal{C}_v$  for  $k \geq 1$ , recall that  $\mathbf{e}_{<k} := (e_1, \dots, e_{k-1})$  where  $\mathbf{e}_{<1} := \lambda$  if  $k = 1$ . Moreover,  $q(\mathbf{e}_{<k}) := \prod_{i=1}^{k-1} (1 - p_{e_i})$ , where  $q(\lambda) := 1$ . Using this notation, define for each  $\mathbf{e} \in \mathcal{C}_v$

$$y_v(\mathbf{e}) := \begin{cases} x_v(\mathbf{e})/q(\mathbf{e}_{<|\mathbf{e}|}) & \text{if } |\mathbf{e}| \geq 1, \\ 1 & \text{otherwise.} \end{cases} \quad (12)$$

Observe that (11) ensures that for each  $e' \in \mathcal{C}_v$ ,

$$\sum_{\substack{e \in \partial(v): \\ (e', e) \in \mathcal{C}_v}} y_v(e', e) \leq y_v(e'), \quad (13)$$

and  $y_v(\lambda) := 1$ . As a result, Proposition 17 implies that for each  $v \in V$ , there exists a distribution  $\mathcal{D}^v$  such that if  $\mathbf{Y}^v \sim \mathcal{D}^v$ , then for each  $e \in \mathcal{C}_v$  with  $|e| = k \geq 1$ ,

$$\mathbb{P}[\mathbf{Y}_{\leq k}^v = e] = y_v(e). \quad (14)$$

Moreover,  $\mathbf{Y}^v$  is drawn independently from the edge states,  $(\text{st}(e))_{e \in E}$ . Consider now the following algorithm  $\mathcal{B}$ , which satisfies the desired properties  $Q_1$  and  $Q_2$  of Lemma 14:

■ **Algorithm 2** Algorithm  $\mathcal{B}$ .

**Require:** a stochastic graph  $G = (U, V, E)$ .

**Ensure:** a one-sided matching  $\mathcal{N}$  of  $G$  of active edges.

```

1: Set $\mathcal{N} \leftarrow \emptyset$.
2: Draw $(\mathbf{Y}^v)_{v \in V}$ according to the product distribution $\prod_{v \in V} \mathcal{D}^v$.
3: for $v \in V$ do
4: for $i = 1, \dots, |\mathbf{Y}^v|$ do
5: Set $e \leftarrow \mathbf{Y}_i^v$. $\triangleright \mathbf{Y}_i^v$ is the i^{th} edge of \mathbf{Y}^v
6: Probe the edge e , revealing $\text{st}(e)$.
7: if $\text{st}(e) = 1$ and v is unmatched by \mathcal{N} then
8: Add e to \mathcal{N} .
9: end if
10: end for
11: end for
12: return \mathcal{N} .
```

Using (14) and the non-adaptivity of  $\mathcal{B}$ , it is clear that for each  $v \in V$ ,

$$\begin{aligned} \mathbb{E}[w(\mathcal{N}(v))] &= \sum_{\substack{e=(e_1, \dots, e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} \mathbb{P}[S(e)] \cdot \mathbb{P}[\mathbf{Y}_{\leq k}^v = e] \\ &= \sum_{\substack{e=(e_1, \dots, e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} p_{e_k} q(e_{<k}) y_v(e) \\ &= \sum_{\substack{e=(e_1, \dots, e_k) \in \mathcal{C}_v: \\ k \geq 1}} w_{e_k} p_{e_k} x_v(e) = \mathbb{E}[w(\mathcal{M}(v))]. \end{aligned}$$

Thus, after summing over  $v \in V$ , it holds that  $\mathbb{E}[w(\mathcal{N})] = \mathbb{E}[w(\mathcal{M})] = \text{OPT}_{\text{rel}}(G)$ , and so in addition to satisfying  $Q_1$  and  $Q_2$ ,  $\mathcal{B}$  is optimal. Finally, it is easy to show that each  $u \in U$  is matched by  $\mathcal{N}$  at most once in expectation since  $\mathcal{M}$  has this property. Thus,  $\mathcal{B}$  is a relaxed probing algorithm which is optimal and satisfies the required properties of Lemma 14. ◀

## 4 Proving Theorems 1 and 3 for a Known Stochastic Graph

Given  $k \geq 1$ , consider the ground set  $[k] := \{1, \dots, k\}$ , and  $\mathcal{P} := \{z \in [0, 1]^k : \sum_{i=1}^k z_i \leq 1\}$ . Fix  $z \in \mathcal{P}$ , and let  $R(z) \subseteq [k]$  denote the random set where each  $i \in [k]$  is included in  $R(z)$  independently with probability  $z_i$ . Feldman et al. [25] considered a restricted class of

contention resolution schemes called **online contention resolution schemes** (OCRS). The elements of  $[k]$  are presented to the OCRS  $\psi$  in adversarial order, where in each step, an arriving  $i \in [k]$  reveals if it is in  $R(\mathbf{z})$ , at which point  $\psi$  must make an irrevocable decision as to whether it wishes to return  $i$  as its output. We refer the reader to [9] for a brief overview of CRS terminology.

Suppose the elements of  $[k]$  arrive according to some permutation  $\sigma : [k] \rightarrow [k]$  (i.e.,  $\sigma(1), \dots, \sigma(k)$ ), and  $\mathbf{z} \in [0, 1]^k$  satisfies  $\sum_{i=1}^k z_i \leq 1$ . Upon the arrival of element  $\sigma(t) \in [k]$ , compute  $q_t := \left(2 - \sum_{i=1}^{t-1} z_{\sigma(i)}\right)^{-1}$ . Observe that  $1/2 \leq q_t \leq 1$ , as  $0 \leq \sum_{i=1}^k z_i \leq 1$ , and so the following OCRS is well-defined:

■ **Algorithm 3** OCRS – Ezra et al. [23].

---

**Require:**  $\mathbf{z} = (z_1, \dots, z_k) \in \mathcal{P}$ .

**Ensure:** at most one element of  $[k]$ .

```

1: for $t = 1, \dots, k$ do
2: if $\sigma(t) \in R(\mathbf{z})$ then
3: Compute q_t based on the arrivals $\sigma(1), \dots, \sigma(t-1)$.
4: return $\sigma(t)$ independently with probability q_t .
5: end if
6: end for
7: return \emptyset . ▷ pass on returning an element of $[k]$

```

---

► **Theorem 18** (Ezra et al. [34]). *Algorithm 3 is an OCRS which is  $1/2$ -selectable.*

Both Lee and Singla [34], as well as Adamczyk and Włodarczyk [3], defined a special type of CRS called a **random order contention resolution scheme** (RCRS). Such a CRS is defined in the same way as an OCRS, except that the elements of  $[k]$  arrive u.a.r. Suppose  $Y_i \sim [0, 1]$  u.a.r. and independently for  $i = 1, \dots, k$ .

■ **Algorithm 4** RCRS – Lee and Singla [34].

---

**Require:**  $\mathbf{z} = (z_1, \dots, z_k) \in \mathcal{P}$ .

**Ensure:** at most one element of  $[k]$ .

```

1: for $i \in [k]$ in increasing order of Y_i do
2: if $i \in R(\mathbf{z})$ then
3: return i independently with probability $\exp(-Y_i \cdot z_i)$
4: end if
5: end for
6: return \emptyset . ▷ pass on returning an element of $[k]$

```

---

► **Theorem 19** (Lee and Singla [34]). *Algorithm 4 is a  $1 - 1/e$ -selectable RCRS.*

Suppose now  $G = (U, V, E)$  is a known stochastic graph, whose online vertices  $v_1, \dots, v_n$  are presented according to the below algorithm via an adversarially chosen permutation  $\pi : [n] \rightarrow [n]$  (i.e.,  $v_{\pi(1)}, \dots, v_{\pi(n)}$ ). Let  $(x_v(\mathbf{e}))_{v \in V, \mathbf{e} \in \mathcal{C}_v}$  be an optimum solution to LP-config for  $G$  with edge variables  $(\tilde{x}_e)_{e \in E}$ . For each  $t \in [n]$  and  $u \in U$ , define  $q_{u,t} := \left(2 - \sum_{i=1}^{t-1} z_{u, v_{\pi(i)}}\right)^{-1}$ , where  $z_e := p_e \tilde{x}_e$  for  $e \in E$ , and  $q_{u,1} := 1/2$ . Clearly,  $\sum_{v \in V} z_{u,v} \leq 1$ , by constraint (1) of LP-config, and so  $1/2 \leq q_{u,t} \leq 1$ :

---

**Algorithm 5** Known Stochastic Graph – AOM – Modified.

---

**Require:** a stochastic graph  $G = (U, V, E)$ .

**Ensure:** a matching  $\mathcal{M}$  of  $G$  of active edges.

```

1: $\mathcal{M} \leftarrow \emptyset$.
2: Compute an optimum solution of LP-config for G , say $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$.
3: for $t = 1, \dots, n$ do
4: Based on the previous arrivals $v_{\pi(1)}, \dots, v_{\pi(t-1)}$ before $v_{\pi(t)}$, compute values $(q_{u,t})_{u \in U}$.
5: Set $e \leftarrow \text{VertexProbe}(v_{\pi(t)}, \partial(v_{\pi(t)}), (x_{v_{\pi(t)}}(e))_{e \in \mathcal{C}_{v_{\pi(t)}}})$.
6: if $e = (u, v_{\pi(t)})$ for some $u \in U$, and u is unmatched then
7: Add e to \mathcal{M} independently with probability $q_{u,t}$. ▷ OCRS is used here
8: end if
9: end for
10: return \mathcal{M} .

```

---

► **Proposition 20.** *Algorithm 5 is  $1/2$ -competitive against adversarial arrivals.*

**Proof.** Given  $u \in U$ , let  $\mathcal{M}(u)$  denote the edge matched to  $u$  by  $\mathcal{M}$ , where  $\mathcal{M}(u) := \emptyset$  if no such edge exists. Observe now that if  $C(e)$  corresponds to the event in which **VertexProbe** commits to  $e \in \partial(u)$ , then  $\mathbb{P}[C(e)] = p_e \tilde{x}_e$  by Lemma 10. Moreover, the events  $(C(e))_{e \in \partial(u)}$  are independent, and satisfy

$$\sum_{e \in \partial(u)} \mathbb{P}[C(e)] = \sum_{e \in \partial(u)} p_e \tilde{x}_e \leq 1, \quad (15)$$

by constraint (1) of LP-config. As such, denote  $\mathbf{z} := (z_e)_{e \in \partial(u)}$  where  $z_e = p_e \tilde{x}_e$ , and observe that (15) ensures that  $\mathbf{z} \in \mathcal{P}$ , where  $\mathcal{P}$  is the convex relaxation of the rank one matroid on  $\partial(u)$ . Let us denote  $R(\mathbf{z})$  as those  $e \in \partial(u)$  for which  $C(e)$  occurs.

If  $\psi$  is the OCRS defined in Algorithm 3, then we may pass  $\mathbf{z}$  to  $\psi$ , and process the edges of  $\partial(u)$  in the order induced by  $\pi$ . Denote the resulting output by  $\psi_{\mathbf{z}}(R(\mathbf{z}))$ . By coupling the random draws of lines (4) and (7) of Algorithms 3 and 5, respectively, we get that

$$w(\mathcal{M}(u)) = \sum_{e \in \partial(u)} w_e \cdot \mathbf{1}_{[e \in R(\mathbf{z})]} \cdot \mathbf{1}_{[e \in \psi_{\mathbf{z}}(R(\mathbf{z}))]}$$

Thus, after taking expectations,

$$\mathbb{E}[w(\mathcal{M}(u))] = \sum_{e \in \partial(u)} w_e \cdot \mathbb{P}[e \in \psi_{\mathbf{z}}(R(\mathbf{z})) \mid e \in R(\mathbf{z})] \cdot \mathbb{P}[e \in R(\mathbf{z})].$$

Now, Theorem 18 ensures that for each  $e \in \partial(u)$ ,  $\mathbb{P}[e \in \psi_{\mathbf{z}}(R(\mathbf{z})) \mid e \in R(\mathbf{z})] \geq 1/2$ . It follows that  $\mathbb{E}[w(\mathcal{M}(u))] \geq \frac{1}{2} \sum_{e \in \partial(u)} w_e p_e \tilde{x}_e$ , for each  $u \in U$ . Thus,

$$\mathbb{E}[w(\mathcal{M})] = \sum_{u \in U} \mathbb{E}[w(\mathcal{M}(u))] \geq \frac{1}{2} \sum_{e \in E} w_e p_e \tilde{x}_e = \frac{\text{LPOPT}(G)}{2},$$

where the equality follows since  $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$  is an optimum solution to LP-config. On the other hand,  $\text{LPOPT}(G) \geq \text{OPT}(G)$  by Theorem 8, and so the proof is complete. ◀

For each  $v \in V$ , draw  $\tilde{Y}_v \in [0, 1]$  independently and u.a.r. We assume that the vertices of  $V$  are presented to the below online probing algorithm in non-decreasing order according to the values  $(\tilde{Y}_v)_{v \in V}$ . Note that this is equivalent to presenting  $V$  to the algorithm in random order.

■ **Algorithm 6** Known Stochastic Graph – ROM– Modified.

**Require:** a stochastic graph  $G = (U, V, E)$ .

**Ensure:** a matching  $\mathcal{M}$  of  $G$  of active edges.

---

```

1: $\mathcal{M} \leftarrow \emptyset$.
2: Compute an optimum solution of LP-config for G , say $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$.
3: for $s \in V$ in increasing order of \tilde{Y}_s do
4: Set $e \leftarrow \text{VertexProbe}(s, \partial(s), (x_s(e))_{e \in \mathcal{C}_s})$.
5: if $e = (u, s)$ for some $u \in U$, and u is unmatched then
6: Add e to \mathcal{M} independently with probability $\exp(-\tilde{Y}_s \cdot p_{u,s} \cdot \tilde{x}_{u,s})$.
7: end if
8: end for
9: return \mathcal{M} .

```

---

► **Proposition 21** (Restatement of Corollary 5 and Remark 6). *Algorithm 6 is non-adaptive and  $1 - 1/e$ -competitive against random order arrivals.*

Algorithm 6 is clearly non-adaptive, and the proof that it is  $1 - 1/e$ -competitive follows similarly to the proof of Proposition 20 (see [9] for the details).

## 5 Open problems

There are some basic questions that are unresolved. Perhaps the most basic question which is also unresolved in the classical setting without probing is to bridge the gap between the positive  $1 - 1/e$  competitive ratio and in-approximations in the context of known i.i.d. random order arrivals. In terms of the single item prophet secretary problem (without probing), Correa et al. [20] obtain a 0.669 competitive ratio following Azar et al. [5] who were the first to surpass the  $1 - 1/e$  “barrier”. Correa et al. [20] also establish a 0.732 in-approximation for the i.i.d. setting, and Huang et al. [32] recently established a 0.703 in-approximation for i.i.d. arrivals in the multi-item case. Can we surpass  $1 - 1/e$  in the probing setting for i.i.d. input arrivals or for the special case of i.i.d. input arrivals? Is there a provable difference between stochastic bipartite matching (with probing constraints) and the classical online settings? Can we obtain the same competitive results against an optimal offline *non-committal* benchmark which respects the probing constraints but doesn’t operate in the probe-commit model? The 0.51 in-approximation result of Fata et al. [24] suggests that 0.51 may be the optimal competitive ratio against this stronger benchmark.

One interesting extension of the probing model is to allow non-Bernoulli edge random variables to describe edge uncertainty. Even for a single online vertex in the unconstrained setting, this problem is interesting as it corresponds to computing an optimal policy for the **free-order prophets problem**, which was recently studied by Segev and Singla in [37].

---

## References

- 1 Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Inf. Process. Lett.*, 111(15):731–737, 2011.
- 2 Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2015.

- 3 Marek Adamczyk and Michał Włodarczyk. Random order contention resolution schemes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 790–801. IEEE, 2018.
- 4 Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC '12*, pages 18–35, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2229012.2229018.
- 5 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Prophet secretary: Surpassing the  $1-1/e$  barrier. In Éva Tardos, Edith Elkind, and Rakesh Vohra, editors, *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 303–318. ACM, 2018.
- 6 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012. doi:10.1007/s00453-011-9511-8.
- 7 Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. Improved bounds in stochastic matching and optimization. *Algorithmica*, 80(11):3225–3252, November 2018. doi:10.1007/s00453-017-0383-4.
- 8 Allan Borodin, Calum MacRury, and Akash Rakheja. Bipartite stochastic matching: Online, random order, and I.I.D. models. *CoRR*, abs/2004.14304, 2020. arXiv:2004.14304.
- 9 Allan Borodin, Calum MacRury, and Akash Rakheja. Prophet matching meets probing with commitment. *CoRR*, abs/2102.04325, 2021. arXiv:2102.04325.
- 10 Allan Borodin, Calum MacRury, and Akash Rakheja. Secretary matching meets probing with commitment. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 13:1–13:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 11 Brian Brubach, Nathaniel Grammel, Will Ma, and Aravind Srinivasan. Follow your star: New frameworks for online stochastic matching with known and unknown patience. *CoRR*, abs/1907.03963, 2021. arXiv:1907.03963.
- 12 Brian Brubach, Nathaniel Grammel, Will Ma, and Aravind Srinivasan. Follow your star: New frameworks for online stochastic matching with known and unknown patience. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2872–2880. PMLR, 13–15 April 2021. URL: <http://proceedings.mlr.press/v130/brubach21a.html>.
- 13 Brian Brubach, Nathaniel Grammel, Will Ma, and Aravind Srinivasan. Improved guarantees for offline stochastic matching via new ordered contention resolution schemes. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27184–27195, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/e43739bba7cdb577e9e3e4e42447f5a5-Abstract.html>.
- 14 Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New algorithms, better bounds, and a novel model for online stochastic matching. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 24:1–24:16, 2016.
- 15 Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Attenuate locally, win globally: Attenuation-based frameworks for online stochastic matching with timeouts. *Algorithmica*, 82(1):64–87, 2020. doi:10.1007/s00453-019-00603-7.

- 16 Constantine Caramanis, Paul Dütting, Matthew Faw, Federico Fusco, Philip Lazos, Stefano Leonardi, Orestis Papadigenopoulos, Emmanouil Pountourakis, and Rebecca Reiffenhäuser. Single-sample prophet inequalities via greedy-ordered selection. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 1298–1325, 2022. doi:10.1137/1.9781611977073.54.
- 17 Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 311–320. ACM, 2010.
- 18 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I, ICALP '09*, pages 266–278, 2009.
- 19 José R. Correa, Patricio Foncea, Dana Pizarro, and Victor Verdugo. From pricing to prophets, and back! *Oper. Res. Lett.*, 47(1):25–29, 2019.
- 20 José R. Correa, Raimundo Saona, and Bruno Ziliotto. Prophet secretary through blind strategies. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1946–1961, 2019.
- 21 Kevin P. Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 822–833, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- 22 Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '18*, pages 700–714, USA, 2018. Society for Industrial and Applied Mathematics.
- 23 Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In *Proceedings of the 21st ACM Conference on Economics and Computation, EC '20*, pages 769–787, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3391403.3399513.
- 24 Elaheh Fata, Will Ma, and David Simchi-Levi. Multi-stage and multi-customer assortment optimization with inventory constraints. *CoRR*, abs/1908.09808, 2019. arXiv:1908.09808.
- 25 Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1014–1033. SIAM, 2016. doi:10.1137/1.9781611974331.ch72.
- 26 Hu Fu, Zhihao Gavin Tang, Hongxun Wu, Jinzhao Wu, and Qianfan Zhang. Random Order Vertex Arrival Contention Resolution Schemes for Matching, with Applications. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.68.
- 27 Buddhima Gamlath, Sagar Kale, and Ola Svensson. Beating greedy for stochastic bipartite matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '19*, pages 2841–2854, USA, 2019. Society for Industrial and Applied Mathematics.
- 28 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, May 2006. doi:10.1145/1147954.1147956.
- 29 Bernd Gärtner and Jirí Matousek. *Understanding and using linear programming*. Universitext. Springer, 2007.



- 30 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1731–1747. SIAM, 2016. doi:10.1137/1.9781611974331.ch120.
- 31 Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 58–65. AAAI Press, 2007.
- 32 Zhiyi Huang, Xinkai Shu, and Shuyi Yan. The power of multiple choices in online stochastic matching, 2022. doi:10.48550/ARXIV.2203.02883.
- 33 Haim Kaplan and David Naor dand Danny Raz. Online weighted bipartite matching with a sample. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, 2022. doi:10.1137/1.9781611977073.52.
- 34 Euiwoong Lee and Sahil Singla. Optimal Online Contention Resolution Schemes via Ex-Ante Prophet Inequalities. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.57.
- 35 Tristan Pollner, Mohammad Roghani, Amin Saberi, and David Wajc. Improved online contention resolution for matchings and applications to the gig economy. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 321–322. ACM, 2022. doi:10.1145/3490486.3538295.
- 36 D. Seese. Groetschel, m., l. lovasz, a. schrijver: Geometric algorithms and combinatorial optimization. (algorithms and combinatorics. eds.: R. l. graham, b. korte, l. lovasz. vol. 2), springer-verlag 1988, xii, 362 pp., 23 figs., dm 148,-. isbn 3-540-13624-x. *Biometrical Journal*, 32(8):930-930, 1990. doi:10.1002/bimj.4710320805.
- 37 Danny Segev and Sahil Singla. Efficient approximation schemes for stochastic probing and prophet problems. In *Proceedings of the 22nd ACM Conference on Economics and Computation, EC '21*, pages 793–794, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465456.3467614.
- 38 Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC '11*, pages 783–792, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993740.

## A Extending to Known I.D. Arrivals

Suppose that  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$  is a known i.d. input, where  $H_{\text{typ}} = (U, B, F)$  has downward-closed online probing constraints  $(\mathcal{C}_b)_{b \in B}$ . If  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ , where  $G = (U, V, E)$  has vertices  $V = \{v_1, \dots, v_n\}$ , then define  $r_i(b) := \mathbb{P}[v_i = b]$  for each  $i \in [n]$  and  $b \in B$ , where we hereby assume that  $r_i(b) > 0$ . We generalize LP-config to account for the distributions  $(\mathcal{D}_i)_{i=1}^n$ . For each  $i \in [n]$ ,  $b \in B$  and  $e \in \mathcal{C}_b$ , we introduce a decision variable  $x_i(e \parallel b)$  to encode the probability that  $v_i$  has type  $b$  and  $e$  is the sequence of edges of  $\partial(v_i)$  probed by the *relaxed* benchmark.

$$\text{maximize} \quad \sum_{i \in [n], b \in B, \mathbf{e} \in \mathcal{C}_b} \text{val}(\mathbf{e}) \cdot x_i(\mathbf{e} \parallel b) \quad (\text{LP-config-id})$$

$$\text{subject to} \quad \sum_{i \in [n], b \in B} \sum_{\substack{\mathbf{e} \in \mathcal{C}_b: \\ (u, b) \in \mathbf{e}}} p_{u, b} \cdot q(\mathbf{e}_{<(u, b)}) \cdot x_i(\mathbf{e} \parallel b) \leq 1 \quad \forall u \in U \quad (16)$$

$$\sum_{\mathbf{e} \in \mathcal{C}_b} x_i(\mathbf{e} \parallel b) = r_i(b) \quad \forall b \in B, i \in [n] \quad (17)$$

$$x_i(\mathbf{e} \parallel b) \geq 0 \quad \forall b \in B, \mathbf{e} \in \mathcal{C}_b, i \in [n] \quad (18)$$

Let us denote  $\text{LPOPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$  as the value of an optimum solution to LP-config-id.

► **Theorem 22.**  $\text{OPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n) \leq \text{LPOPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .

One way to prove Theorem 22 is to use the properties of the relaxed benchmark on  $G$  guaranteed by Lemma 14, and the above interpretation of the decision variables to argue that  $\mathbb{E}[\text{OPT}_{\text{rel}}(G)] \leq \text{LPOPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ , where  $\text{OPT}_{\text{rel}}(G)$  is the value of the relaxed benchmark on  $G$ . Specifically, we can interpret (16) as saying that the relaxed benchmark matches each offline vertex at most once in expectation. Moreover, (17) holds by observing that if  $v_i$  is of type  $b$ , then the relaxed benchmark selects some  $\mathbf{e} \in \mathcal{C}_b$  to probe (note  $\mathbf{e}$  could be the empty-string). We provide a morally equivalent proof of Theorem 22 in [9]. Specifically, we consider an optimum solution of LP-config with respect to  $G$ , and apply a conditioning argument in conjunction with Theorem 8.

Given a feasible solution to LP-config-id, say  $(x_i(\mathbf{e} \parallel b))_{i \in [n], b \in B, \mathbf{e} \in \mathcal{C}_b}$ , for each  $u \in U, i \in [n]$  and  $b \in B$  define

$$\tilde{x}_{u, i}(b) := \sum_{\substack{\mathbf{e} \in \mathcal{C}_b: \\ (u, b) \in \mathbf{e}}} q(\mathbf{e}_{<(u, b)}) \cdot x_i(\mathbf{e} \parallel b). \quad (19)$$

We refer to  $\tilde{x}_{u, i}(b)$  as an **edge variable**, thus extending the definition from the known stochastic graph setting. Suppose now that we fix  $i \in [n]$  and  $b \in B$ , and consider the variables,  $(x_i(\mathbf{e} \parallel b))_{\mathbf{e} \in \mathcal{C}_b}$ . Observe that (17) ensures that  $\frac{\sum_{\mathbf{e} \in \mathcal{C}_b} x_i(\mathbf{e} \parallel b)}{r_i(b)} = 1$ . Hence, regardless of which

type node  $v_i$  is drawn as,  $\frac{\sum_{\mathbf{e} \in \mathcal{C}_{v_i}} x_i(\mathbf{e} \parallel v_i)}{r_i(v_i)} = 1$ . We can therefore generalize **VertexProbe** as follows. Given vertex  $v_i$ , draw  $\mathbf{e}' \in \mathcal{C}_{v_i}$  with probability  $x_i(\mathbf{e}' \parallel v_i)/r_i(v_i)$ . If  $\mathbf{e}' = \lambda$ , then return the empty-set. Otherwise, set  $\mathbf{e}' = (e'_1, \dots, e'_k)$  for  $k := |\mathbf{e}'| \geq 1$ , and probe the edges of  $\mathbf{e}'$  in order. Return the first edge which is revealed to be active, if such an edge exists. Otherwise, return the empty-set. We denote the output of **VertexProbe** on the input  $(v_i, \partial(v_i), (x_i(\mathbf{e} \parallel v_i)/r_i(v_i))_{\mathbf{e} \in \mathcal{C}_{v_i}})$  by  $\text{VertexProbe}(v_i, \partial(v_i), (x_i(\mathbf{e} \parallel v_i)/r_i(v_i))_{\mathbf{e} \in \mathcal{C}_{v_i}})$ . Define  $C(u, v_i)$  as the event in which **VertexProbe** outputs the edge  $(u, v_i)$ , and observe the following extension of Lemma 10:

► **Lemma 23.** *If **VertexProbe** is passed  $(v_i, \partial(v_i), (x_i(\mathbf{e} \parallel v_i)/r_i(v_i))_{\mathbf{e} \in \mathcal{C}_{v_i}})$ , then for any  $b \in B$  and  $u \in U$ ,  $\mathbb{P}[C(u, v_i) \mid v_i = b] = \frac{p_{u, b} \cdot \tilde{x}_{u, i}(b)}{r_i(b)}$ .*

► **Remark 24.** As in Definition 12, if  $C(u, v_i)$  occurs, then  $u$  **commits** to  $(u, v_i)$  (or  $v_i$ ).

We now generalize Algorithm 1 where  $\pi$  is generated either u.a.r. or adversarially.

---

**Algorithm 7** Known I.D.
 

---

**Require:** a known i.d. input  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .

**Ensure:** a matching  $\mathcal{M}$  of active edges of  $G \sim (H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .

```

1: $\mathcal{M} \leftarrow \emptyset$.
2: Compute an optimum solution of LP-config-id for $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$, say
 $(x_i(e \parallel b))_{i \in [n], b \in B, e \in \mathcal{C}_b}$.
3: for $t = 1, \dots, n$ do
4: Let $a \in B$ be the type of the current arrival $v_{\pi(t)}$. ▷ to simplify notation
5: Set $e \leftarrow \text{VertexProbe} \left(v_{\pi(t)}, \partial(v_{\pi(t)}), \left(x_{\pi(t)}(e \parallel a) \cdot r_{\pi(t)}^{-1}(a) \right)_{e \in \mathcal{C}_a} \right)$.
6: if $e = (u, v_{\pi(t)})$ for some $u \in U$, and u is unmatched then
7: Add e to \mathcal{M} .
8: end if
9: end for
10: return \mathcal{M} .

```

---

Similarly, to Algorithm 1, one can show that Algorithm 7 attains a competitive ratio of  $1/2$  for random order arrivals. Interestingly, if the distributions  $(\mathcal{D}_i)_{i=1}^n$  are identical – that is, we work with known i.i.d. arrivals – then it is relatively easy to show that this algorithm becomes  $1 - 1/e$ -competitive.

► **Proposition 25.** *If Algorithm 7 is presented a known i.i.d. input, say the type graph  $H_{\text{typ}}$  together with the distribution  $\mathcal{D}$ , then  $\mathbb{E}[w(\mathcal{M})] \geq (1 - 1/e) \text{OPT}(H_{\text{typ}}, \mathcal{D})$ .*

► **Remark 26.** Proposition 25 is proven explicitly in an earlier 2020 arXiv version of this paper for the case of patience values.

Returning to the case of non-identical distributions, observe that in the execution of Algorithm 7 the probability that  $v_i$  commits to the edge  $(u, v_i)$  for  $u \in U$  is precisely

$$z_{u,i} := \sum_{b \in B} p_{u,b} \cdot \tilde{x}_{u,i}(b) = \sum_{b \in B} \sum_{\substack{e \in \mathcal{C}_b: \\ (u,b) \in e}} p_{u,b} \cdot q(e_{<(u,b)}) \cdot x_i(e \parallel b). \quad (20)$$

Moreover, the events  $(C(u, v_i))_{i=1}^n$  are independent, so this suggests applying the same contention resolutions schemes as in the known stochastic graph setting. We first focus on the adversarial arrival model, where we assume the vertices  $v_1, \dots, v_n$  are presented in some unknown order  $\pi : [n] \rightarrow [n]$ . We make use of the OCRS from before (Algorithm 3). For each  $t \in [n]$  and  $u \in U$ , define

$$q_{u,t} := \frac{1}{2 - \sum_{i=1}^{t-1} z_{u,\pi(i)}}, \quad (21)$$

where  $q_{u,1} := 1/2$ . Note that  $1/2 \leq q_{u,t} \leq 1$  as  $\sum_{j \in [n]} z_{u,j} \leq 1$  by constraint (16) of LP-config-id. We define Algorithm 8 by modifying Algorithm 7 using the OCRS to ensure that each  $i \in [n]$  is matched to  $u \in U$  with probability  $z_{u,i}/2$ . However, to achieve a competitive ratio of  $1/2$ , we require the stronger claim that for each type node  $a \in B$ , the probability  $(u, v_i)$  is added to the matching and  $v_i$  is of type  $a$  is lower bounded by  $p_{u,a} \tilde{x}_{u,i}(a)/2$ . Crucially, if we condition on  $u \in U$  being unmatched when  $v_i$  is processed,  $v_i$  having type  $a$ , and  $C(u, v_i)$ , then the probability the OCRS matches  $u$  to  $v_i$  does *not* depend on  $a$ . This implies the desired lower bound of  $p_{u,a} \tilde{x}_{u,i}(a)/2$ , and so Algorithm 8 attains a competitive ratio of  $1/2$  by (19) and Theorem 22 (we provide the details in the proof below).

■ **Algorithm 8** Known I.D. – AOM – Modified.

**Require:** a known i.d. input  $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$ .

**Ensure:** a matching  $\mathcal{M}$  of active edges of  $G \sim (H_{\text{typ}}, (\mathcal{D}_t)_{t=1}^n)$ .

---

```

1: $\mathcal{M} \leftarrow \emptyset$.
2: Compute an optimum solution of LP-config-id for $(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n)$, say
 $(x_i(e \parallel b))_{i \in [n], b \in B, e \in \mathcal{C}_b}$.
3: for $t = 1, \dots, n$ do
4: Let $a \in B$ be the type of the current arrival $v_{\pi(t)}$.
5: Based on the previous arrivals $v_{\pi(1)}, \dots, v_{\pi(t-1)}$ before $v_{\pi(t)}$, compute values $(q_{u,t})_{u \in U}$.
6: Set $e \leftarrow \text{VertexProbe} \left(v_{\pi(t)}, \partial(v_{\pi(t)}), \left(x_{\pi(t)}(e \parallel a) \cdot r_{\pi(t)}^{-1}(a) \right)_{e \in \mathcal{C}_a} \right)$.
7: if $e = (u, v_t)$ for some $u \in U$, and u is unmatched then
8: Add e to \mathcal{M} independently with probability $q_{u,t}$.
9: end if
10: end for
11: return \mathcal{M} .
```

---

**Proof of Theorem 1.** For notational simplicity, let us assume that  $\pi(t) = t$  for each  $t \in [n]$ , so that the online vertices arrive in order  $v_1, \dots, v_n$ . Now, the edge variables  $(\tilde{x}_{u,t}(b))_{u \in U, t \in [n], b \in B}$  satisfy  $\text{LPOPT}(H_{\text{typ}}, (\mathcal{D}_i)_{i=1}^n) = \sum_{u \in U, t \in [n], b \in B} p_{u,b} w_{u,b} \tilde{x}_{u,t}(b)$ . Thus, to complete the proof it suffices to show that

$$\mathbb{P}[(u, v_t) \in \mathcal{M} \text{ and } v_t = b] \geq \frac{\tilde{x}_{u,t}(b)}{2} \quad (22)$$

for each  $u \in U, t \in [n]$  and  $b \in B$ , where we hereby assume w.l.o.g. that  $\tilde{x}_{u,t}(b) > 0$ . In order to prove this, we first observe that by the same coupling argument used in the proof of Proposition 20,

$$\mathbb{P}[(u, v_t) \in \mathcal{M}] \geq \frac{z_{u,t}}{2} = \frac{1}{2} \sum_{b \in B} p_{u,b} \tilde{x}_{u,t}(b) \quad (23)$$

as a result of the  $1/2$ -selectability of Algorithm 3. Let us now define  $R_t$  as the unmatched vertices of  $U$  when  $v_t$  arrives. Observe then that

$$\mathbb{P}[(u, v_t) \in \mathcal{M} \mid v_t = b, C(u, v_t) \text{ and } u \in R_t] = q_{u,t}. \quad (24)$$

Now,  $\mathbb{P}[v_t = b, C(u, v_t) \text{ and } u \in R_t] = p_{u,b} \cdot \tilde{x}_{u,t}(b) \cdot \mathbb{P}[u \in R_t]$ , by Lemma 23 and the independence of the events  $\{v_t = b\} \cap \{C(u, v_t)\}$  and  $\{u \in R_t\}$ . Thus, by the law of total probability,

$$\sum_{b \in B} p_{u,b} \tilde{x}_{u,t}(b) q_{u,t} \cdot \mathbb{P}[u \in R_t] = \mathbb{P}[(u, v_t) \in \mathcal{M}] \geq \frac{z_{u,t}}{2} = \frac{1}{2} \sum_{b \in B} p_{u,b} \tilde{x}_{u,t}(b)$$

where the second inequality follows from (23). Thus,  $q_{u,t} \cdot \mathbb{P}[u \in R_t] \geq 1/2$ , and so combined with (24), (22) follows, thus completing the proof. ◀

Suppose now that each vertex  $v_t$  has an arrival time, say  $\tilde{Y}_t \in [0, 1]$ , drawn u.a.r. and independently for  $t \in [n]$ . The values  $(\tilde{Y}_t)_{t=1}^n$  indicate the increasing order in which the vertices  $v_1, \dots, v_n$  arrive.

---

**Algorithm 9** Known I.D. – ROM – Modified.

---

**Require:** a known i.d. input  $(H_{\text{typ}}, (\mathcal{D}_t)_{t=1}^n)$ .

**Ensure:** a matching  $\mathcal{M}$  of active edges of  $G \sim (H_{\text{typ}}, (\mathcal{D}_t)_{t=1}^n)$ .

- 1:  $\mathcal{M} \leftarrow \emptyset$ .
  - 2: Compute an optimum solution of LP-config-id for  $(H_{\text{typ}}, (\mathcal{D}_t)_{t=1}^n)$ , say  $(x_t(e \parallel b))_{t \in [n], b \in B, e \in \mathcal{C}_b}$ .
  - 3: **for**  $t \in [n]$  in increasing order of  $\tilde{Y}_t$  **do**
  - 4:   Set  $e \leftarrow \text{VertexProbe}(v_t, \partial(v_t), (x_t(e \parallel v_t)/r_t(v_t))_{e \in \mathcal{C}_{v_t}})$ .
  - 5:   **if**  $e = (u, v_t)$  for some  $u \in U$ , and  $u$  is unmatched **then**
  - 6:     Add  $e$  to  $\mathcal{M}$  independently with probability  $\exp(-\tilde{Y}_t \cdot z_{u,t})$ .
  - 7:   **end if**
  - 8: **end for**
  - 9: **return**  $\mathcal{M}$ .
- 

**Proof of Theorem 3.** The competitive ratio of  $1 - 1/e$  follows by the same coupling argument as in Proposition 21, together with the same observations used in the proof of Theorem 1, and so we omit the argument.  $\blacktriangleleft$

## B LP Relations

Suppose that we are given an arbitrary stochastic graph  $G = (U, V, E)$ . In this section, we first prove the equivalence between the relaxed stochastic matching problem and LP-config. We then state LP-std, the standard LP in the stochastic matching literature, as introduced by Bansal et al. [6], as well as LP-QC, the LP introduced by Gamlath et al. [27]. We then show that LP-QC and LP-config have the same optimum value when  $G$  is unconstrained.

► **Theorem 27.**  $\text{OPT}(G) = \text{LPOPT}_{\text{conf}}(G)$

**Proof.** Clearly, Theorem 8 accounts for one side of the inequality, so it suffices to show that  $\text{LPOPT}(G) \leq \text{OPT}_{\text{rel}}(G)$ . Suppose we are presented a feasible solution  $(x_v(e))_{v \in V, e \in \mathcal{C}_v}$  to LP-config. Consider then the following algorithm:

1.  $\mathcal{M} \leftarrow \emptyset$ .
2. For each  $v \in V$ , set  $e \leftarrow \text{VertexProbe}(v, \partial(v), (x_v(e))_{e \in \mathcal{C}_v})$ . If  $e \neq \emptyset$ , then add  $e$  to  $\mathcal{M}$ .
3. Return  $\mathcal{M}$ .

Using Lemma 10, it is clear that  $\mathbb{E}[w(\mathcal{M})] = \sum_{v \in V} \sum_{e \in \mathcal{C}_v} \text{val}(e) \cdot x_v(e)$ . Moreover, each vertex  $u \in U$  is matched by  $\mathcal{M}$  at most once in expectation, as a consequence of constraint (1) of LP-config, and so the algorithm satisfies the required properties of a relaxed probing algorithm. The proof is therefore complete.  $\blacktriangleleft$

Consider LP-std, which is defined only when  $G$  has patience values  $(\ell_v)_{v \in V}$ . Here each  $e \in E$  has a variable  $x_e$  corresponding to the probability that the offline adaptive benchmark probes  $e$ .

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} w_e \cdot p_e \cdot x_e \end{array} \quad (\text{LP-std})$$

$$\begin{array}{ll} \text{subject to} & \sum_{e \in \partial(u)} p_e \cdot x_e \leq 1 \quad \forall u \in U \end{array} \quad (25)$$

$$\sum_{e \in \partial(v)} p_e \cdot x_e \leq 1 \quad \forall v \in V \quad (26)$$

$$\sum_{e \in \partial(v)} x_e \leq \ell_v \quad \forall v \in V \quad (27)$$

$$0 \leq x_e \leq 1 \quad \forall e \in E. \quad (28)$$

Gamlath et al. modified LP-std in the unconstrained setting by adding in exponentially many extra constraints. Specifically, for each  $v \in V$  and  $S \subseteq \partial(v)$ , they ensure that

$$\sum_{e \in S} p_e \cdot x_e \leq 1 - \prod_{e \in S} (1 - p_e), \quad (29)$$

In the same variable interpretation as LP-std, the left-hand side of (29) corresponds to the probability the adaptive benchmark matches an edge of  $S \subseteq \partial(v)$ , and the right-hand side corresponds to the probability an edge of  $S$  is active<sup>2</sup>.

$$\text{maximize} \quad \sum_{e \in E} w_e \cdot p_e \cdot x_e \quad (\text{LP-QC})$$

$$\text{subject to} \quad \sum_{e \in S} p_e \cdot x_e \leq 1 - \prod_{e \in S} (1 - p_e) \quad \forall v \in V, S \subseteq \partial(v) \quad (30)$$

$$\sum_{e \in \partial(u)} p_e \cdot x_e \leq 1 \quad \forall u \in U \quad (31)$$

$$x_e \geq 0 \quad \forall e \in E. \quad (32)$$

Let us denote  $\text{LPOPT}_{\text{QC}}(G)$  as the optimum value of LP-QC.

► **Proposition 28.** *If  $G$  is unconstrained, then  $\text{LPOPT}_{\text{QC}}(G) = \text{LPOPT}(G)$ .*

In order to prove Proposition 28, we make use of a result of Gamlath et al. We mention that an almost identical result is also proven by Costello et al. [21] using different techniques.

► **Theorem 29** ([27]). *Suppose that  $G = (U, V, E)$  is an unconstrained stochastic graph, and  $(x_e)_{e \in E}$  is a solution to LP-QC. For each  $v \in V$ , there exists an online probing algorithm  $\mathcal{B}_v$  whose input is  $(v, \partial(v), (x_e)_{e \in \partial(v)})$ , and which satisfies  $\mathbb{P}[\mathcal{B}_v \text{ matches } v \text{ to } e] = p_e x_e$  for each  $e \in \partial(v)$ .*

**Proof of Proposition 28.** Observe that by Theorem 27, in order to prove the claim it suffices to show that  $\text{LPOPT}_{\text{QC}}(G) = \text{OPT}_{\text{rel}}(G)$ . Clearly,  $\text{OPT}_{\text{rel}}(G) \leq \text{LPOPT}_{\text{QC}}(G)$ , as can be seen by defining  $x_e$  as the probability that the relaxed benchmark probes the edge  $e \in E$ . Thus, we focus on showing that  $\text{LPOPT}_{\text{QC}}(G) \leq \text{OPT}_{\text{rel}}(G)$ . Suppose that  $(x_e)_{e \in E}$  is an optimum solution to  $\text{LPOPT}_{\text{QC}}(G)$ . We design the following algorithm, which we denote by  $\mathcal{B}$ :

1.  $\mathcal{M} \leftarrow \emptyset$ .
2. For each  $v \in V$ , execute  $\mathcal{B}_v$  on  $(v, \partial(v), (x_e)_{e \in \partial(v)})$ , where  $\mathcal{B}_v$  is the online probing algorithm of Theorem 29. If  $\mathcal{B}_v$  matches  $v$ , then let  $e'$  be this edge, and add  $e'$  to  $\mathcal{M}$ .
3. Return  $\mathcal{M}$ .

Using Theorem 29, it is clear that  $\mathbb{E}[w(\mathcal{M})] = \sum_{e \in E} w_e p_e x_e$ . Moreover, each vertex  $u \in U$  is matched by  $\mathcal{M}$  at most once in expectation, as a consequence of constraint (32). As a result,  $\mathcal{B}$  is a relaxed probing algorithm. Thus,  $\text{LPOPT}_{\text{QC}}(G) = \sum_{e \in E} w_e p_e x_e \leq \text{OPT}_{\text{rel}}(G)$ , and so the proof is complete. ◀

<sup>2</sup> The LP considered by Gamlath et al. in [27] also places the analogous constraints of (29) on the vertices of  $U$ . That being said, these additional constraints are not used anywhere in the work of Gamlath et al., so we omit them.

# The Biased Homogeneous $r$ -Lin Problem

Suprovat Ghoshal ✉

University of Michigan, Ann Arbor, MI, USA

---

## Abstract

The  $p$ -biased Homogeneous  $r$ -Lin problem ( $\text{Hom-}r\text{-Lin}_p$ ) is the following: given a *homogeneous* system of  $r$ -variable equations over  $\mathbb{F}_2$ , the goal is to find an assignment of relative weight  $p$  that satisfies the maximum number of equations. In a celebrated work, Håstad (JACM 2001) showed that the unconstrained variant of this i.e., Max-3-Lin, is hard to approximate beyond a factor of  $1/2$ . This is also tight due to the naive random guessing algorithm which sets every variable uniformly from  $\{0, 1\}$ . Subsequently, Holmerin and Khot (STOC 2004) showed that the same holds for the *balanced*  $\text{Hom-}r\text{-Lin}$  problem as well. In this work, we explore the approximability of the  $\text{Hom-}r\text{-Lin}_p$  problem beyond the balanced setting (i.e.,  $p \neq 1/2$ ), and investigate whether the ( $p$ -biased) random guessing algorithm is optimal for every  $p$ . Our results include the following:

- The  $\text{Hom-}r\text{-Lin}_p$  problem has no efficient  $\frac{1}{2} + \frac{1}{2}(1 - 2p)^{r-2} + \varepsilon$ -approximation algorithm for every  $p$  if  $r$  is even, and for  $p \in (0, 1/2]$  if  $r$  is odd, unless  $\text{NP} \subset \cup_{\varepsilon > 0} \text{DTIME}(2^{n^\varepsilon})$ .
- For any  $r$  and any  $p$ , there exists an efficient  $\frac{1}{2}(1 - e^{-2})$ -approximation algorithm for  $\text{Hom-}r\text{-Lin}_p$ . We show that this is also tight for odd values of  $r$  (up to  $o_r(1)$ -additive factors) assuming the Unique Games Conjecture.

Our results imply that when  $r$  is even, then for large values of  $r$ , random guessing is near optimal for every  $p$ . On the other hand, when  $r$  is odd, our results illustrate an interesting contrast between the regimes  $p \in (0, 1/2)$  (where random guessing is near optimal) and  $p \rightarrow 1$  (where random guessing is far from optimal). A key technical contribution of our work is a generalization of Håstad's 3-query dictatorship test to the  $p$ -biased setting.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Problems, reductions and completeness

**Keywords and phrases** Biased Approximation Resistance, Constraint Satisfaction Problems

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.47

**Category** APPROX

**Acknowledgements** Most of this work was carried out when the author was visiting KTH, Stockholm. The author thanks Johan Håstad for inviting him to visit KTH, for the many insightful discussions and ideas that have played a key role in this work, and for his comments on a previous draft of the manuscript. The author thanks Per Austrin for several useful discussions on related topics. The author also thanks the anonymous reviewers for several helpful suggestions on the manuscript, and for fixing an issue in a previous version of the work.

## 1 Introduction

The problem of finding solutions to systems of linear equations is one of fundamental importance. While in theory, the exact running time complexity of even efficiently solvable instances has profound implications in the theory of algorithms [26], the question of approximability of infeasible systems is also fundamental and has been studied widely [19, 12, 24, 14, 8]. A particularly useful instantiation of this is the  $\text{Max-}r\text{-Lin}$  problem<sup>1</sup> where given a (possibly infeasible) system of  $r$ -variables equations over  $\mathbb{F}_2$ , the objective is to find an assignment

---

<sup>1</sup> In the literature, the  $\text{Max-}r\text{-Lin}$  problem is typically referred to as  $\text{Max-}r\text{-Lin}_q$ , where the indexing by  $q$  indicates that the equations are over  $\mathbb{F}_q$ . We drop the indexing by  $q$  since the current work deals only with the setting  $q = 2$ .



© Suprovat Ghoshal;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 47; pp. 47:1–47:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



to the variables that satisfies the maximum fraction of equations. The Max- $r$ -Lin problem manifests as various computational problems in error correcting codes, combinatorial optimization, probabilistically checkable proofs among many others. In particular, its study played a seminal role in the development Probabilistically Checkable Proofs based reductions [7, 19, 24], and techniques introduced for studying its hardness have now become staple tools in hardness of approximation.

A standout result among these is the celebrated work of Håstad [19] who showed that for  $r \geq 3$ , the Max- $r$ -Lin problem is NP-hard to approximate beyond a factor of  $1/2$ . This is clearly tight since the naive algorithm which outputs a uniformly random assignment also satisfies at least  $1/2$ -fraction of constraints<sup>2</sup>. This property of the “random guessing algorithm being optimal” also happens to hold for a much broader class of combinatorial optimization problems, and is formally studied under the notion of “approximation resistance”. The ubiquity of this notion has lead to several works which systematically study such problems [5, 4, 10], including landmark results such as which give complete conditional and unconditional characterizations of approximation resistant predicates [3, 25].

A key problem studied in this context is the *Balanced Homogeneous Max-3-Lin* problem, where given a *homogeneous* system of linear equations, the goal is to find a *balanced* assignment that satisfies the maximum fraction of constraints. Clearly, naive random guessing is still a candidate algorithm for this setting as well, since it produces balanced<sup>3</sup> assignments that satisfy at least half of the constraints. Naturally, this leads one to ask if random guessing is still optimal in this setting as well? This was answered in the affirmative by Holmerin and Khot [22] who ruled out efficiently approximability beyond  $1/2$  assuming SAT does not admit sub-exponential time algorithms. Subsequently, Håstad and Manokaran [21] strengthened the above hardness result to rule out quasi-polynomial time algorithms which give better than  $1/2$  approximation assuming  $\text{NP} \not\subseteq \text{DTIME}(\exp(\log n)^{O(1)})$ .

In this work, we study a natural generalization of the above and investigate the approximability of the homogeneous Max-3-Lin problem beyond the balanced setting. Formally, we study the  $p$ -biased version of the above problem, which we refer to as the **Hom- $r$ -Lin $_p$**  problem. We define it formally below:

► **Definition 1 (Hom- $r$ -Lin $_p$ ).** Given  $p \in (0, 1)$ , an instance  $\psi([n], E)$  of the  $p$ -biased Hom- $r$ -Lin problem is given by a set of homogeneous equations over  $\mathbb{F}_2$  defined by a set of  $r$ -arity hyperedges  $E := \{e_1, \dots, e_m\}$  over variables  $\{x_1, \dots, x_n\}$ , where the  $i^{\text{th}}$  hyperedge  $e_i$  implies the constraint  $\bigoplus_{j \in e_i} x_j = 0$ . Here, the objective is to find a labeling of relative weight  $p$  which satisfies the maximum fraction of hyperedges (constraints).

Clearly, for  $p = 1/2$ , the above recovers the balanced setting, for which the aforementioned works show that the uniformly random guessing algorithm is optimal. On the other hand, for any  $p \in (0, 1)$ , one can naturally consider the following extension of random guessing: set each variable to 1 independently with probability  $p$  – we refer to this as  $p$ -biased random guessing. Clearly, with high probability,  $p$ -biased random guessing will return an assignment

<sup>2</sup> In fact, a simpler deterministic  $1/2$ -approximation algorithm is known for Max- $r$ -Lin: given any assignment to the set of the variables, that or its negation will always satisfy at least  $1/2$  of the constraints – hence, outputting the best of any assignment and its negation gives a trivial  $1/2$ -approximation. However, since negating the assignment can also change its relative weight, this approach doesn’t yield an algorithm for the weight constrained setting considered in this paper.

<sup>3</sup> Strictly speaking, it produces almost balanced assignments, which can be converted to exactly balanced assignments by changing  $o_n(1)$ -variables. This only affects the approximation factor in lower order  $o(1)$  terms.

with relative weight  $\approx p$ , i.e., it is again a feasible candidate algorithm. And keeping with the above trend, one may ask if the  $p$ -biased random guessing algorithm is still optimal for the  $\text{Hom-}r\text{-Lin}_p$  problem, for every  $p$ . In other words, we ask the following:

Is the  $\text{Hom-}r\text{-Lin}_p$  problem approximation resistant for every  $p \in (0, 1)$ ?

The above is the main motivating question which we seek to address in the current work. At a finer level, our goal is to understand the approximability of the  $\text{Hom-}r\text{-Lin}_p$  problem as a function of the parameter  $p$  and the arity  $r$ . This formulation of the problem brings in several additional dimensions to the existing literature on approximation resistance which typically deals with uniformly random guessing, as opposed to the more general  $p$ -biased guessing studied in the current work.

## 1.1 Our Results

In this work, we study the bias dependent approximability of  $\text{Hom-}r\text{-Lin}_p$ , and make substantial progress towards understanding the above question. In the interest of keeping the presentation concise, we will first state our results for the setting when  $r$  is odd, since this setting exhibits a more interesting dependence on the parameter  $p$ . We will then point out how the results change when  $r$  is even.

**The  $p \leq 1/2$  setting.** Our first result is the following theorem which shows that  $\text{Hom-}r\text{-Lin}_p$  predicate is close to being approximation resistant for large values of  $r$ .

► **Theorem 2.** *Fix  $p, \eta \in (0, 1/2)$ , and  $r \geq 3$ . Then assuming  $\text{NP} \not\subseteq \cup_{\varepsilon>0} \text{DTIME}(2^{n^\varepsilon})$  the following holds. Given an instance  $\psi$  of  $\text{Hom-}r\text{-Lin}$ , there is no polynomial time algorithm that can distinguish between the following cases:*

- **YES Case.** *There exists an assignment of relative hamming weight  $p$ , which satisfies at least  $1 - \eta$  fraction of constraints.*
- **NO Case.** *No assignment of relative hamming weight  $p$  satisfies more than  $\frac{1}{2} + \frac{1}{2}(1 - 2p)^{r-2} + \eta$  fraction of constraints.*

The above theorem implies that there are no efficient algorithms which give a better than  $\frac{1}{2}(1 + (1 - 2p)^{r-2})$ -approximation. On the other hand, it is easy to verify that the  $p$ -biased random guessing is a  $\frac{1}{2}(1 + (1 - 2p)^r)$ -approximation algorithm for  $\text{Hom-}r\text{-Lin}_p$ . Hence, the above theorem implies that for large  $r$ , biased random guessing is almost optimal.

**The  $p > 1/2$ -setting.** Our second result shows that the almost approximation resistant behavior of the  $\text{Hom-}r\text{-Lin}_p$  predicate breaks down in the  $p > 1/2$  setting. This is implied by the following theorem which gives an efficient randomized  $p$ -independent approximation algorithm for  $\text{Hom-}r\text{-Lin}_p$ .

► **Theorem 3.** *For every  $r \geq 3$  and every  $p \in (0, 1)$ , there exists an efficient randomized  $\beta_r/2$ -approximation algorithm for  $\text{Hom-}r\text{-Lin}_p$ . Here  $\beta_r := 1 - (1 - 1/r)^{2r}$  is a decreasing function of  $r$  satisfying  $\lim_{r \rightarrow \infty} \beta_r = (1 - e^{-2})$ .*

The algorithm for the above theorem is based on a linear programming + rounding approach inspired by algorithms for hitting set and is markedly different from random guessing. The above theorem implies that even for  $r = 3$ , the random guessing algorithm is *strictly sub-optimal* for all  $p > 1/2(1 + e^{-2/3})$ . We also show that the above approximation guarantee is tight (up to  $o_r(1)$ -factors) assuming the Unique Games Conjecture [23].

► **Theorem 4.** *Assuming the Unique Games Conjecture, the following holds for every odd  $r \geq 3$  and  $\eta \in (0, 1)$ . Let  $p := 1 - 1/r$ . Given a Hom- $r$ -Lin instance  $\psi$  it is NP-hard to distinguish between the following cases:*

- **YES Case.** *There exists an assignment of relative weight  $p$  which satisfies at least  $(1 - \eta)$ -fraction of constraints.*
- **NO Case.** *No assignment of relative weight  $p$  satisfies more than  $\left(\frac{\beta_r}{2} + O(1/r)\right)$ -fraction of constraints.*

**The even  $r$ -setting** . It is easy to see that when  $r$  is even, the  $p \geq 1/2$  and  $p \leq 1/2$  regimes are symmetric. To see this, given a system of equations  $\psi([n], E)$ , and an assignment  $(x_1, \dots, x_n)$  of relative weight  $p$ , consider the negated assignment  $x'_i = 1 \oplus x_i$ . Since  $r$  is even and the constraints are homogeneous, the negated assignment will satisfy a constraint if and only if the original assignment satisfied the constraint. Furthermore, the negated assignment will have a relative weight of  $1 - p$ . This observation implies that the Hom- $r$ -Lin $_p$  problem behaves identically under the bias constraints  $p$  and  $1 - p$  for every  $p$ . This observation along with Theorem 2 results in the following corollary.

► **Corollary 5.** *For every  $p, \varepsilon \in (0, 1)$  and even  $r \geq 4$ , there is no polynomial time  $\frac{1}{2}(1 + (1 - 2p)^{r-2}) + \varepsilon$ -approximation algorithm for Hom- $r$ -Lin $_p$  unless  $\text{NP} \subset \text{DTIME}(2^{n^\varepsilon})$  for any  $\varepsilon > 0$ .*

**Threshold Phenomena.** The above results show that when  $p < 1/2$ , then the random guessing algorithm is almost optimal, whereas this behavior breaks down for the  $p > 1/2$  regime when  $r$  is odd. In particular, in the setting of large  $p$ 's the Hom- $r$ -Lin predicate exhibits a *hitting set* like behavior – our algorithm and hardness results (Theorem 4 and 3) are based on this connection. In fact, we believe that when  $p \leq 1/2$ , the  $p$ -biased random guessing algorithm is indeed optimal, and the current gap in the hardness result is due to technical bottlenecks<sup>4</sup> that arise more generally in the context of hardness reductions involving problems with global constraints.

Furthermore, our results hint at the possibility of the existence of a threshold  $p_r$  beyond which the approximation resistance of Hom- $r$ -Lin breaks. In particular, the hard distribution for Theorem 4 seems to indicate that this threshold is  $1 - 1/r$ . Lastly, we point out that the even and odd  $r$  settings contrast nicely against each other as while Hom- $r$ -Lin $_p$  can be approximation resistant only for a certain range of  $p$  when  $r$  is odd, it is possibly approximation resistant for every  $p$  when  $r$  is even.

## 1.2 Related Works

**The Max- $r$ -Lin Problem.** The Max- $r$ -Lin problem has been studied extensively in the literature. In particular, when  $r = 2$ , it expresses the affine UNIQUEGAMES problem which is central to Khot's Unique Games Conjecture (UGC) [23], and has been extensively studied by several works [12, 11, 24]. In particular, the algorithmic results from [11] show that the

<sup>4</sup> In particular, the gap of  $(1 - 2p)^2$  in the second term is primarily due to the following reason. As is standard in Label Cover based reductions, out of the  $r$ -queries made by our dictatorship test, 2 of its queries are made to the large side table for the consistency check. However, the outer verifier (i.e., Mixing Label Cover) can only guarantee mixing w.r.t. vertices on the smaller side (due to which we are able to recover the  $(1 - 2p)^{r-2}$ -factor, and doesn't guarantee mixing on the larger side (due to which we lose out by a factor  $(1 - 2p)^2$  in the  $p$ -dependent term). The question of constructing hard outer verifiers with mixing guarantees with respect all vertices is a fundamental technical challenge in itself in this area.

$r = 2$  setting is not approximation resistant. For  $r \geq 3$ , Håstad [19] showed that Max- $r$ -Lin is hard to approximate beyond a factor of  $1/2 + \varepsilon$ . In fact, Håstad's result actually shows that the Max- $r$ -Lin problem is approximation resistant over any finite abelian group, which was later strengthened to the setting of infeasible instances over non-abelian groups [14]. More recently, Bhangale and Khot [8] give tight hardness results for satisfiable Hom- $r$ -Lin instances over non-abelian finite groups. Specifically, given a non-abelian group  $G$ , they showed that it is hard to approximate the satisfiable Hom- $r$ -Lin problem beyond a factor of  $1/|[G, G]| + \varepsilon$ , where  $[G, G]$  is the commutator sub-group of  $G$  – this is matched by a folklore algorithm for the same, we refer interested readers to [8] for more details on this.

**Approximation Resistance.** Starting with the work of Håstad [19], the question of understanding the conditions under which random guessing is optimal has generated great interest, and has been studied by several works. In particular, the work of Håstad [20] showed that 2-arity CSPs can never be approximation resistant. For when the arity  $r$  is 3, it is known that a CSP on a predicate can be approximation resistant if and only if the predicate is implied by 3-XOR [19, 32]. The work of Hast [18], gives an almost complete characterization for 4-arity CSPs. On the other hand, stronger results are known assuming stronger hypotheses. For e.g., assuming UGC, Håstad and Austrin [3] showed that a uniformly random predicate is approximation resistant with high probability. Austrin and Khot [4] gave a complete characterization of approximation resistance for  $k$ -partite CSPs, which was later strengthened to the setting of all CSPs by Khot, Tulsiani and Worah [25].

**Globally Constrained CSPs.** The Hom- $r$ -Lin $_p$  problem also falls within the framework of Max-CSPs with global cardinality constraints i.e., CSPs where the objective is to find a labeling that satisfies the maximum number of edge constraints while “strictly” respecting global constraints. Such CSPs express extensively studied problems such as Max-Bisection [28, 2], Densest- $k$ -Subgraph [15, 9], Small Set Expansion [29, 30]. There have been several works which also systematically study such CSPs under a more general framework. For e.g., the works of Guruswami and Sinop [17] and Bansal et al. [1] propose general purpose algorithmic frameworks for solving globally constrained CSPs. A closely related work is that of Ghoshal and Lee [16], who study the bias parameter dependent approximation curve for globally constrained Boolean CSPs, and give upper and lower bounds which are matching up to constant factors for constant arity, assuming the Small Set Expansion Hypothesis. In particular, their results imply that there exists a  $\Omega(2^{-r})$ -approximation algorithm, which again is  $p$ -independent but deteriorates with increasing  $r$ .

## 2 Warm-up : The $p \leq 1/2$ vs. $p > 1/2$ settings

In this section, we first provide some intuition on why the behavior of the approximation curve of the Hom- $r$ -Lin predicate in the  $p \geq 1/2$  regime is different from that of  $p < 1/2$  when  $r$  is odd. Consider the random guessing algorithm for the Hom- $r$ -Lin problem which sets every variable to 1 with probability  $p$ . Then, elementary computation shows that this assignment satisfies  $\frac{1}{2} + \frac{1}{2}(1 - 2p)^r$ -fraction of constraints in expectation and w.h.p. the relative weight of the assignment returned by the algorithm is  $\approx p$ . Clearly, the approximation guarantee of this algorithm improves as  $p$  decreases and is trivially optimal in the limit  $p = 0$ , and hence it might not be a stretch to posit that the random guessing algorithm is indeed optimal in this regime.

On the other hand, as  $p$  increases, the approximation guarantee of the random guessing algorithm worsens, eventually reaching 0 as  $p \rightarrow 1$ . However, while random guessing might be almost optimal when  $p \in (0, 1/2)$ , there is an inherent reason as to why it should be sub-optimal in the almost all-ones regime. The key insight here is that the random guessing can be wasteful in term of choosing the zeros when the budget of zeros is small (i.e., when  $p \rightarrow 1$ ). For instance, consider a  $\text{Hom-}r\text{-Lin}_p$  instance  $\psi([n], E)$  whose optimal assignment satisfies nearly all constraints. Then one can show that there exists a small set of vertices in  $V$  which intersects (hits) nearly all hyperedges in  $\psi$  – this is witnessed by the zero set of the optimal assignment. In contrast, the solutions output by the random guessing algorithm will have the set of zeros spread uniformly throughout the constraint hypergraph, and hence such assignments are likely to miss out on the potential exploits guaranteed by the combinatorial structure of the instance, thus ending up satisfying far fewer than the optimal fraction of constraints. On the other hand, the existence of such a hitting set opens up other possible approach which can use this. For instance, the surrogate problem of finding a small size set which hits the maximum number of hyperedges itself is known to admit efficient constant factor approximation algorithms using simple greedy/linear programming based approaches. The above observations indicate that one might be able to strictly better than random guessing by exploiting the combinatorial structure of the instance.

### 3 Hardness for $p \leq 1/2$

Our result for  $p \leq 1/2$  (Theorem 2) is based on a careful generalization of the standard 3-query dictatorship test of Håstad to the setting of biased long codes. In order to highlight the challenges towards establishing the hardness result, it will be useful to go over the reduction for the hardness of balanced setting (i.e.,  $p = 1/2$ ) from [22]. The reduction in [22] for  $\text{Hom-}r\text{-Lin}$  consisted of two key components:

- (i) The 3-query dictatorship test of Håstad for 3-Lin.
- (ii) A variant of LABEL COVER<sup>5</sup> with one-sided *mixing* properties.

While the 3-query test from (i) was established much before the work of Holmerin and Khot [22], the key contribution of [22] was a variant of LABEL COVER with the property that the larger side of the LABEL COVER instance  $\mathcal{L}$  has good expansion – this is needed to ensure that globally balanced assignments also translate to locally balanced assignments in the reduction. As we shall see, while we can use the outer verifier from (ii) as is in our reduction, most of the work will go into modifying and analyzing the inner verifier (i.e., the dictatorship test). Now we shall first briefly describe how (i) and (ii) can be put together to prove the hardness for the balanced setting, and then we will discuss the challenges and the techniques used for going beyond the balanced case.

**The 3-query test.** In order to understand the challenges towards designing a  $r$ -query test that works in our setting, it is instructive to recall the well known 3-query test of [19]. The design of the 3-query test is based on the principle that linear functions with small number of influential coordinates must be close to being dictators, which readily yields test in Figure 1.

<sup>5</sup> A Label Cover instance  $\mathcal{L}(U, V, E, [s], [l], \{\pi_e\}_{e \in E})$  is a 2-CSP on the bipartite constraint graph with left and right vertex sets  $U$  and  $V$  respectively, with the edge constraint set  $E$ . Every edge  $(u, v) \in E$  is identified with a projection constraint  $\pi_{u,v} : [l] \rightarrow [s]$ . The objective is to find a global labeling of  $U$  and  $V$  using  $[s]$  and  $[l]$  respectively, that satisfies the maximum fraction of edge constraints in  $E$ .

**Input.** Long code table  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$  satisfying  $\mathbb{E}_{x \sim \{0, 1\}^k} [f(x)] = 0$ .

**Test.**

1. Sample  $x, y \sim \{0, 1\}_{1/2}^k$  and  $\rho \sim \{0, 1\}_\eta^k$ . Set  $z := x \oplus y \oplus \rho$ .
2. Accept if and only if

$$f(x) \cdot f(y) \cdot f(z) = 1.$$

■ **Figure 1** 3-query test.

The analysis of this test proceeds through the following well-traversed path. Firstly, it is easy to see that the dictator assignment  $f = \chi_i$  is balanced and passes the test with probability  $(1 - \eta)$ . On the other hand, the soundness direction proceeds as follows: given a balanced assignment  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$ , we can arithmetize the acceptance probability of the test in term of  $f$  and express it as:

$$\Pr[\text{Test Accepts}] = \frac{1}{2} + \frac{1}{2} \mathbb{E}_{x, y, z} [f(x)f(y)f(z)].$$

Furthermore, by standard Fourier analytic arguments, we can manipulate the RHS of the above and further write the RHS in terms of the Fourier coefficients of  $f$  as

$$\Pr[\text{Test Accepts}] = \frac{1}{2} + \frac{1}{2} \sum_{\beta \neq \mathbf{0}_k} \widehat{f}(\beta)^3 (1 - \eta)^{|\beta|}, \quad (1)$$

where note that the summation term does not feature the term  $\beta = \mathbf{0}_k$  since the Fourier coefficient corresponding to the all zeros term vanishes due to the *balancedness* of the long code table. Hence, if the test passes with probability strictly bounded away from  $1/2$ , then the summation term is strictly positive, which allows one to show that there exists  $\beta \in \{0, 1\}^k$  such that  $\widehat{f}(\beta) \geq \Omega(1)$  and  $|\beta| \leq O(1/\eta)$  i.e.,  $f$  has non-trivial correlation with a low degree term. In particular, note that since the summation omits the  $\beta = \mathbf{0}_k$  term, the low degree term is guaranteed to be non-trivial – this property is used crucially in the composition step which we describe next.

**Composing with Mixing Label Cover.** Given the above dictatorship test, the next step is to compose it with the outer verifier (i.e., Label Cover). This is a standard step in dictatorship test based reductions, and roughly goes as follows. Informally, given a Label Cover instance  $\mathcal{L}(U, V, E, \Sigma, \{\pi_e\}_{e \in E})$ , the reduction introduces a long code table  $f_w : \{0, 1\}^{|\Sigma_w|} \rightarrow \{\pm 1\}$  for every vertex  $w \in U \cup V$  – the entries of the long code tables correspond to the variable set of the reduction. The constraint set of the reduction corresponds to the distribution over checks generated by the following process:

- Sample a random edge  $(u, v) \sim \mathcal{L}$ .
- Run the 3-query test by querying the positions from the long code tables  $f_u(x)$   $f_v(y)$  and  $f_v(\pi_e(x) \oplus y \oplus z)$ .

The above seeks to simultaneously test the following (i) the tables  $f_u$  and  $f_v$  are correlated with non-trivial low degree terms, and (ii) the sets corresponding to the low degree terms have non-trivial intersection under the projection map  $\pi_e$ . These checks are indeed useful due to the following principle: if (i) and (ii) hold simultaneously for a non-negligible fraction of fraction of edges incident on a vertex  $v \in V$ , then the assignment to the long code tables can be used to decode a labeling that satisfies a non-trivial fraction of constraints incident



on  $v$  in  $\mathcal{L}$ . However, lifting the soundness guarantee of the test to ensure (i) and (ii) from above requires the empty set Fourier coefficient term to vanish in expectation for a random choice of  $u \sim N_{\mathcal{L}}(v)$  (recall that this is critical in ensuring that the low degree term in (i) is non-trivial). This is precisely where the mixing property is useful – it ensures that if the global set of  $U$ -tables are balanced, then for most choices of  $v \in V$ , the long code tables in the neighborhood  $N_{\mathcal{L}}(v)$  of  $v$  are also balanced (in expectation). In terms of the soundness analysis of the full reduction, this translates to the guarantee that for most choices of  $v$ , empty set Fourier coefficient term has negligible contribution. However, the mixing property of the Label Cover instance comes at the cost of super-polynomial sized constructions, due to which the inapproximability only holds under stronger assumptions such as  $\text{NP} \not\subseteq \cup_{\varepsilon>0} \text{DTIME}(2^{n^\varepsilon})$ .

### 3.1 The Biased $r$ -query test

Towards generalizing the above to the setting of any  $p \leq 1/2$ , a clear first obstacle is to design a dictatorship test that has the right completeness and soundness guarantees as functions of  $p$  and  $r$ . It turns out that this is the key issue that we need to address, as once we are equipped with the right test, the composition step and its analysis follows almost as is using the techniques from the balanced case. Formally, our goal is to design a  $r$ -query test with the following property:

- **Completeness:** If  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$  is a dictator, then it passes the test with probability  $1 - \eta$ , and induces an assignment of relative weight  $p$  i.e.,  $\mathbb{E}_{x \sim \{0, 1\}_p^k} [f(x)] = 1 - 2p$ .
- **Soundness:** If  $f$  is an assignment of relative weight  $p$  and passes the probability at least  $\frac{1}{2}(1 + (1 - 2p)^{r-2}) + \Omega(1)$ , then  $f$  is correlated with a non-trivial Fourier character of low-degree.

It is easy to see that the 3-query test from Figure 1 does not imply the above conditions, and in particular, fails the completeness guarantee. This is due to the observation that since the marginal distribution of the points queried by the test is uniform over  $\{0, 1\}^k$ , any dictator assignment will be balanced under the distribution. This leads us to consider analogous  $p$ -“biased” dictatorship tests over the  $p$ -biased hypercube<sup>6</sup>.

Towards designing such a “biased” dictatorship test, a natural first approach (while ignoring the noise component  $\rho$ ) would be to consider the distribution over triples over choices of  $(x, y, z)$  such that  $x \oplus y \oplus z = \mathbf{0}_k$  such that  $x, y$  and  $z$  are marginally distributed as  $\{0, 1\}_p^k$  – such a test was explored (for slices of the hypercube) in the context of *direct sum testing* in [13] for the regime where the soundness of the test approaches 1. However, to the best of our knowledge, the techniques used in [13] seem to rely on the soundness parameter being close to 1, whereas our application would require us to establish guarantees in settings where the soundness parameter is bounded away from 1. Furthermore, even in the regime where the soundness approaches 1, it is not easy to see how the techniques of [13] generalize to the setting of higher values of  $r$ . Finally, the techniques used in the analysis of the above test (Figure 1) do not generalize well to this setting, since establishing (1) heavily relies on the fact that the Fourier characters for the unbiased distribution are linear operators over  $\mathbb{F}_2^k$  i.e.,  $\chi_\alpha(x \oplus y) = \chi_\alpha(x) \oplus \chi_\alpha(y)$ , a property that does not hold for the general  $p$ -biased Fourier characters (i.e., when  $p \neq 1/2$ ).

<sup>6</sup> The  $p$ -biased Boolean  $k$ -hypercube is the  $k$ -hypercube  $\{0, 1\}^k$  equipped with the following measure:  $\mu(x) := p^{|x|}(1 - p)^{k - |x|}$ .



The above observations instead motivate us to consider a test with an asymmetric test distribution, where the first  $(r - 2)$ -query positions are independent  $p$ -biased strings and the remaining two strings are uniformly distributed but correlated. Formally, we consider the distribution in the following figure.

**Input.** Long code table  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$  satisfying  $\mathbb{E}_{x \sim \{0, 1\}_p^k} [f(x)] = 1 - 2p$ .

**Test.**

1. Sample  $x_1, \dots, x_{r-2} \sim \{0, 1\}_p^k$ .
2. Sample  $y \sim \{0, 1\}_{1/2}^k$  and  $\rho \sim \{0, 1\}_\eta^k$ . Set  $z := (\oplus_{i \in [r-2]} x_i) \oplus y \oplus \rho$ .
3. Accept if and only if

$$\left( \prod_{i \in [r-2]} f(x_i) \right) \cdot f(y) \cdot f(z) = 1.$$

■ **Figure 2** The  $p$ -biased  $r$ -query test.

The above distribution allows us to have *the best of both worlds*: the  $(r - 2)$  independent  $p$ -biased strings ensure that the completeness and soundness parameters have the desired dependence on the parameters  $p$  and  $r$ , where as the uniformity of  $y$  and  $z$  ensures that the analysis can exploit the linearity of Fourier characters in the necessary steps and recover the quadratic term required to show correlation with a low degree term. We conclude our discussion by giving a brief sketch of the completeness and soundness analysis of the above test. As in the 3-query test, it is straightforward to establish that a dictator function will again pass the test with probability  $1 - \eta$  and induce an assignment of weight  $1 - 2p$ . On the other hand, analyzing the soundness direction is requires additional care. In particular, note that since the marginal distributions of the first  $(r - 2)$ -distribution are  $p$ -biased, where as the remaining two strings are uniformly distributed, we have to expand the arithmetization in a hybrid bases consisting of  $p$ -biased Fourier expansion for the first  $(r - 2)$ -strings and the unbiased Fourier expansion for the others. This approach introduces additional complications since the  $p$ -biased Fourier characters are non-linear operators over  $\mathbb{F}_2$  and hence the arguments from the  $p = 1/2$  setting don't apply as is. Nevertheless, using the properties of the Fourier characters we can still establish the following analogue of (1).

$$\Pr [\text{Test Accepts}] \leq \frac{1}{2} + \frac{1}{2}(1 - 2p)^{r-2} + 2^r \cdot \sum_{\beta \neq \mathbf{0}_k} \sum_{\alpha \subseteq \beta} \widehat{f}_p(\alpha) \widehat{f}(\beta)^2 (1 - \eta)^{|\beta|}$$

where  $\{\widehat{f}_p(\alpha)\}_\alpha$  and  $\{\widehat{f}(\beta)\}_\beta$  are the Fourier coefficients of  $f$  with respect to the  $p$ -biased and unbiased Fourier expansion respectively. Establishing the above involves the most of the work in the soundness analysis and requires a careful application of properties of the  $p$ -biased Fourier characters. Note that the above immediately implies the soundness guarantee of the test: if the test passes with probability bounded away from  $\frac{1}{2}(1 + (1 - 2p)^{r-2})$ , the summation term of the above equation is strictly positive, which with some additional work can be used to show that  $f$  has a low degree term of significant magnitude, thus implying non-trivial correlation with a non-trivial low degree Fourier character.

#### 4 $\frac{1}{2}(1 - e^{-2})$ -approximation for all $p$

As observed earlier in Section 2, the key to doing better than the random guessing algorithm lies in carefully identifying the zero set of the assignment – in particular, we use the fact that the zero set of the optimal assignment must hit a large fraction of hyperedges. Formally we observe the following. Let  $\psi([n], E)$  be a **Hom- $r$ -Lin $_p$**  instance whose optimal value is  $\alpha$ . Then we claim that there exists a set of size  $(1 - p)n$  which hits at least  $\alpha$ -fraction of hyperedges (constraints) in  $\psi$ . This is due to the observation that since  $r$  is odd, the all-ones string is not a satisfying assignment to the **Hom- $r$ -Lin $_p$**  predicate, and hence if an assignment satisfies a constraint  $e \in E$ , then at least one of the variables in  $e$  must be set to 0. Furthermore, the problem of hitting the largest number of hyperedges with a cardinality constraint is a *coverage type* problem, which readily admits a linear programming based  $(1 - 1/e)$ -approximation algorithm.

The above observations immediately suggests the following approach which leads to a constant factor but sub-optimal approximation guarantee. (i) Find the set  $S$  of size  $(1 - p)n$  which is a  $(1 - 1/e)$ -approximation to the coverage problem. (ii) Set all variables in  $V \setminus S$  to 1 and all variables variables in  $S$  uniformly. This results in a  $(1 - (1 - p)/2)$ -weight assignment which satisfies  $\frac{1}{2}(1 - 1/e) \cdot \text{Opt}(\psi)$ -fraction of constraints<sup>7</sup>. Our actual algorithm uses a slight modification of the above approach, and is based on the observation that simply using a single hitting set of size  $(1 - p)n$  to round off the final assignment is wasteful, since that only sets  $(1 - p)/2$ -fraction of variables to 0, where as the final solution allows for  $(1 - p)$ -fraction of variables to be set to 0. Instead, our algorithm actually first independently rounds off two hitting sets of size  $(1 - p)n$  (from the same fractional solution), and then uses the union of these two sets to construct an assignment of weight  $p$ . We outline the steps of our algorithm below.

- Solve the following linear programming relaxation for finding a set which hits maximum fraction of hyperedges in  $\psi$ .

$$\begin{aligned}
 & \text{Maximize } \sum_{e \in E} x_e \\
 & \text{Subject to } \sum_{i \in e} z_i \geq x_e \quad \forall e \in E \\
 & \quad \sum_{i \in V} z_i \leq 2(1 - p)n \\
 & \quad 0 \leq x_e, z_i \leq 1 \quad \forall e \in E, i \in V.
 \end{aligned}$$

- Independently round off two sets  $S_1, S_2$  of size  $\approx (1 - p)n$  by independent rounding using  $\{z_i\}_{i \in V}$ .
- Set all variables in  $[n] \setminus (S_1 \cup S_2)$  to 1 and every variable in  $S_1 \cup S_2$  is set to  $\{0, 1\}$  uniformly.

We give a brief sketch of the correctness of the above algorithm. Firstly, using a slight modification of the standard analysis of LP rounding for coverage type problems (for e.g., see Section 16.3 [31]), we can show that  $S_1 \cup S_2$  will hit at least  $(1 - e^{-2}) \cdot \text{Opt}(\psi)$ -fraction of hyperedges – the negative exponent of  $e$  in the approximation factor is 2 instead of 1 due to the fact that we are using the union of two independently rounded sets to hit hyperedges.

<sup>7</sup> Here  $\text{Opt}(\psi)$  denotes the optimal number of constraints that can be satisfied in  $\psi$ .

Furthermore, we observe that any constraint hit by  $S_1 \cup S_2$  is going to be satisfied with probability  $1/2$  under the rounding, and hence the expected fraction of constraints satisfied by the assignment returned is  $\frac{1}{2}(1 - e^{-2}) \cdot \text{Opt}(\psi)$ . Finally, observe that using the constraint  $\sum_i z_i = (1 - p)n$ , we have  $\mathbb{E}[|S_1|] = \mathbb{E}[|S_2|] = (1 - p)n$  and hence using Chernoff bound, we can argue that w.h.p. we have  $|S_1 \cup S_2| \leq 2(1 - p)n$ , and the weight of the assignment returned by the algorithm will be at least  $p(1 - o(1))n$  with high probability..

## 5 $\frac{1}{2}(1 - e^{-2})$ -hardness assuming UGC

As is standard, our UGC based matching hardness from Theorem 3 is again a dictatorship test based reduction. However, unlike the reduction for Theorem 2, here our test isn't a generalization of Håstad's 3-query test and instead uses the existence of pairwise independent distributions with certain biases that are supported on the set of accepting strings of the Hom- $r$ -Lin predicate. Formally, we show the following.

► **Lemma 6 (Informal).** *For every large odd  $r \in \mathbb{N}$ , there exists a pairwise independent permutation invariant distribution supported on the set of accepting strings of Hom- $r$ -Lin predicate, such that marginally each bit is  $(1 - 1/(r - 1))$ -biased.*

As in [5], the above distribution can be immediately used to construct a dictatorship test, as described in Figure 3.

**Input.** Long code  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$  satisfying  $\mathbb{E}_{x \sim \{0, 1\}_p^k}[f(x)] = 1 - 2p$  where  $p := 1 - 2/r$ .

**Test** Let  $\mu$  be the distribution from Lemma 6.

1. Sample row vectors  $x^{(1)}, \dots, x^{(k)} \sim \mu$  independently.
2. Sample  $(1 - \eta)$ -correlated copies  $x'_i \sim_{1-\eta} x_i$  for every  $i \in [r]$ .
3. Accept if and only if

$$\prod_{i \in [r]} f(x'_i) = 1.$$

■ **Figure 3** UGC Test.

We now briefly summarize the completeness and soundness guarantees of the test. For the completeness direction, observe that if  $f = \chi_\ell$  is a dictator function, then  $\mathbb{E}_{x \sim \{0, 1\}_p^k}[f(x)] = 1 - 2p$  i.e.,  $f$  is feasible for the test. Furthermore, with probability at least  $1 - r\eta$  we have

$$\prod_{i \in [r]} f(x'_i) = \prod_{i \in [r]} \chi_\ell(x'_i) = \prod_{i \in [r]} x_i(\ell) = 1,$$

where in the last step we used the fact that  $x^{(\ell)} = (x_i(\ell))_{i \in [k]}$  is always an accepting string for Hom- $r$ -Lin due to our choice of  $\mu$  from Lemma 6. On the other hand, for the soundness direction, suppose  $f : \{0, 1\}^k \rightarrow \{\pm 1\}$  is a long code satisfying the weight constraint having no influential coordinates<sup>8</sup>. Then as a first step, we again proceed by arithmetizing the probability of the test accepting as

<sup>8</sup> In the context of this dictatorship test, we use a function having no influential coordinates as the notion of being non-correlated with low-degree terms – this is quite standard in Unique Games based reductions [24].

## 47:12 The Biased Homogeneous $r$ -Lin Problem

$$\Pr[\text{Test Accepts}] = \frac{1}{2} + \frac{1}{2} \mathbb{E}_{x \sim \mu^{\otimes k}} \left[ \prod_{i \in [r]} f(x'_i) \right].$$

Now note that the distribution of  $(x_i)_{i \in [r]}$  is pairwise independent and hence its covariance structure matches that of the fully independent  $p$ -biased distribution  $\{0, 1\}_p^{k \otimes r}$  i.e., the distribution where  $x_1, \dots, x_r$  are independent  $p$ -biased  $k$ -length strings. This along with the fact that  $f$  has no influential coordinates allows us to pass on from  $\mu$  to be the fully independent  $p$ -biased distribution using the Invariance principle [27] i.e.,

$$\frac{1}{2} + \frac{1}{2} \mathbb{E}_{x \sim \mu^{\otimes k}} \left[ \prod_{i \in [r]} f(x'_i) \right] \approx \frac{1}{2} + \frac{1}{2} \mathbb{E}_{x \sim (\{0, 1\}_p^r)^{\otimes k}} \left[ \prod_{i \in [r]} f(x'_i) \right].$$

Finally, using the fact that the expected average weight of  $f$  is  $1 - 2p$  and using the independence of  $x'_i$ 's in the new distribution we have

$$\frac{1}{2} + \frac{1}{2} \mathbb{E}_{x \sim (\{0, 1\}_p^r)^{\otimes k}} \left[ \prod_{i \in [r]} f(x'_i) \right] = \frac{1}{2} + \frac{1}{2} (1 - 2(1 - 2/r)^r) \approx \frac{1}{2} (1 - e^{-2}),$$

which gives us the desired soundness. Our final reduction composes the above test with UNIQUEGAMES as the outer verifier. While the completeness of the reduction follows as is, additional care has to be taken in establishing the soundness of the full reduction owing to the following issue. In the setting of the full reduction from UNIQUEGAMES instance  $\mathcal{G}(V_G, E_G, [k], \{\pi_e\}_{e \in E})$ , the set of variables is defined by a collection of long codes  $\{f_v\}_{v \in V_G}$  satisfying the global bias constraint  $\mathbb{E}_v \mathbb{E}_x f_v(x) = 1 - 2p$ . Now, by combining standard techniques for analyzing UNIQUEGAMES based reductions with the soundness analysis of the test from above, we can show that for NO instances, the soundness of the reduction can be expressed as:

$$\frac{1}{2} + \frac{1}{2} \mathbb{E}_{v \sim V} [(1 - 2p_v)^r]$$

where  $p_v = \mathbb{E}_{w \sim N_G(v)} \mathbb{E}_{x \sim \{0, 1\}_p^k} [f_v(x)]$  is the local average weight of long codes around  $v$ . As before, if we could show that  $p_v \approx p$  for most choices of  $v \in V_G$ , then we would be done. However, unlike in the setting of Theorem 2, the outer UNIQUEGAMES instance cannot have strong mixing properties<sup>9</sup>, and hence here we cannot hope to show that  $p_v \approx p$  for most choices of  $v$ . Instead, by combining the fact that  $r$  is odd with a careful argument we can show that the mapping  $p_v \mapsto (1 - 2p_v)^{r-2}$  is *concave* for most points of the distribution over  $p_v$ , and hence using Jensen's inequality, we can push the expectation operator inside to show that

$$\frac{1}{2} + \frac{1}{2} \mathbb{E}_{v \sim V} [(1 - 2p_v)^r] \leq \frac{1}{2} + \frac{1}{2} (1 - 2\mathbb{E}_{v \sim V_G} [p_v])^r + o_r(1) = \frac{1}{2} (1 - e^{-2}) + o_r(1),$$

where the  $o_r(1)$ -additive factor is due to the fact that the function is concave in all but  $o_r(1)$ -mass of the distribution.

---

<sup>9</sup> In fact, even mildly expanding UNIQUEGAMES instances are known to admit polynomial time algorithms [6]

## References

- 1 Sepehr Abbasi-Zadeh, Nikhil Bansal, Guru Guruganesh, Aleksandar Nikolov, Roy Schwartz, and Mohit Singh. Sticky brownian rounding and its applications to constraint satisfaction problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 854–873. SIAM, 2020.
- 2 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. *ACM Transactions on Algorithms (TALG)*, 13(1):1–27, 2016.
- 3 Per Austrin and Johan Håstad. Randomly supported independence and resistance. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 483–492. ACM, 2009.
- 4 Per Austrin and Subhash Khot. A characterization of approximation resistance for even  $k$ -partite csps. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 187–196, 2013.
- 5 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18(2):249–271, 2009.
- 6 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *2011 IEEE 52nd annual symposium on foundations of computer science*, pages 472–481. IEEE, 2011.
- 7 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.
- 8 Amey Bhangale and Subhash Khot. Optimal inapproximability of satisfiable  $k$ -lin over non-abelian groups. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1615–1628, 2021.
- 9 Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an  $o(n^{1/4})$  approximation for densest  $k$ -subgraph. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 201–210, 2010.
- 10 Siu On Chan. Approximation resistance from pairwise-independent subgroups. *Journal of the ACM (JACM)*, 63(3):1–32, 2016.
- 11 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2006.
- 12 Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 687–696. IEEE, 2006.
- 13 Roei David, Irit Dinur, Elazar Goldenberg, Guy Kindler, and Igor Shinkar. Direct sum testing. *SIAM Journal on Computing*, 46(4):1336–1369, 2017.
- 14 Lars Engebretsen, Jonas Holmerin, and Alexander Russell. Inapproximability results for equations over finite groups. *Theoretical Computer Science*, 312(1):17–45, 2004.
- 15 U Feige and M Seltser. On the densest  $k$ -subgraph problems, 1997.
- 16 Suprovat Ghoshal and Euiwoong Lee. A characterization of approximability for biased csps. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 989–997. ACM, 2022. doi:10.1145/3519935.3520072.
- 17 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 482–491. IEEE Computer Society, 2011.
- 18 Gustav Hast. *Beating a random assignment: Approximating constraint satisfaction problems*. PhD thesis, KTH, 2005.

- 19 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- 20 Johan Håstad. Every 2-csp allows nontrivial approximation. *Comput. Complex.*, 17(4):549–566, 2008. doi:10.1007/s00037-008-0256-y.
- 21 Johan Håstad and Rajsekar Manokaran. On the hardness of approximating balanced homogeneous 3-lin. *Theory Of Computing*, 13(1):1–19, 2017.
- 22 Jonas Holmerin and Subhash Khot. A new pcg outer verifier with applications to homogeneous linear equations and max-bisection. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2004.
- 23 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775, 2002.
- 24 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.
- 25 Subhash Khot, Madhur Tulsiani, and Pratik Worah. A characterization of strong approximation resistance. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 634–643, 2014.
- 26 Rasmus Kyng and Peng Zhang. Hardness results for structured linear systems. *SIAM J. Comput.*, 49(4), 2020.
- 27 Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geometric and Functional Analysis*, 19(6):1713–1756, 2010.
- 28 Prasad Raghavendra and Tselil Schramm. Gap amplification for small-set expansion via random walks. *arXiv preprint*, 2013. arXiv:1310.1493.
- 29 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 755–764. ACM, 2010.
- 30 Prasad Raghavendra, David Steurer, and Prasad Tetali. Approximations for the isoperimetric and spectral profile of graphs and related parameters. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 631–640. ACM, 2010.
- 31 Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.
- 32 Uri Zwick. Computer assisted proof of optimal approximability results. In *SODA*, pages 496–505, 2002.

# Asymptotically Optimal Bounds for Estimating H-Index in Sublinear Time with Applications to Subgraph Counting

Sepehr Assadi  

Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Hoai-An Nguyen  

Department of Computer Science, Rutgers University, Piscataway, NJ, USA

---

## Abstract

The  $h$ -index is a metric used to measure the impact of a user in a publication setting, such as a member of a social network with many highly liked posts or a researcher in an academic domain with many highly cited publications. Specifically, the  $h$ -index of a user is the largest integer  $h$  such that at least  $h$  publications of the user have at least  $h$  units of positive feedback.

We design an algorithm that, given query access to the  $n$  publications of a user and each publication's corresponding positive feedback number, outputs a  $(1 \pm \varepsilon)$ -approximation of the  $h$ -index of this user with probability at least  $1 - \delta$  in time  $O\left(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h}\right)$ , where  $h$  is the actual  $h$ -index which is unknown to the algorithm a-priori. We then design a novel lower bound technique that allows us to prove that this bound is in fact **asymptotically optimal** for this problem in **all parameters**  $n, h, \varepsilon$ , and  $\delta$ .

Our work is one of the first in sublinear time algorithms that addresses obtaining asymptotically optimal bounds, especially in terms of the error and confidence parameters. As such, we focus on designing novel techniques for this task. In particular, our lower bound technique seems quite general – to showcase this, we also use our approach to prove an asymptotically optimal lower bound for the problem of estimating the number of triangles in a graph in sublinear time, which now is also optimal in the error and confidence parameters. This latter result improves upon prior lower bounds of Eden, Levi, Ron, and Seshadhri (FOCS'15) for this problem, as well as multiple follow-up works that extended this lower bound to other subgraph counting problems.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Sublinear time algorithms,  $h$ -index, asymptotically optimal bounds, lower bounds, subgraph counting

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.48

**Category** APPROX

**Funding** *Sepehr Assadi*: Department of Computer Science, Rutgers University. Research supported in part by a NSF CAREER Grant CCF-2047061, a gift from Google Research, and a Fulcrum award from Rutgers Research Council.

*Hoai-An Nguyen*: Research supported in part by a NSF CAREER Grant CCF-2047061.

**Acknowledgements** We thank Janani Sundaresan for helpful feedback on the presentation of our paper. We are also grateful to the anonymous reviewers of APPROX 2022 for their helpful feedback on previous work and the presentation of this paper.



© Sepehr Assadi and Hoai-An Nguyen;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 48; pp. 48:1–48:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

The *Hirsch* index, or *h*-index for short, is a metric used to measure the impact of a researcher's publications [20]. It is an integer that considers both the number of publications and citations a researcher has and is used in a number of contexts including consideration for grants and job opportunities. We can abstract out this problem by modeling each individual researcher as an array  $A[1 : n]$  where  $n$  is the number of papers they have published and  $A[i]$  is the number of citations paper  $i \in [n]$  has. The *h*-index of  $A$  is then defined as follows.

► **Definition 1.** *The **h-index** of an array  $A[1 : n]$ , denoted by  $h(A)$ , is the maximum integer  $h$  such that  $A[1 : n]$  has at least  $h$  indices,  $i_j$ , where for each  $j \in [h]$ ,  $A[i_j] \geq h$ .*

There are simple algorithms that can compute the value of  $h(A)$  for any given array  $A$  in  $O(n)$  time. For instance, we can change each entry of  $A[i]$  to  $\min\{A[i], n\}$  without changing  $h(A)$  (since  $h(A) \leq n$ ) and then run counting sort on  $A$  in linear time to sort  $A$  in decreasing order. We can then make another pass over  $A$  and output the largest index  $i \in [n]$  such that  $A[i] \geq i$  which will be equal to  $h(A)$  now that  $A$  is sorted. This solves the *h*-index problem in  $\Theta(n)$  time.

The question we focus on in this paper is whether we can solve this problem even faster than reading the entire input, namely, via a *sublinear time* algorithm, assuming we can read each single entry of  $A$  in  $O(1)$  time. There are easy observations that show that the answer to this question is *No* without relaxing the problem: deterministic algorithms cannot solve this problem in sublinear time even approximately, and randomized algorithms cannot find an exact answer<sup>1</sup>. Such observations however are commonplace when it comes to sublinear time algorithms. Our goal in this paper is thus to solve this problem allowing both randomization and approximation.

► **Result 2.** *There is an algorithm that for any array  $A$  and any  $\varepsilon, \delta \in (0, 1)$ , with probability at least  $1 - \delta$ , outputs an estimate  $\tilde{h}$  such that  $|\tilde{h} - h(A)| \leq \varepsilon \cdot h(A)$  in  $O(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)})$  time. Moreover, we prove that this algorithm is asymptotically optimal in all parameters involved.*

Result 2 gives a randomized sublinear time algorithm for a  $(1 \pm \varepsilon)$ -approximation of the *h*-index problem, where the runtime improves depending on the value of the *h*-index itself. This is quite common in sublinear time algorithms; see, e.g. [9, 11, 1] for estimating the number of subgraphs, [4] for minimum cut, or [14, 15, 10, 31] for sampling small subgraphs, among others. In all the aforementioned examples, such dependences are necessary, which is also the case for ours by the lower bound we prove.

Our Result 2, however, is quite novel from a different perspective: the obtained bounds are asymptotically optimal in *all* the parameters of the problem, including  $\varepsilon$  and  $\delta$ . We are not aware of any prior work with such strong guarantees as we will discuss in more detail in the next subsection. Moreover, as a corollary of our techniques in proving the lower bound

<sup>1</sup> A deterministic algorithm running in  $o(n)$  time cannot distinguish between an array  $A$  which is all zeros and an array  $B$  obtained from  $A$  by making  $n/2$  entries have value  $n/2$  instead. This is because the first  $n/2$  queries of the algorithm to indices of  $A$  or  $B$  can be 0 in both cases. Yet, we have  $h(A) = 0$  and  $h(B) = n/2$ . Similarly, a randomized algorithm running in  $o(n)$  time cannot distinguish between an array  $A$  with value  $n$  as every entry and an array  $B$  obtained from  $A$  by changing exactly one of the entries to  $n - 1$  instead. This can be proven for instance by using the  $\Omega(n)$  lower bound on the query complexity of the OR problem [6]. In this case  $h(A) = n$  and  $h(B) = n - 1$ .

for Result 2 with dependence on both  $\varepsilon$  and  $\delta$ , we also obtain an asymptotically optimal lower bound for the well-studied problem of counting triangles in sublinear time that now matches the dependence on  $\varepsilon$  and  $\delta$  as well, improving upon the prior work in [9, 13, 1].

## 1.1 Key Motivations

There are two key, yet disjoint, motivations behind our work that we elaborate on below.

### Measuring “impact” quickly

Consider any “publication setting” that allows for user feedback. This can range from social networks with users posting topics and others liking them all the way to the academic domain with researchers publishing papers and others citing them. A question studied frequently in social sciences is how to measure the “impact” of a single user in such a setting for many different contexts, including identifying impactful users for marketing or propagating information; see, e.g. [28] and the references therein.

One of the well-accepted measures of impact in these publication settings is the  $h$ -index measure we study in this paper [20, 28]. Given the ubiquity of massive publication settings and their evolving nature, say, social networks, we need algorithms that are able to compute the  $h$ -index of different users efficiently; see, e.g. [18] that design such algorithms in the closely related *streaming* model (which focuses on the space usage of algorithms instead of their time). Thus, a key motivation behind our Result 2 is to provide a time-efficient algorithm for this purpose. In general, it seems like a fascinating area of research to obtain efficient algorithms for measuring various notions of impact in these massive publication settings in parallel to the line of work, e.g., in [28], that searches for the “right” measure itself.

In particular, the  $h$ -index has numerous applications within network science. In [8], it is shown that when the  $h$ -index of a graph is large enough, the algorithm they design to approximate the degree distribution is sublinear. In [24], the focus is on computing coreness through iteratively using an operator that can calculate the  $h$ -index of any node to identify influential nodes: an important step in understanding a network’s dynamics and structure. Both works do not specify how their algorithm computes the  $h$ -index, so the use of our algorithm could help prevent impractical runtimes. Building on [24], [29] generalizes using an iterative  $h$ -index operator for truss and nucleus decomposition to find dense subgraphs. They use the classical linear algorithm for calculating the  $h$ -index, which therefore leaves the opportunity to use our algorithm to achieve better efficiency.

### Asymptotically optimal sublinear time algorithms

Traditionally, the work on sublinear time algorithms have been rather cavalier with the dependence on the error parameter  $\varepsilon$ , confidence parameter  $\delta$ , and logarithmic factors. It is certainly important to focus on the “high order terms” in the complexity of problems, say, in numerous works on subgraph counting; see, e.g., [9, 11, 12] and references therein. However, as already observed in [17]: “the dependence of the complexity on the approximation parameter is a key issue”. For instance, in any  $(1 \pm \varepsilon)$ -approximation algorithm, for a typical value of  $\varepsilon \sim 1\%$ , one extra factor of  $1/\varepsilon$  in the runtime translates to roughly a  $100x$  slower algorithm, which is almost always a deal breaker for the practical purposes of sublinear time algorithms! Similar considerations also apply, but perhaps to a lower extent, to having a large dependence on logarithmic factors instead of asymptotically optimal bounds. In terms of the confidence parameter,  $\delta$ , the runtime dependence of sublinear time algorithms almost always includes the term  $\ln(1/\delta)$ . It is important for practical considerations to determine whether this dependence is necessary.

Despite this, such considerations have not been studied in sublinear time algorithms. The only prior work we are aware of is the very recent work of [31] that improved the  $O(\varepsilon^{-1/2})$ -dependence of the algorithm of [14] for sampling edges  $\varepsilon$ -point-wise close to uniform to an  $O(\log(1/\varepsilon))$ -dependence. This is in stark contrast with the large body of work in related areas such as streaming [21, 23, 5], graph streaming [25, 2], compressed sensing [26, 27], sampling [22], and dynamic graph algorithms [30, 19, 3] which put emphasis on obtaining asymptotically optimal algorithms and lower bounds on all parameters.

In light of this discussion, another key motivation of our work has been to use the  $h$ -index problem as a *medium* for designing general techniques for obtaining asymptotic bounds for sublinear time algorithms in general. For instance, our algorithm involves careful subroutines that side-step typical “binary search” approaches in prior work that results in additional  $O(\varepsilon^{-1} \cdot \log n)$  terms in the runtimes of algorithms and a more careful analysis of the error that bypasses a trivial union bound which leads to additional  $O(\log n)$  factors. More importantly, we design a new lower bound technique, based on a new query complexity result that we establish, that allows us to prove lower bounds that depend on both parameters  $\varepsilon$  and  $\delta$ . This approach can now be used to replace prior sublinear time lower bounds both based on ad-hoc arguments such as the ones in [9] or the ones based on communication complexity [14, 1]. As a result, we also obtain asymptotically optimal lower bounds for the problem of counting triangles in a graph that now matches the dependence on  $\varepsilon$  and  $\delta$  as well, improving upon the prior work in [9, 13, 1].

## 1.2 Notation

For any integer  $t \geq 1$ , we define  $[t] := \{1, 2, \dots, t\}$ . For any  $p \in (0, 1)$ , we use  $\mathcal{B}(p)$  to denote the *Bernoulli* distribution with mean  $p$ . For a set  $S$  of integers, we write  $i \in_R S$  to mean  $i$  is chosen uniformly at random from  $S$ .

## 1.3 Appendix

Due to space limitations, some details and proofs marked by a star are postponed to the full version of the paper which appears on arXiv. Appendix A includes the concentration results, other basic probabilistic tools, basic definitions and tools from query complexity, and measures of distance between distributions that we use in this paper.

## 2 The Algorithm

We describe our main algorithm for the  $h$ -index problem in this section.

► **Theorem 3.** *There exists a sublinear time algorithm that given query access to an integer array  $A[1 : n]$ , approximation and confidence parameters  $\varepsilon, \delta \in (0, 1)$ , with probability at least  $1 - \delta$  outputs an estimate  $\tilde{h}$  of  $h(A)$  such that  $|\tilde{h} - h(A)| \leq \varepsilon \cdot h(A)$  in  $O(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)})$  time.*

The algorithm in Theorem 3 is a combination of a “weak” and “strong” estimator that we design. The weak estimator only outputs whether  $h(A)$  is at least as large as a given threshold, but it is efficient and can be used to provide a lower bound on  $h(A)$ . The strong estimator, which has a slower runtime, then uses the lower bound to output an estimate of  $h(A)$ . In the next two subsections, we present these two estimators and then conclude the proof of Theorem 3 through a careful combination of them that preserves the asymptotic runtime of the overall algorithm.

## 2.1 A Weak Estimator

We present an algorithm that determines with high probability whether  $h(A)$  is at least as large as a given threshold.

► **Lemma 4.** *There exists a sublinear time algorithm that given query access to an integer array  $A[1 : n]$  and an integer  $T \geq 1$  in  $O(n/T)$  time outputs an answer satisfying the following:*

- (i) *if  $h(A) \geq T$ , the answer is Large with probability at least  $1 - 1/16$ ;*
- (ii) *if  $h(A) < T/4$ , the answer is Small with probability at least  $1 - h(A)/(4T)$ ;*
- (iii) *either Small or Large can be outputted in the remaining cases.*

Let us point out the asymmetric guarantee of the algorithm: it does not underestimate  $h(A)$  with a certain constant probability while it does not overestimate  $h(A)$  with probability proportional to the “rate” of overestimation. This guarantee will be crucial in our final algorithm. We also note that the guarantee on the runtime of the algorithm is deterministic.

### 2.1.1 The Algorithm

At a high level, our algorithm, **h-index-weak-estimator**, queries random indices from  $A$  and calculates the proportion of those indices that are above a threshold representing the mid-point between a  $h$ -index of  $T/4$  and  $T$ . If the proportion is below the threshold, the algorithm outputs *Small*; otherwise, it outputs *Large*.

■ **Algorithm 1** **h-index-weak-estimator**( $A[1 : n], T$ ).

- 
- 1 Sample  $k := 64 \cdot n/T$  indices  $S$  independently and uniformly with repetition from  $[n]$ .
  - 2 Let  $X$  denote the number of indices  $i \in S$  such that  $A[i] \geq T$ .
  - 3 If  $X \geq kT/(2n)$ , output *Large*. Otherwise, output *Small*.
- 

The runtime of **h-index-weak-estimator** is simply  $O(n/T)$  as we are sampling these many indices in  $S$  and then for each  $i \in S$ , we need to query  $A[i]$ ; counting the value of  $X$  and outputting the answer can also be done in  $O(n/T)$  time, which bounds the runtime as desired.

### 2.1.2 The Analysis

We now analyze the correctness of the algorithm. For any  $j \in [k]$ , define an indicator random variable  $X_j$  which is 1 iff the  $j$ -th sample in  $S$ , namely,  $i_j \in [n]$ , satisfies  $A[i_j] \geq T$ . This way, for the counter  $X$  in the algorithm, we have  $X = \sum_{j=1}^k X_j$ . Recall that the output of the algorithm depends on the value of  $X$ . In the following, we will separately consider the value of  $X$  in the case when the output is supposed to be *Large* versus when it is supposed to be *Small*.

#### Case I: the “Large” case

We first consider the case when the output should be *Large*, or when  $h(A) \geq T$ . Thus,

$$\mathbb{E}[X] = \sum_{j=1}^k \mathbb{E}[X_j] = \sum_{j=1}^k \Pr_{i_j \in_R [n]} (A[i_j] \geq T) \geq k \cdot \frac{T}{n}, \quad (1)$$

since  $A$  consists of at least  $T$  indices with value  $\geq T$  when  $h(A) \geq T$ , and we are sampling indices  $i_j \in [n]$  for  $j \in [k]$  uniformly at random. We can similarly bound the variance of  $X$  using Fact 29 since variables  $X_j$  for  $j \in [k]$  are independent, and thus,

$$\text{Var}[X] = \text{Var}\left[\sum_{j=1}^k X_j\right] = \sum_{j=1}^k \text{Var}[X_j] \leq \sum_{j=1}^k \mathbb{E}[X_j^2] = \sum_{j=1}^k \mathbb{E}[X_j] = \mathbb{E}[X], \quad (2)$$

where the second to last equality is because for all  $j \in [k]$ ,  $X_j$  is an indicator random variable.

We use Chebyshev's inequality (Proposition 30) to finalize the proof of this case.

▷ **Claim 5 (★).** When  $h(A) \geq T$ , we have  $\Pr(\text{algorithm outputs } \textit{Small}) \leq 1/16$ .

This claim is now enough to establish property (i) in Lemma 4.

### Case II: the “Small” case

We now consider the case when the output should be *Small*, namely, when  $h(A) < T/4$ . In this case, we have,

$$\mathbb{E}[X] = \sum_{j=1}^k \mathbb{E}[X_j] = \sum_{j=1}^k \Pr_{i_j \in_R[n]}(A[i_j] \geq T) < k \cdot \frac{T}{4n}, \quad (3)$$

as there are less than  $T/4$  indices in  $A$  with value  $\geq T$  when  $h(A) < T/4$ , and we are sampling indices  $i_j \in [n]$  for  $j \in [k]$  uniformly at random. We will also bound the variance of  $X$  similarly to Equation (2) but in a slightly more careful manner. By Fact 29, since variables  $X_j$  for  $j \in [k]$  are independent, we have,

$$\text{Var}[X] = \sum_{j=1}^k \text{Var}[X_j] \leq \sum_{j=1}^k \mathbb{E}[X_j] = \sum_{j=1}^k \Pr_{i_j \in_R[n]}(A[i_j] \geq T) \leq k \cdot \frac{h(A)}{n}, \quad (4)$$

where in the last inequality, we use the fact that the number of indices in  $A$  with value larger than  $T$  is at most  $h(A)$  (since we already know that  $h(A) < T$ ).

To conclude the proof, we again use Chebyshev's inequality but with a slightly different analysis.

▷ **Claim 6 (★).** When  $h(A) < T/4$ , we have  $\Pr(\text{algorithm outputs } \textit{Large}) \leq h(A)/(4T)$ .

Lemma 4 now follows from the previous two claims.

## 2.2 A Strong Estimator

We now present our second intermediate algorithm which outputs an estimate of  $h(A)$  when given the guarantee that  $h(A)$  is at least as large as a given threshold.

► **Lemma 7.** *There exists a sublinear time algorithm that given query access to an integer array  $A[1 : n]$ , an integer  $T \leq h(A)$ , and approximation parameter  $\varepsilon \in (0, 1)$ , in  $O(n/(\varepsilon^2 T))$  time outputs an estimate  $\tilde{h}$  of  $h(A)$  such that  $\Pr(|\tilde{h} - h(A)| \leq \varepsilon \cdot h(A)) \geq 2/3$ .*

*The guarantee on the runtime of the algorithm holds deterministically even when  $T > h(A)$ .*

We emphasize that while the guarantee on the runtime of the algorithm in Lemma 7 holds even when  $T > h(A)$ , we clearly have no guarantee on the correctness in this case.

■ **Algorithm 2**  $\text{h-index-strong-estimator}(A[1 : n], T, \varepsilon)$ .

- 
- 1 Sample  $k := 6n/(\varepsilon^2 T)$  indices  $S$  independently and uniformly with repetition from  $[n]$ .
  - 2 Let  $B[1 : k]$  be an array consisting of integers  $A[i]$  for  $i \in S$ .
  - 3 Return<sup>2</sup> the largest integer  $q \in [n]$  such that  $k \cdot q/n$  indices in  $B$  are at least  $q$ .
- 

### 2.2.1 The Algorithm

The algorithm, **h-index-strong-estimator**, queries a set of random indices from  $A$  and finds a scaled estimate of the  $h$ -index.

The first two lines of **h-index-strong-estimator** can be implemented in  $O(k) = O(n/(\varepsilon^2 T))$  time in a straightforward way. We show that the last step can also be implemented in  $O(k)$  time.

► **Lemma 8** ( $\star$ ). ***h-index-strong-estimator** runs in  $O(n/(\varepsilon^2 T))$  time.*

### 2.2.2 The Analysis

We prove the correctness of **h-index-strong-estimator** in this subsection. We consider each case in which the algorithm may overestimate or underestimate  $h(A)$  separately.

#### Probability of overestimation

We first bound the probability that  $\tilde{h} > (1 + \varepsilon) \cdot h(A)$ . For this event to happen, we need  $B$  to have more than  $(k/n) \cdot (1 + \varepsilon) \cdot h(A)$  indices with a value greater than  $(1 + \varepsilon) \cdot h(A)$ . We bound the probability of this happening in the following.

For any  $j \in [k]$ , define an indicator random variable  $X_j$  which is 1 iff the  $j$ -th sample  $i_j \in S$  satisfies  $A[i_j] > (1 + \varepsilon) \cdot h(A)$ . Define  $X = \sum_{j=1}^k X_j$ . By the above discussion,

$$\Pr(\tilde{h} > (1 + \varepsilon) \cdot h(A)) = \Pr(X > (k/n) \cdot (1 + \varepsilon) \cdot h(A)). \quad (5)$$

We bound the probability of the RHS of this equation.

► **Claim 9** ( $\star$ ).  $\Pr(X > (k/n) \cdot (1 + \varepsilon) \cdot h(A)) < 1/6$ .

#### Probability of underestimation

We now bound the probability that  $\tilde{h} < (1 - \varepsilon) \cdot h(A)$ . This case is essentially symmetric to the other one and is provided for completeness. For this event to happen, we need  $B$  to have less than  $(k/n) \cdot (1 - \varepsilon) \cdot h(A)$  indices with a value of at least  $(1 - \varepsilon) \cdot h(A)$ . We bound the probability of this happening in the following.

For any  $j \in [k]$ , define an indicator random variable  $Y_j$  which is 1 iff the  $j$ -th sample  $i_j \in S$  satisfies  $A[i_j] \geq (1 - \varepsilon) \cdot h(A)$ . Define  $Y = \sum_{j=1}^k Y_j$ . By the above discussion,

$$\Pr(\tilde{h} < (1 - \varepsilon) \cdot h(A)) = \Pr(Y < (k/n) \cdot (1 - \varepsilon) \cdot h(A)). \quad (6)$$

We bound the probability of the RHS of this equation.

► **Claim 10** ( $\star$ ).  $\Pr(Y < (k/n) \cdot (1 - \varepsilon) \cdot h(A)) < 1/6$ .

Combining Claim 9 and Claim 10 concludes the proof of Lemma 7.

### 2.3 The Sublinear Time h-Index-Estimator Algorithm

We now combine our weak and strong estimators to obtain a sublinear time algorithm for estimating the  $h$ -index and prove Theorem 3. The algorithm runs **h-index-weak-estimator** on smaller and smaller thresholds to determine a threshold that tightly lower bounds  $h(A)$ . Then, **h-index-strong-estimator** uses that threshold to output an estimate of  $h(A)$ . Finally, to ensure a probability of success of at least  $1 - \delta$ , we combine the median/majority trick in a rather non-black-box way using the asymmetric guarantee of **h-index-weak-estimator** in part (ii) of Lemma 4.

■ **Algorithm 3**  $\text{h-index-estimator}(A[1 : n], \varepsilon, \delta)$ .

- 
- 1 Let  $r_1 := 7 \ln(8/\delta)$  and  $r_2 := 108 \ln(8/\delta)$  and initialize  $T$  to  $n$ .
  - 2 While the *majority* answer of running **h-index-weak-estimator**( $A, T$ )  $r_1$  times returns *Small*, update  $T \leftarrow T/4$ .
  - 3 For the current value of  $T$ , run **h-index-strong-estimator**( $A, T/16, \varepsilon$ )  $r_2$  times and return the median answer as the final estimate  $\tilde{h}$ .
- 

We bound the runtime of the algorithm in the following lemma.

► **Lemma 11.** **h-index-estimator** runs in  $O\left(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)}\right)$  time with probability  $1 - \delta/2$ .

**Proof.** The runtime depends on both running **h-index-weak-estimator** on (potentially) multiple thresholds and running **h-index-strong-estimator**.

We define  $T^*$  as the “optimal” threshold: the *first* threshold given to **h-index-weak-estimator** that is not larger than  $h(A)$ , namely,  $T^* \leq h(A) < 4 \cdot T^*$ . The following claim bounds the probability that the while-loop in step two of **h-index-estimator** does not stop even after iteration  $T^*$ .

▷ **Claim 12** (\*).  $\Pr(\text{h-index-estimator continues its while-loop beyond } T^*) \leq \delta/2$ .

In the following, we condition on the complement of the event in Claim 12 which happens with probability at least  $1 - \delta/2$ , which means we have only run the while-loop until at most iteration  $T^*$ . Let  $T_0 = n, T_1 = n/4, \dots, T_t = n/4^t = T^*$  denote the thresholds in these iterations. By Lemma 4 on the runtime of **h-index-weak-estimator** we have,

$$\begin{aligned} \text{runtime of while-loop} &= \sum_{j=0}^t O\left(\frac{n}{T_j}\right) \cdot O(\ln(1/\delta)) = O\left(\frac{n}{T^*} \cdot \ln(1/\delta)\right) \cdot \sum_{j=0}^t \frac{1}{4^j} \\ &= O\left(\frac{n}{h(A)} \cdot \ln(1/\delta)\right), \end{aligned}$$

since  $T^*$  is a 4-approximation to  $h(A)$  by definition and the given geometric series converges.

Moreover, by Lemma 7 on the runtime of **h-index-strong-estimator**, in this case, we have that the last line of the algorithm takes  $O\left(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot T^*}\right) = O\left(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)}\right)$  time as well, again since  $T^*$  is a 4-approximation to  $h(A)$  (computing the medians can be done with the Median-of-Medians algorithm in  $O(r_2)$  time which is negligible in the above bounds).

All in all, we have that with probability  $1 - \delta/2$ , the algorithm runs in  $O\left(\frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)}\right)$  time. ◀



### 2.3.1 The Analysis

We prove the correctness of our algorithm in this subsection. Consider the parameter  $T^*$  defined earlier as the “optimal” threshold in the while-loop, meaning that  $T^* \leq h(A) < 4 \cdot T^*$ . There are two potential sources for error:

1. **Event  $\mathcal{E}_{weak}$ :** In the while-loop, **h-index-weak-estimator** outputs *Large* for an iteration  $T > 16T^*$ ; assuming this happens, the threshold passed to **h-index-strong-estimator** is not necessarily valid, meaning that it may not be a lower bound on  $h(A)$ .
2. **Event  $\mathcal{E}_{strong}$ :** The threshold  $T$  obtained by the runs of **h-index-weak-estimator** in the while-loop satisfies  $T \leq 16T^*$  and thus is valid, but **h-index-strong-estimator** nevertheless fails to output an accurate estimate of  $h(A)$ .

Among these, the probability of the second event is quite easy to bound using Lemma 7. Thus, in the following, we focus primarily on proving the first part.

▷ **Claim 13 (★).** In **h-index-estimator**, for any  $T = 4^\ell \cdot T^*$  for an integer  $\ell \geq 2$ ,  $\Pr(\text{the while-loop terminates at iteration } T) \leq (\delta/8)^{\ell-1}$ .

We can now bound the error probability due to event  $\mathcal{E}_{weak}$ . We have,

$$\begin{aligned} \Pr(\mathcal{E}_{weak}) &\leq \sum_{\ell \geq 2} \Pr(\text{the while-loop terminates at } T = 4^\ell \cdot T^*) \\ &\leq \sum_{\ell \geq 2} \left(\frac{\delta}{8}\right)^{\ell-1} && \text{(by Claim 13)} \\ &= \frac{(\delta/8)}{1 - (\delta/8)} < \frac{\delta}{4}. && \text{(as } \sum_{j=1}^{\infty} x^j = \frac{x}{1-x} \text{ for } x \in (0, 1)) \end{aligned}$$

We now bound the other source of error. Assuming  $\mathcal{E}_{weak}$  does not happen, for the parameter  $T$  that the while-loop terminates on, we have  $T \leq 16T^* \leq 16h(A)$  by the definition of  $T^*$ . This implies that the parameter  $T/16$  passed to **h-index-strong-estimator** is a lower bound on  $h(A)$ . Thus, by Lemma 7, each of the  $r_2$  runs of **h-index-strong-estimator** outputs a  $(1 \pm \varepsilon)$ -approximation to  $h(A)$  with probability at least  $2/3$ .

▷ **Claim 14 (★).**  $\Pr(\mathcal{E}_{strong} \mid \overline{\mathcal{E}_{weak}}) \leq \delta/4$ .

Therefore, by the union bound, the total probability of error is at most  $\delta/4 + \delta/4 = \delta/2$ . This concludes the analysis of **h-index-estimator**.

## 3 The Lower Bound

We now prove the asymptotic optimality of the bounds obtained by our algorithm in Theorem 3.

► **Theorem 15.** Any algorithm that, given query access to an array  $A[1 : n]$ , approximation parameter  $\varepsilon \in (0, 1/4)$ , and confidence parameter  $\delta \in (0, 1/100)$ , with probability  $1 - \delta$  uses at most  $q$  queries and outputs an estimate  $\tilde{h}$  such that  $|\tilde{h} - h(A)| \leq \varepsilon \cdot h(A)$  needs to satisfy  $q = \Omega(\min(n, \frac{n \cdot \ln(1/\delta)}{\varepsilon^2 \cdot h(A)}))$ .

To prove Theorem 15, we define a new problem which we call the *Popcount Thresholding Problem (PTP)* and prove a lower bound on its randomized query complexity. We will then perform a reduction from this problem to establish our theorem.

► **Remark 16.** Let us suppose that  $100 < h(A) < \ln(1/\delta) \cdot 12/\varepsilon^2$ . There exists some  $\varepsilon' > \varepsilon$  and  $\delta' > \delta$  such that  $h(A) = \ln(1/\delta') \cdot 12/\varepsilon'^2$ , and therefore,  $(n \cdot \ln(1/\delta'))/(\varepsilon'^2 \cdot h(A)) = \Omega(n)$ . So, the lower bound in Theorem 15 of  $\Omega(n)$  given the above promises on the value of  $h(A)$  is arbitrarily proven. In the following, we focus on proving that when  $h(A) \geq \ln(1/\delta) \cdot 12/\varepsilon^2$ , the randomized query complexity is  $\Omega((n \cdot \ln(1/\delta))/(\varepsilon^2 \cdot h(A)))$ .

In passing, we note that *PTP* seems quite a natural and general problem of its own independent interest; we will also use this problem in the subsequent section to prove asymptotically optimal lower bounds for the well-studied problem of estimating the number of triangles in a graph in sublinear time.

### 3.1 Popcount Thresholding Problem (PTP)

We define the Popcount Thresholding Problem as follows.

► **Problem 17.** In  $PTP_{m,k,\gamma}$ , for integers  $m, k \geq 1$  and parameter  $\gamma \in (0, 1)$ , we are given a string  $x \in \{0, 1\}^m$  sampled with equal probability from either  $D_0$  where for each index  $i \in [m]$ ,  $x_i$  is independently set to 1 with probability  $p_0 := (1 - 2\gamma) \cdot k/m$  or  $D_1$  where for each index  $i \in [m]$ ,  $x_i$  is independently set to 1 with probability  $p_1 := (1 + 2\gamma) \cdot k/m$ . The answer is Yes if  $x$  was drawn from  $D_1$ , and it is No if  $x$  was drawn from  $D_0$ .

We prove the following lemma on the query complexity of *PTP*.

► **Lemma 18.** For any  $\gamma \in (0, 1/4)$ ,  $\delta \in (0, 1/100)$ , and integers  $m \geq 1$ ,  $\ln(1/\delta) \cdot 12/\gamma^2 \leq k \leq m/6$ ,  $R_\delta(PTP_{m,k,\gamma}) \geq \frac{m \cdot \ln(1/(4\delta))}{24\gamma^2 \cdot k}$  where  $R_\delta(\cdot)$  denotes the randomized query complexity with error probability  $\delta$ .

To prove Lemma 18, we use the easy direction of Yao's minimax principle (Proposition 28) which allows us to focus on *deterministic* algorithms for *PTP* on the input distribution. As per Problem 17, the input distribution is  $D = (1/2) \cdot D_0 + (1/2) \cdot D_1$ .

► **Lemma 19** ( $\star$ ). In the distribution  $D$ ,

$$\Pr(|x|_1 > (1 - \gamma) \cdot k \mid D_0) \leq \delta \quad \text{and} \quad \Pr(|x|_1 < (1 + \gamma) \cdot k \mid D_1) \leq \delta.$$

Lemma 19 implies that any algorithm that can differentiate whether  $|x|_1 \geq (1 + \gamma) \cdot k$  or  $|x|_1 \leq (1 - \gamma) \cdot k$  with probability  $1 - \delta$  can also solve *PTP* with probability  $1 - 2\delta$ . This is simply because when  $x \sim D_\theta$  for  $\theta \in \{0, 1\}$ , with probability at most  $\delta$ ,  $|x|_1$  is not within the “right” range for such an algorithm to detect, and with another probability  $\delta$ , the algorithm may fail to output the correct answer. A union bound then implies the bound of  $1 - 2\delta$  on the probability of correctly solving *PTP*. We will use this later to prove Theorem 15 and in our extension to triangle counting.

For the rest of the proof, let  $\mathcal{A}$  be any deterministic query algorithm on  $D$  with the worst-case number of queries  $q(\mathcal{A}) := q < \frac{m \cdot \ln(1/(4\delta))}{24\gamma^2 \cdot k}$ . Without loss of generality, we assume that  $\mathcal{A}$  always makes  $q$  queries on any input (by potentially making “dummy” queries to reach  $q$  if needed). For an input  $x \sim D$ , we use  $Q_{\mathcal{A}}(x) \in \{0, 1\}^q$  to denote the string of answers returned to the query algorithm based on  $x$ .

#### Distribution of $Q_{\mathcal{A}}(x)$

A key observation is that given only  $Q_{\mathcal{A}}(x) = (b_1, \dots, b_q)$ , since  $\mathcal{A}$  is a deterministic algorithm, we will learn the value of exactly  $q$  specific entries in  $x$ :  $b_1$  is the value of the index of  $x$  queried first by  $\mathcal{A}$ , then,  $b_2$  is the value of the second index queried by  $\mathcal{A}$  where the query is uniquely

determined after seeing the answer  $b_1$  to the first query, and so on and so forth. Thus, for any choice of  $\theta \in \{0, 1\}$ , *conditioned on*  $x$  being sampled from  $D_\theta$ , for any  $i \in [m]$ , *independent* of the value of  $(b_1, \dots, b_{i-1})$ , the value of  $b_i$  is sampled from a Bernoulli distribution with mean  $p_\theta$ . This means that:

distribution  $(Q_{\mathcal{A}}(x) \mid D_0)$  is  $\mathcal{B}(p_0)^q$  and distribution  $(Q_{\mathcal{A}}(x) \mid D_1)$  is  $\mathcal{B}(p_1)^q$ .

The following claim bounds the KL-divergence (Equation (8)) between these two distributions.

▷ **Claim 20.** For any  $q \geq 1$  and  $0 < p_0, p_1 < 1/3$ , we have,  $\mathbb{D}(\mathcal{B}(p_0)^q \parallel \mathcal{B}(p_1)^q) < \ln(1/(4\delta))$ .

*Proof.* By the chain rule of KL-divergence and using the fact that both arguments are product distributions (Fact 32), we have

$$\mathbb{D}(\mathcal{B}(p_0)^q \parallel \mathcal{B}(p_1)^q) = q \cdot \mathbb{D}(\mathcal{B}(p_0) \parallel \mathcal{B}(p_1)).$$

Moreover, for each term, using Proposition 33, we have

$$\mathbb{D}(\mathcal{B}(p_0) \parallel \mathcal{B}(p_1)) \leq \frac{(p_0 - p_1)^2}{p_1 \cdot (1 - p_1)} \leq \frac{(4\gamma \cdot k/m)^2}{(1 + 2\gamma) \cdot k/m \cdot 2/3} \leq 24\gamma^2 \cdot \frac{k}{m},$$

concluding the proof.  $\triangleleft$

Let us now use Claim 20 to conclude the proof. As argued earlier, all the information that is revealed to the algorithm  $\mathcal{A}$  is the string  $Q_{\mathcal{A}}(x)$  on an input  $x \sim D$ , and its task is to distinguish whether  $x$  is sampled from  $D_0$  or  $D_1$ . By Fact 31, the best probability of success of  $\mathcal{A}$  is then:

$$\begin{aligned} \frac{1}{2} + \frac{1}{2} \cdot \|(Q_{\mathcal{A}}(x) \mid D_0) - (Q_{\mathcal{A}}(x) \mid D_1)\|_{\text{td}} &\leq 1 - \frac{1}{4} \cdot \exp(-\mathbb{D}(Q_{\mathcal{A}}(x) \mid D_0 \parallel Q_{\mathcal{A}}(x) \mid D_1)) \\ &\quad \text{(by the extension of Pinsker's inequality in Proposition 34)} \\ &= 1 - \frac{1}{4} \cdot \exp(-\mathbb{D}(\mathcal{B}(p_0)^q \parallel \mathcal{B}(p_1)^q)) \\ &\quad \text{(by the distribution of } Q_{\mathcal{A}}(x) \text{ argued earlier)} \\ &< 1 - \frac{1}{4} \cdot \exp(\ln(4\delta)) = 1 - \delta. \\ &\quad \text{(by Claim 20 as } k \leq m/6, \gamma < 1/4, \text{ and thus } p_0, p_1 < 1/3) \end{aligned}$$

This means that  $\mathcal{A}$  can succeed with probability  $< 1 - \delta$  in distinguishing between  $D_0$  and  $D_1$ . Combined with the easy direction of Yao's minimax principle (namely, an averaging principle, Proposition 28), this concludes the proof of Lemma 18.

### 3.2 Reducing PTP to the H-Index Problem

We now prove Theorem 15 via a reduction from *PTP* and our lower bound for the latter problem in Lemma 18. Suppose towards a contradiction that there is an algorithm  $\mathcal{A}_h$  for  $h$ -index that with probability  $1 - \delta/2$  uses  $o(n \ln(1/\delta)/(\varepsilon^2 h(A)))$  queries on input array  $A$  and estimates  $h(A)$  to within a  $(1 \pm \varepsilon)$ -factor. Given an instance of  $PTP_{m,k,\gamma}$ , we use  $\mathcal{A}_h$  to solve *PTP* with probability  $1 - \delta$  in *PTP-estimator*.

It is clear that the worst-case query complexity of *PTP-estimator* is  $< \tau(n, k, \varepsilon, \delta)$  by the condition on the second line of the algorithm. In terms of parameters for  $PTP_{m,k,\gamma}$ , this translates to the bound of  $\frac{m \cdot \ln(1/(4\delta))}{24\gamma^2 \cdot k}$  on the worst-case query complexity of *PTP-estimator*. In the following, we will prove that if  $\mathcal{A}_h$  truly exists, then *PTP-estimator* solves  $PTP_{m,k,\gamma}$  with probability of success at least  $1 - \delta$ . But, then *PTP-estimator* contradicts the lower bound of Lemma 18 – this implies that  $\mathcal{A}_h$  cannot exist, and we get our desired lower bound in Theorem 15.

■ **Algorithm 4** PTP-estimator( $x, k, \gamma, \delta$ ).

- 
- 1 Run  $\mathcal{A}_h$  with parameters  $n = m$ ,  $\varepsilon = \gamma$  and error  $\delta/2$  on an array  $A$  defined as follows: for any query of  $\mathcal{A}_h$  to  $A[i]$  for  $i \in [n]$ , return  $A[i] = (1 + \varepsilon) \cdot k$  if  $x_i = 1$  and return 0 otherwise.
  - 2 If at any point, the number of queries of  $\mathcal{A}_h$  reaches

$$\tau(n, k, \varepsilon, \delta) = \frac{n \cdot \ln(1/(4\delta))}{24\varepsilon^2 \cdot k},$$

stop  $\mathcal{A}_h$  and return *No* as the answer.

- 3 If we never stopped  $\mathcal{A}_h$ , return *Yes* if  $\mathcal{A}_h$  returns  $\tilde{h} \geq k - \varepsilon^2 \cdot k$ ; otherwise return *No*.
- 

► **Lemma 21.** *PTP-estimator outputs the correct answer to any instance of  $PTP_{m,k,\gamma}$  with probability at least  $1 - \delta$ .*

**Proof.** Lemma 19 implies that any algorithm that can differentiate whether  $|x|_1 \geq (1 + \gamma) \cdot k$  or  $|x|_1 \leq (1 - \gamma) \cdot k$  with probability  $1 - \delta/2$  can also solve  $PTP$  with probability  $1 - \delta$ . Therefore, it is sufficient to prove that PTP-estimator outputs *Yes* when  $|x|_1 \geq (1 + \gamma) \cdot k$  and *No* when  $|x|_1 \leq (1 - \gamma) \cdot k$  with probability at least  $1 - \delta/2$ . We consider each case of the right answer to  $PTP$  separately.

**Case I.** Suppose first that the input  $x$  to  $PTP$  is a *Yes*-instance, meaning that  $|x|_1 \geq (1 + \gamma) \cdot k$ . Consider the array  $A$  implicitly constructed by PTP-estimator. Given that  $\varepsilon = \gamma$ ,  $A$  contains at least  $(1 + \varepsilon) \cdot k$  entries each with a value of at least  $(1 + \varepsilon) \cdot k$ . Moreover, it does not contain any entry with a value larger than  $(1 + \varepsilon) \cdot k$ . Thus, we have  $h(A) = (1 + \varepsilon) \cdot k$ . By the guarantee of  $\mathcal{A}_h$  on its correctness and since  $h(A) > k$ , the probability that  $\mathcal{A}_h$  outputs a value

$$\tilde{h} < h(A) - \varepsilon \cdot h(A) = (1 + \varepsilon) \cdot k - \varepsilon \cdot k - \varepsilon^2 \cdot k = k - \varepsilon^2 \cdot k$$

or makes more than  $\tau(n, k, \varepsilon, \delta)$  queries on  $A$  and thus we stop it is at most  $\delta/2$ .

**Case II.** Suppose now that the input  $x$  to  $PTP$  is a *No*-instance, meaning that  $|x|_1 \leq (1 - \gamma) \cdot k$ . Consider the array  $A$  implicitly constructed by PTP-estimator. Given that  $\varepsilon = \gamma$ ,  $A$  contains at most  $(1 - \varepsilon) \cdot k$  non-zero entries, so  $h(A) \leq (1 - \varepsilon) \cdot k$ . Thus, by the guarantee of  $\mathcal{A}_h$  on its correctness, the probability that  $\mathcal{A}_h$  outputs a value

$$\tilde{h} \geq k - \varepsilon^2 \cdot k = (1 - \varepsilon) \cdot k + \varepsilon \cdot k - \varepsilon^2 \cdot k \geq h(A) + \varepsilon \cdot h(A)$$

is at most  $\delta/2$ . This means that if we do not stop  $\mathcal{A}_h$  (because it has made too many queries), the output will only be wrong with probability at most  $\delta/2$ . But now note that we do not have any particular guarantee on the probability that we stop  $\mathcal{A}_h$  as it is possible that  $h(A)$  is much less than  $k$  and thus the bound of  $o(n \ln(1/\delta)/(\varepsilon^2 h(A)))$  on the queries of  $\mathcal{A}_h$  will still be way less than  $\tau(n, k, \varepsilon, \delta)$ . Nevertheless, even if we stop the algorithm, we output *No* as the answer and thus make no error here. Thus, in this case also, the probability of outputting a wrong answer is  $\delta/2$  at most as desired.

This concludes the proof of Lemma 21. ◀

Theorem 15 now follows immediately from Lemma 18 and Lemma 21 as argued earlier.

## 4 Triangle Counting Problem

In this section, we switch from the main theme of our paper which was on the  $h$ -index problem and instead show an application of our lower bound techniques to the well-studied problem of subgraph counting using local queries, in particular, the triangle counting problem.

► **Problem 22.** In  $TCP_{n,m,\varepsilon}$ , for integers  $n, m \geq 1$  and parameter  $\varepsilon \in (0, 1)$ , we are given an undirected graph  $G = (V, E)$  with  $n$  vertices and  $m$  edges, and the goal is to estimate the number of triangles, namely, cliques on three vertices, in  $G$  to within a  $(1 \pm \varepsilon)$ -factor. In order to do this, we can make the following queries to the graph:

1. *Degree queries:* Given a vertex  $v \in V$ , return the degree of  $v$  ( $\deg(v)$ ).
2. *Neighbor queries:* Given a vertex  $v \in V$  and  $i \in [n]$ , return the  $i^{\text{th}}$  neighbor of  $v$  if  $i \leq \deg(v)$  and “None” otherwise.
3. *Pair queries:* Given two vertices  $u, v \in V$ , return 1 if  $(u, v) \in E$  and 0 otherwise.
4. *Edge-sample queries:* Return an edge  $e \in E$  independently and uniformly at random.

We refer the reader to [9, 11, 13, 1] and references therein for more on the background of this problem. Here, we only note that [9] designed an algorithm for this problem with time complexity  $O^*(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t})$ , where  $t$  is the number of triangles and  $O^*$  hides the dependence on  $\varepsilon$ , error probability  $\delta$ , and logarithmic factors in  $n$ . The algorithm of [9] only requires the first three types of queries mentioned above, which is generally considered the baseline for sublinear time algorithms and is referred to as the general query model. Later, by using the fourth type of query also, [1] obtained an algorithm for this problem with time complexity  $O(\frac{m^{3/2} \cdot \ln(1/\delta)}{\varepsilon^2 \cdot t})$  (the algorithm of [1] extends to counting *all* subgraphs, not just triangles, with a runtime depending on the fractional edge cover of the subgraph we are counting; see [1]).

On the lower bound front, [13], building on [9], proved a lower bound of  $\Omega(\frac{m^{3/2}}{t})$  for the triangle counting problem under the four queries mentioned. This lower bound, however, only holds for some constant  $\varepsilon$  and  $\delta$  and does not incorporate the dependence on them.

In this section, using our lower bound for the  $PTP$  problem in Lemma 18, we will improve the lower bound of [13] and obtain a lower bound that matches the algorithmic bounds of [1], settling the asymptotic complexity of the triangle counting problem in all parameters involved.

► **Theorem 23.** Any algorithm that given access to an undirected graph  $G = (V, E)$  through degree, neighbor, pair, and edge-sample queries, approximation parameter  $\varepsilon \in (0, 1/4)$ , and confidence parameter  $\delta \in (0, 1/100)$ , outputs an estimate  $\tilde{t}$  of the number of triangles,  $t$ , in  $G$  such that  $\Pr(|\tilde{t} - t| \leq \varepsilon \cdot t) \geq 1 - \delta$  requires  $\Omega(\min(m, \frac{m^{3/2} \cdot \ln(1/\delta)}{\varepsilon^2 \cdot t}))$  queries to the graph provided that  $t = o(\varepsilon \cdot m)$ .

Similarly to the  $h$ -index problem, we prove Theorem 23 via a reduction from  $PTP$  and our lower bound for that problem in Lemma 18.

► **Remark 24.** For concreteness, we focused on proving a lower bound only for the triangle counting problem as a representative of the wider family of subgraph counting problems. However, by using our  $PTP$  in place of the lower bound arguments in [11] and [1], one can also extend their lower bounds to asymptotically optimal bounds (matching the algorithm of [1]) for larger cliques as well as odd-cycles.

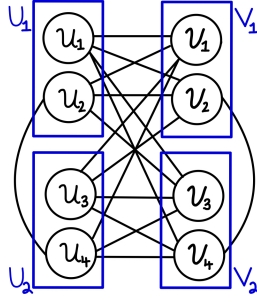
► **Remark 25.** To avoid confusion, in the rest of this proof, we use  $m$  to denote the number of edges in the triangle counting problem and instead use  $M$  (in place of the original  $m$ ) for the dimension of the  $PTP$  problem.

Suppose towards a contradiction that there is an algorithm  $\mathcal{A}_t$  for triangle counting that queries input undirected graph,  $G$ , and estimates  $t$  to within a  $(1 \pm \varepsilon)$ -factor with probability at least  $1 - \delta/2$  using  $o(m^{3/2} \ln(1/\delta)/(\varepsilon^2 t))$  queries. Given an instance of  $PTP_{M,k,\gamma}$ , we use  $\mathcal{A}_t$  to solve  $PTP$  with probability  $1 - \delta$ .

Define  $M = (\sqrt{m}/2)^2 = m/4$ . We define a mapping from inputs of  $PTP$ ,  $x \in \{0, 1\}^M$ , to  $G_x(V, E)$  on  $n = 2\sqrt{m}$  vertices and  $m$  edges.

- Let the vertices of  $G_x$  consist of two sets,  $U \cup V$ , such that  $U = \{u_1, \dots, u_{\sqrt{m}}\}$  and  $V = \{v_1, \dots, v_{\sqrt{m}}\}$ . There is no overlap between the two sets, so  $U \cap V = \emptyset$ . Let  $U$  consist of two sets,  $U_1 \cup U_2$ , such that  $U_1 = \{u_1, \dots, u_{\sqrt{m}/2}\}$  and  $U_2 = \{u_{\sqrt{m}/2+1}, \dots, u_{\sqrt{m}}\}$ . Similarly, let  $V$  consist of two sets,  $V_1 \cup V_2$ , such that  $V_1 = \{v_1, \dots, v_{\sqrt{m}/2}\}$  and  $V_2 = \{v_{\sqrt{m}/2+1}, \dots, v_{\sqrt{m}}\}$ .
- We view  $x$  as being indexed by pairs  $i \in [\sqrt{m}/2], j \in [\sqrt{m}/2+1, \sqrt{m}]$  such that  $i < j$ . Now, we add edges in the following way. If  $x_{ij} = 1$ ,  $G_x$  contains edges  $(u_i, u_j) \in U_1 \times U_2$  and  $(v_i, v_j) \in V_1 \times V_2$ . If  $x_{ij} = 0$ ,  $G_x$  contains edges  $(u_i, v_j) \in U_1 \times V_2$  and  $(v_i, u_j) \in V_1 \times U_2$ . Additionally, for each vertex  $u_1 \in U_1$  and  $v_1 \in V_1$ ,  $G_x$  contains edge  $(u_1, v_1)$ . For each vertex  $u_2 \in U_2$  and  $v_2 \in V_2$ ,  $G_x$  contains edge  $(u_2, v_2)$ . There are no other edges that are added.

See Figure 1 for an illustration.



■ **Figure 1** The graph  $G_x$  for  $x = 0001$ . The bits are indexed by the vertex pairs  $(13, 14, 23, 24)$ .

We call the reduction algorithm **PTP-estimator-two**.

It is clear that the worst-case query complexity of **PTP-estimator-two** is  $< \tau(m, k, \varepsilon, \delta)$ . In terms of parameters for  $PTP_{M,k,\gamma}$ , this translates to the bound of  $\frac{M \cdot \ln(1/(4\delta))}{24\gamma^2 \cdot k}$  on the worst-case query complexity of **PTP-estimator-two**. In the following, we will prove that if  $\mathcal{A}_t$  exists, then **PTP-estimator-two** solves  $PTP_{M,k,\gamma}$  with probability of success at least  $1 - \delta$ . But then, **PTP-estimator-two** contradicts the lower bound of Lemma 18 which implies that  $\mathcal{A}_t$  cannot exist, and we get our desired lower bound in Theorem 23.

We note that in the following lemma, the lower bound on  $k$  and upper bound on  $\varepsilon$  is benign as otherwise the  $\Omega(m)$  part of our lower bound in Theorem 23 should instead kick in.

► **Lemma 26.** *PTP-estimator-two outputs the correct answer to any instance of  $PTP_{M,k,\gamma}$  with probability at least  $1 - \delta$  as long as  $k = \omega(\ln(1/\delta)/\varepsilon^2)$ ,  $k = o(\varepsilon \cdot m)$ , and  $\varepsilon = \omega(1/\sqrt{m})$ .*

**Proof.** Lemma 19 implies that any algorithm that can differentiate whether  $|x|_1 \geq (1 + \gamma) \cdot k$  or  $|x|_1 \leq (1 - \gamma) \cdot k$  with probability  $1 - \delta/2$  can also solve  $PTP$  with probability  $1 - \delta$ . Therefore, it is sufficient to prove that **PTP-estimator-two** outputs *Yes* when  $|x|_1 \geq (1 + \gamma) \cdot k$  and *No* when  $|x|_1 \leq (1 - \gamma) \cdot k$  with probability at least  $1 - \delta/2$ .

---

**Algorithm 5** PTP-estimator-two( $x \in \{0, 1\}^M, k, \gamma, \delta$ ).

---

- 1 Run  $\mathcal{A}_t$  with parameters  $n = 2\sqrt{m}$ ,  $m = 4M$ ,  $\varepsilon = \gamma$ , and error  $\delta/2$  on an undirected graph  $G$  defined as follows:
- 2 *Degree queries.* For any degree query of  $\mathcal{A}_t$ , return  $\sqrt{m}$ .
- 3 *Neighbor queries.* For any neighbor query of  $\mathcal{A}_t$ , do the following. Assume w.l.o.g. that we get a vertex  $u_i \in U_1$  and want to find the  $k^{th}$  neighbor. If  $k \leq \sqrt{m}/2$ , return  $v_i$ . Otherwise, set  $j \leftarrow k$ . Then, if  $x_{ij}$  is 1, return  $u_j$ ; else,  $v_j$ .
- 4 *Pair queries.* For any pair query of  $\mathcal{A}_t$ , if an edge between a vertex  $u \in U_1$  and a vertex  $v \in V_1$  or between  $u \in U_2$  and  $v \in V_2$  is queried, return 1. If an edge between any two vertices in  $U_1, U_2, V_1$ , or  $V_2$  is queried, return 0. Else, for some query  $(u_i, v_j)$  such that  $i < j$ , return  $\neg x_{ij}$ . For some query  $(u_i, u_j)$  such that  $i < j$ , return  $x_{ij}$ .
- 5 *Edge-sample queries.* For any random edge-sample query made by  $\mathcal{A}_t$ , uniformly at random pick a vertex  $v \in V$  and then uniformly at random pick one of its neighbors  $u$ . Return the edge  $(u, v)$ .
- 6 If at any point, the number of queries of  $\mathcal{A}_t$  reaches

$$\tau(m, k, \varepsilon, \delta) = \frac{m \cdot \ln(1/(4\delta))}{9600\varepsilon^2 \cdot k},$$

stop  $\mathcal{A}_t$  and return *No* as the answer.

- 7 If we never stopped  $\mathcal{A}_t$ , return *Yes* if  $\mathcal{A}_t$  returns  $\tilde{t} \geq 2k(\sqrt{m} - 2)(1 - \varepsilon^2)$ ; otherwise, return *No*.
- 

Within  $G_x$ , we will define **red edges**. Let the red edges include any edges between any two vertices  $\in U_1$ . The set of red edges will also include any edges between any two vertices  $\in V_1$ . For every vertex  $v$ , we define **reddeg**( $v$ ) as the number of red edges incident on  $v$ .

We consider each case of the right answer to *PTP* separately.

**Case 1.** Suppose first that the input  $x$  to *PTP* is a *Yes*-instance, meaning that for each index  $i \in [M]$ ,  $x_i$  was set to 1 independently with probability  $(1 + 2\gamma) \cdot k/M$ . Consider the graph  $G$  implicitly constructed by **PTP-estimator-two**. For every bit set to 1 in  $x$ , there are two red edges in  $G_x$ . Each red edge  $(u, v)$  creates  $(\sqrt{m} - 2) - \text{reddeg}(u) - \text{reddeg}(v)$  triangles.

We want to ensure that in the *Yes*-instance, there are enough triangles. We first lower bound the total number of red edges. Since the number of red edges corresponds to  $|x|_1$ , we can use Lemma 19. By the choice of  $k = \omega(\ln(1/\delta)/\varepsilon^2)$ , we can see that the probability that  $|x|_1 < (1 + \gamma) \cdot k$  is bounded by  $\delta/2$ . Now, we bound for each edge,  $(u, v)$ ,  $\text{reddeg}(u) + \text{reddeg}(v)$ . Let us first bound the number of red edges incident on each vertex.

▷ **Claim 27.** When  $x$  is a *Yes*-instance, for each vertex  $v$ ,  $\Pr(\text{reddeg}(v) > \varepsilon/3 \cdot \sqrt{m}) \leq \delta/\sqrt{m}$ .

*Proof.* For each vertex  $v$ , the probability of an edge incident on it being red is  $(1 + 2\varepsilon) \cdot k/(m/4)$  and there are potentially  $\sqrt{m}/2$  red edges. Therefore,  $\mathbb{E}[\text{reddeg}(v)] = (1 + 2\varepsilon) \cdot k/(m/4) \cdot \sqrt{m}/2$ . By the lower bound on  $k$ ,  $\mathbb{E}[\text{reddeg}(v)] \leq \varepsilon/4 \cdot \sqrt{m}$ . We now use the Chernoff bound (Proposition 30) to bound the probability that  $\text{reddeg}(v)$  is too large and have

$$\Pr(\text{reddeg}(v) > \varepsilon/3 \cdot \sqrt{m}) \leq \exp\left(-\frac{(1/3)^2 \cdot \mathbb{E}[\text{reddeg}(v)]}{3}\right) \leq \delta/\sqrt{m}$$

where the last inequality is because of the lower bound on  $\varepsilon$ . ◁



Claim 27 implies that any edge  $(u, v)$ ,  $\text{reddeg}(u) + \text{reddeg}(v)$  is at most  $2/3 \cdot \varepsilon/\sqrt{m}$ . Thus, by the guarantee of  $\mathcal{A}_t$  on its correctness, the probability that  $\mathcal{A}_t$  outputs a value

$$\tilde{t} < t - \varepsilon \cdot t \leq 2(\sqrt{m} - 2)(1 + \varepsilon) \cdot k - \varepsilon \cdot 2(\sqrt{m} - 2)(1 + \varepsilon) \cdot k = 2k(\sqrt{m} - 2)(1 - \varepsilon^2)$$

is at most  $\delta/2$ . This means that *if* we do not stop  $\mathcal{A}_t$  (because it has made too many queries), the output will only be wrong with probability at most  $\delta/2$ . Additionally, since  $t/(2(\sqrt{m} - 2)) > k$  and the number of queries made by  $\mathcal{A}_t$  is supposed to be  $o(m^{3/2} \ln(1/\delta)/(\varepsilon^2 t))$ ,  $\mathcal{A}_t$  will never make more than  $\tau(m, k, \varepsilon, \delta)$  queries on  $G$ . Therefore, in this case, the probability of outputting a wrong answer is at most  $\delta/2$  as desired.

**Case II.** Suppose instead that the input  $x$  to *PTP* is a *No*-instance, meaning that for each index  $i \in [M]$ ,  $x_i$  was set to 1 independently with probability  $(1 - 2\gamma) \cdot k/M$ . Consider the graph  $G$  *implicitly* constructed by *PTP-estimator-two*. Every red edge can create at most  $(\sqrt{m} - 2)$  triangles with vertices on the other side of the bipartition.

We first bound the total number of red edges. Since the number of red edges corresponds to  $|x|_1$ , we can use Lemma 19. By the choice of  $k = \omega(\ln(1/\delta)/\varepsilon^2)$ , we can see that the probability that  $|x|_1 > (1 - \gamma) \cdot k$  is bounded by  $\delta/2$ . Therefore, by the guarantee of  $\mathcal{A}_t$  on its correctness, the probability that  $\mathcal{A}_t$  outputs a value

$$\tilde{t} \geq 2k(\sqrt{m} - 2)(1 - \varepsilon^2) = 2(\sqrt{m} - 2)(1 - \varepsilon) \cdot k + \varepsilon \cdot 2(\sqrt{m} - 2)(1 - \varepsilon) \cdot k \geq t + \varepsilon \cdot t$$

is at most  $\delta/2$ . This means that *if* we do not stop  $\mathcal{A}_t$  (because it has made too many queries), the output will only be wrong with probability at most  $\delta/2$ . But now note that we do not have any particular guarantee on the probability that we stop  $\mathcal{A}_t$  since it is possible that  $t/(2(\sqrt{m} - 2))$  is much less than  $k$  and thus the bound of  $o(m^{3/2} \ln(1/\delta)/(\varepsilon^2 t))$  on the queries of  $\mathcal{A}_t$  will still be much less than  $\tau(m, k, \varepsilon, \delta)$ . Nevertheless, even if we stop the algorithm, we output *No* as the answer and thus make no error here. Thus, in this case also, the probability of outputting a wrong answer is  $\delta/2$  at most as desired.

This concludes the proof of Lemma 26. ◀

---

## References

- 1 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 2 Sepehr Assadi and Vihan Shah. An asymptotically optimal algorithm for maximum matching in dynamic streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 3 Sayan Bhattacharya, Fabrizio Grandoni, Janardhan Kulkarni, Quanquan C. Liu, and Shay Solomon. Fully dynamic  $(\Delta + 1)$ -coloring in  $O(1)$  update time. *ACM Trans. Algorithms*, 18(2):10:1–10:25, 2022.
- 4 Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar. Query complexity of global minimum cut. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 6:1–6:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

- 5 Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. An optimal algorithm for large frequency moments using  $\tilde{O}(n^{1-2/k})$  bits. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 531–544. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.
- 6 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.
- 7 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.
- 8 Talya Eden, Shweta Jain, Ali Pinar, Dana Ron, and C. Seshadhri. Provable and practical approximations for the degree distribution using sublinear graph samples. *CoRR*, abs/1710.08607, 2017. [arXiv:1710.08607](https://arxiv.org/abs/1710.08607).
- 9 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 614–633. IEEE Computer Society, 2015.
- 10 Talya Eden, Saleet Mossel, and Ronitt Rubinfeld. Sampling multiple edges efficiently. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 51:1–51:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 11 Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of  $k$ -cliques in sublinear time. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 722–734. ACM, 2018.
- 12 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of  $k$ -cliques in low-arboricity graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1467–1478. SIAM, 2020.
- 13 Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 14 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASICS*, pages 7:1–7:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 15 Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 45:1–45:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 16 Alison L Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002.
- 17 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- 18 Priya Govindan, Morteza Monemizadeh, and S. Muthukrishnan. Streaming algorithms for measuring  $h$ -impact. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '17*, pages 337–346, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3034786.3056118.

- 19 Monika Henzinger and Pan Peng. Constant-time dynamic  $(\Delta+1)$ -coloring. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 53:1–53:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 20 Jorge E. Hirsch. An index to quantify an individual’s scientific research output. *Proc. Natl. Acad. Sci. USA*, 102(46):16569–16572, 2005.
- 21 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In Jan Paredaens and Dirk Van Gucht, editors, *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52. ACM, 2010.
- 22 Michael Kapralov, Jelani Nelson, Jakub Pachocki, Zhengyu Wang, David P. Woodruff, and Mobin Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 475–486. IEEE Computer Society, 2017.
- 23 Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 623–638. Springer, 2013.
- 24 Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H. Eugene Stanley. The H-index of a network node and its relation to degree and coreness. *Nature Communications*, 7(1):1–7, April 2016. doi:10.1038/ncomms10168.
- 25 Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1844–1860, 2019.
- 26 Eric Price and David P. Woodruff.  $(1 + \epsilon)$ -approximate sparse recovery. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 295–304. IEEE Computer Society, 2011.
- 27 Eric Price and David P. Woodruff. Lower bounds for adaptive sparse recovery. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 652–663. SIAM, 2013.
- 28 Fabián Riquelme and Pablo Gonzalez Cantergiani. Measuring user influence on twitter: A survey. *Inf. Process. Manag.*, 52(5):949–975, 2016.
- 29 Ahmet Erdem Sariyüce, C. Seshadhri, and Ali Pinar. Local algorithms for hierarchical dense subgraph discovery. *Proc. VLDB Endow.*, 12(1):43–56, 2018. doi:10.14778/3275536.3275540.
- 30 Shay Solomon. Fully dynamic maximal matching in constant update time. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334. IEEE Computer Society, 2016.
- 31 Jakub Tětek and Mikkel Thorup. Sampling and counting edges via vertex accesses. *arXiv preprint arXiv:2107.03821. To appear in STOC 2022*, 2021.
- 32 Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer series in statistics. Springer, 2009. doi:10.1007/b13794.
- 33 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977.

## A Detailed Preliminaries

### A.1 Basics of Query Complexity

We use the basics of query complexity to establish our lower bounds on the runtime of sublinear algorithms (as the number of queries made to the input is always a lower bound on the runtime).

Let  $f : \{0, 1\}^n \mapsto \{0, 1\}$  be any Boolean function. A query algorithm for  $f$  on any input  $x$  can *query* the values of  $x_i$  for  $i \in [n]$  and determine the value of  $f(x)$  with a minimal number of queries. We will work with the following definitions:

- **Randomized query complexity:** For any  $\delta \in (0, 1)$ ,  $R_\delta(f)$  denotes the worst-case number of queries made by the best randomized algorithm that computes  $f$  on any input with probability of success at least  $1 - \delta$ .
- **Distributional query complexity:** For any  $\delta \in (0, 1)$  and any distribution  $\mu$  on  $\{0, 1\}^n$ ,  $D_{\mu, \delta}(f)$  denotes the worst-case number of queries made by the best deterministic algorithm that computes  $f$  on inputs sampled from  $\mu$  with probability of success at least  $1 - \delta$ .

Yao's minimax principle [33] relates these two measures.

- **Proposition 28** (Yao's minimax principle [33]). *For any  $f : \{0, 1\}^n \mapsto \{0, 1\}$  and  $\delta \in (0, 1)$ :*
- (i) Easy direction (averaging argument): *For any distribution  $\mu$  on  $\{0, 1\}^n$ ,  $D_{\mu, \delta}(f) \leq R_\delta(f)$ .*
  - (ii) Hard direction (duality): *There is some distribution  $\mu^*$  on  $\{0, 1\}^n$  such that  $D_{\mu^*, \delta}(f) = R_\delta(f)$ .*

### A.2 Basic Probabilistic Tools

We use the linearity of variance of independent random variables.

- **Fact 29.** *For any two independent random variables  $X$  and  $Y$ ,  $\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y]$ .*

The following proposition lists the standard concentration inequalities we use in this paper.

- **Proposition 30** (Concentration Inequalities; cf. [7]).

- (i) Chebyshev's inequality: *For any random variable  $X$  and  $t > 0$ ,*

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\text{Var}[X]}{t^2}.$$

- (ii) Chernoff bound: *Suppose  $X_1, \dots, X_n$  are  $n$  independent random variables in  $[0, 1]$  and define  $X := \sum_{i=1}^n X_i$ . Then, for any  $\varepsilon \in (0, 1)$  and  $\mu \geq \mathbb{E}[X]$ ,*

$$\Pr(X > (1 + \varepsilon) \cdot \mu) \leq \exp\left(-\frac{\varepsilon^2 \cdot \mu}{3}\right) \text{ and } \Pr(X < (1 - \varepsilon) \cdot \mu) \leq \exp\left(-\frac{\varepsilon^2 \cdot \mu}{3}\right).$$

*Moreover, for any  $t \geq 1$  and  $\mu \geq \mathbb{E}[X]$ ,  $\Pr(|X - \mathbb{E}[X]| \geq t \cdot \mu) \leq 2 \cdot \exp\left(-\frac{t \cdot \mu}{3}\right)$ .*

### A.3 Measures of Distance Between Distributions

We use two main measures of distance (or divergence) between distributions, namely the *total variation distance* and the *Kullback-Leibler divergence* (KL-divergence).

### Total variation distance

We denote the total variation distance between two distributions  $\mu$  and  $\nu$  on the same support  $\Omega$  by  $\|\mu - \nu\|_{\text{tvd}}$ , defined as:

$$\|\mu - \nu\|_{\text{tvd}} := \max_{\Omega' \subseteq \Omega} (\mu(\Omega') - \nu(\Omega')) = \frac{1}{2} \cdot \sum_{x \in \Omega} |\mu(x) - \nu(x)|. \quad (7)$$

We use the following basic property of total variation distance.

► **Fact 31.** *Suppose  $\mu$  and  $\nu$  are two distributions with same support  $\Omega$ ; then, given a single sample from either  $\mu$  or  $\nu$ , the best probability of successfully deciding whether  $s$  came from  $\mu$  or  $\nu$  is  $\frac{1}{2} + \frac{1}{2} \cdot \|\mu - \nu\|_{\text{tvd}}$ .*

### KL-divergence

For two distributions  $\mu$  and  $\nu$  over the same probability space, the *Kullback-Leibler divergence* between  $\mu$  and  $\nu$  is denoted by  $\mathbb{D}(\mu \parallel \nu)$  and defined as:

$$\mathbb{D}(\mu \parallel \nu) := \mathbb{E}_{a \sim \mu} \left[ \log \frac{\Pr_{\mu}(a)}{\Pr_{\nu}(a)} \right]. \quad (8)$$

A key property of KL-divergence is that it satisfies a chain rule.

► **Fact 32** (Chain rule for KL-divergence). *Given two distributions  $p(x_1, \dots, x_t)$  and  $q(x_1, \dots, x_t)$  on  $t$ -tuples, we have,*

$$\mathbb{D}(p \parallel q) = \sum_{i=1}^t \mathbb{E}_{p(x_{<i})} \mathbb{D}(p(x_i \mid x_{<i}) \parallel q(x_i \mid x_{<i})).$$

*In particular, if  $p$  and  $q$  are product distributions, then,*

$$\mathbb{D}(p \parallel q) = \sum_{i=1}^t \mathbb{D}(p(x_i) \parallel q(x_i)).$$

The following result gives a simple upper bound for the KL-divergence of two Bernoulli distributions that we shall use in our proofs.

► **Proposition 33** (KL-divergence on Bernoulli distributions; c.f. [16, Theorem 5]). *For any  $0 < p, q < 1$ , the following is true:*

$$\mathbb{D}(\mathcal{B}(p) \parallel \mathcal{B}(q)) \leq \frac{(p - q)^2}{q \cdot (1 - q)}.$$

We shall also use the following extension of Pinsker's inequality to relate total variation distance and Kullback-Leibler divergence.

► **Proposition 34** (c.f. [32], p. 88-89). *Given two distributions  $\mu$  and  $\nu$  over the same discrete support,  $\|\mu - \nu\|_{\text{tvd}} \leq 1 - \frac{1}{2} \exp(-\mathbb{D}(\mu \parallel \nu))$ .*

# Maximizing a Submodular Function with Bounded Curvature Under an Unknown Knapsack Constraint

Max Klimm ✉

Institute for Mathematics, Technische Universität Berlin, Germany

Martin Knaack ✉

Institute for Mathematics, Technische Universität Berlin, Germany

---

## Abstract

This paper studies the problem of maximizing a monotone submodular function under an unknown knapsack constraint. A solution to this problem is a policy that decides which item to pack next based on the past packing history. The robustness factor of a policy is the worst case ratio of the solution obtained by following the policy and an optimal solution that knows the knapsack capacity. We develop an algorithm with a robustness factor that is decreasing in the curvature  $c$  of the submodular function. For the extreme cases  $c = 0$  corresponding to a modular objective, it matches a previously known and best possible robustness factor of  $1/2$ . For the other extreme case of  $c = 1$  it yields a robustness factor of  $\approx 0.35$  improving over the best previously known robustness factor of  $\approx 0.06$ .

**2012 ACM Subject Classification** Mathematics of computing → Submodular optimization and polymatroids; Theory of computation → Packing and covering problems; Theory of computation → Online algorithms

**Keywords and phrases** submodular function, knapsack, approximation algorithm, robust optimization

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.49

**Category** APPROX

**Acknowledgements** The authors wish to thank Daniel Schmidt genannt Waldschmidt for fruitful discussion.

## 1 Introduction

This paper is concerned with the problem

$$\text{maximize } \left\{ f(S) \mid S \subseteq N \text{ and } \sum_{i \in S} s(i) \leq C \right\} \quad (1)$$

of maximizing a submodular, monotone, and normalized function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  under a knapsack constraint, where  $N$  is a finite set of items,  $s(i) \in \mathbb{R}_{>0}$  is the size of item  $i \in N$ , and  $C \in \mathbb{R}_{>0}$  is a knapsack capacity. This optimization problem is an important abstraction of many problems that appear in various applications, such as facility location (Cornuéjols et al. [4]), sensor placement (Krause and Guestrin [12], Krause et al. [13]), marketing in social networks (Kempe et al. [10]), and maximum entropy sampling (Lee [14]).

For the special case of a cardinality constraint where  $s(i) = 1$  for all  $i \in N$ , a straightforward greedy algorithm by Nemhauser et al. [17] computes a solution with an approximation guarantee of  $1 - 1/e$  and this ratio is best possible for any polynomial algorithm unless  $P = NP$  (Feige [7]). For the case of a general knapsack constraint, combining the greedy algorithm with a partial enumeration of all subsolutions with at most three items yields the same approximation guarantee (Sviridenko [19]).



© Max Klimm and Martin Knaack;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 49; pp. 49:1–49:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

While these results are tight, the algorithms often perform much better than their theoretical guarantees. In order to explain and quantify this phenomenon, Conforti and Cornuéjols [3] introduce the concept of the *curvature* of a submodular function. Recall that a function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  is submodular if the marginal increase  $f(A \cup \{u\}) - f(A)$  of an element  $u \in N \setminus A$  is non-increasing as  $A$  increases. The curvature  $c \in [0, 1]$  measures how much this marginal increase of an item  $u$  varies when varying  $A$  and is defined as

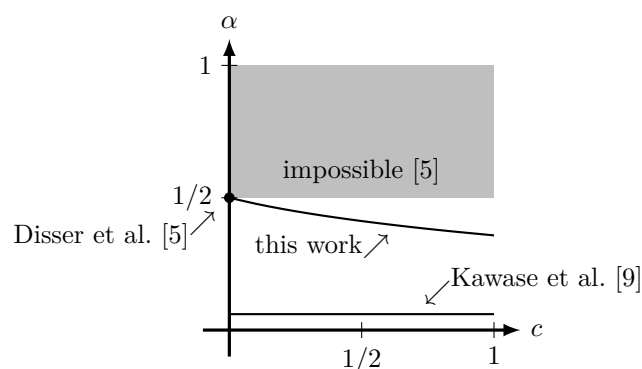
$$c = 1 - \min_{j \in N} \frac{f(N) - f(N \setminus \{j\})}{f(\{j\})},$$

where we further used that  $f$  is normalized, i.e.,  $f(\emptyset) = 0$ . It is easy to see that  $c = 0$  if and only if the function is modular (i.e., linear). The other extreme case  $c = 1$  is, e.g., attained when  $f$  is the rank function of a matroid. Conforti and Cornuéjols [3] show that the greedy algorithm for the cardinality constraint case has an improved approximation guarantee of  $(1 - e^{-c})/c$ . A more sophisticated algorithm for the same problem by Sviridenko et al. [20] achieves an even better approximation guarantee of  $1 - c/e - \varepsilon$  for any  $\varepsilon > 0$ .

In all of the results above it is assumed that all data of the problem (1) is given completely. In this paper, we consider a variant of the problem where the set of items  $N$ , their sizes  $s(i) \in \mathbb{R}_{>0}$ , and the function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  are known, but the knapsack capacity  $C \in \mathbb{R}_{>0}$  is unknown. In this context, a solution to the problem is a policy that decides which item to pack next, based on the previous packing history. More formally, a policy is a binary decision tree where nodes correspond to items with the property that no item appears more than once on a path from the root to a leaf. The item at the root of the tree is the item that is attempted to be packed first. If it fits, it is irrevocably included in the solution, the (unknown) capacity is reduced by the size of the item, and the solution proceeds with the left subtree of the decision tree. If the item does not fit, it is discarded, the (unknown) capacity stays the same, and the solution proceeds with the right subtree. This process stops after a leaf is reached. The assumption that the policy can resume packing smaller items after a larger item does not fit is suitable when the knapsack capacity is interpreted as a monetary budget. Generally speaking, such packing policies are desirable when packing problems of this kind have to be solved repeatedly for varying knapsack capacities. For illustration, consider the marketing problem in social networks. By analyzing the social network, a packing policy can be constructed that can then be used in order to run marketing campaigns *for all possible budgets*, without the need to rerun any optimization. In a similar vein, consider the problem of maximum entropy sampling. The Shannon entropy of a set of (dependent) random variables is a submodular function of the (index set) of the variables. Suppose that observing the realization of a random variable comes at a cost (for market research, for evaluating the data, etc.). With our algorithm, one can compute a policy that *for all budgets* allows to retrieve close to optimal information without any need to rerun the optimization for different budgets.

In the examples above, we clearly want to obtain solutions that are good for any possible capacity. We evaluate the quality of a policy in terms of its *robustness factor*. Fix an instance of (1), and a corresponding policy  $\Pi$ . For a capacity  $C \in \mathbb{R}_{>0}$ , let  $\Pi(C)$  be the set of items packed by the policy when the knapsack capacity is  $C$ , and let  $\text{OPT}(C)$  be the items included in an optimal solution for capacity  $C$ . The robustness factor is defined as  $\alpha := \inf_C f(\Pi(C))/f(\text{OPT}(C))$ . A policy with robustness factor of  $\alpha \in [0, 1]$  is called  $\alpha$ -optimal.





■ **Figure 1** Robustness factors  $\alpha$  as a function of the curvature  $c$  achieved by this and previous work.

## 1.1 Our Results and Techniques

For the case that  $f$  is modular (corresponding to the case that  $c = 0$ ), Disser et al. [5] show that every instance admits a  $1/2$ -optimal policy, and that the factor of  $1/2$  is best possible. Kawase et al. [9] consider the fully submodular case corresponding to the case  $c = 1$ . They provide a deterministic policy with robustness factor  $2(1 - 1/e)/21 \approx 0.06$  and a randomized policy with robustness factor of  $(1 - 1/e)/2 \approx 0.32$ .

We provide a deterministic polynomial algorithm that constructs a deterministic policy  $\Pi$  with a robustness factor of

$$\alpha = \frac{1 - x}{2 - (2 - c)x} \quad (2)$$

where  $x$  is the unique root in  $[0, 1]$  of the equation  $\frac{1}{c}(1 - e^{-cz}) = \frac{1-z}{2-(2-c)z}$ .

For the most general case of a submodular function with curvature  $c = 1$ , this yields a robustness factor of  $\approx 0.35$  which improves over the factor of  $\approx 0.06$  by Kawase et al.; for smaller values of  $c < 1$  the robustness factor increases and retains the optimal factor of  $\alpha = 1/2$  in the modular case when  $c = 0$ . For an illustration; see Figure 1.

A central technique for solving submodular maximization problems with a known or unknown knapsack capacity are greedy algorithms, and this work is no exception. Disser et al. [5] compare the solution obtained by their policy with a greedy algorithm called MGREEDY that either takes the greedy sequence or the first item that does not fit the knapsack anymore. As discussed by Kawase et al. [9] this approach seems difficult to apply to submodular functions because the greedy sequence is different for different sizes of the knapsack due to the substitute effects among the items for the objective. They instead single out valuable items that provide a significant ratio of the optimum solution. This approach, however, comes at the expense of a much lower robustness factor.

We circumvent this issue by analyzing a different kind of greedy algorithm that we call AGREEDY and that seems to be more compatible with robust policies. We first analyze this algorithm for the case of a known knapsack capacity in Section 3 and show that it provides an approximation guarantee as in (2). As a byproduct of our analysis, we further obtain that the MGREEDY algorithm also has the approximation guarantee as in (2). This generalizes a result of Wolsey [22] who analyzed this algorithm only for the general submodular case where  $c = 1$ . In Section 4, we then devise a robust policy that achieves a robustness ratio that is at least as good as the approximation guarantee of AGREEDY.

## 1.2 Further Related Work

The problem of maximizing a submodular function under different constraints has a long history in the optimization literature. Nemhauser et al. [18] consider the problem of maximizing a monotonic submodular function under a cardinality constraint and show that the greedy algorithm that iteratively adds an item that maximizes the increase of the objective function achieves an approximation guarantee of  $(1 - 1/e) \approx 0.63$ . Nemhauser and Wolsey [17] prove that this ratio is best possible for algorithms that have only access to  $f$  via a value oracle that can only be queried a polynomial number of times. For the special case that  $f$  is given explicitly and corresponds to a maximum coverage function, there is no better approximation possible in polynomial time, unless  $P = NP$ , as shown by Feige [7]. Wolsey [22] considers the more general problem of maximizing a submodular function under a knapsack constraint and achieves an approximation guarantee of  $1 - e^{-x} \approx 0.35$  where  $x$  is the unique root of the equation  $e^x = 2 - x$ . Sviridenko [19] shows that a combination of the greedy algorithm with a partial enumeration scheme achieves an approximation guarantee of  $1 - 1/e$ . Another way to generalize the cardinality constrained case is to allow for arbitrary matroid constraints. For this case, the greedy algorithm yields an approximation guarantee of  $1/2$ , as shown by Fisher et al. [8]. Calinescu et al. [2] achieve a  $1 - 1/e$  approximation by solving a fractional relaxation of the problem and combining it with a suitable rounding technique.

Conforti and Cornuéjols [3] introduce the curvature  $c$  as a measure for the non-linearity of a (submodular) function and show that the greedy algorithm has an approximation guarantee of  $(1 - e^{-c})/c$  for the case of a cardinality constraint and  $1/(c + 1)$  for the case of a matroid constraint. Vondrák [21] shows that the continuous greedy algorithm yields an approximation guarantee of  $(1 - e^{-c})/c$  for the case of a matroid constraint, and proves that no better approximation is possible in the value oracle model with a polynomial number of queries. Sviridenko et al. [20] give an algorithm with approximation guarantee of  $1 - c/e - \mathcal{O}(\varepsilon)$  for the problem with a matroid constraint. Yoshida [23] obtains the same approximation guarantee for the problem under a knapsack constraint. The algorithm relies on a continuous version of the greedy algorithm which seems to be incompatible with an unknown knapsack constraint since many items will be fractional during the course of the algorithm for smaller knapsack constraints. Also the distinction between small and large items which is elementary in the algorithm cannot be employed when the capacity is not known.

Packing problems with an unknown knapsack are studied by Megow and Mestre [15]. They consider the modular case and assume that the policy stops when an item does not fit the knapsack. In this setting, no constant robustness factor is achievable on all instances and Megow and Mestre provide a polynomial time approximation scheme (PTAS) for the computation of an optimal policy. Navarra and Pinotti [16] show how to construct a policy with robustness factor  $1/2$  for instances that have the property that every item fits into the empty knapsack. Disser et al. [6] consider the optimization of a fractionally subadditive objective with the additional property that every singleton set has a value of 1, and give a policy with robustness factor of  $\approx 0.30$ . For the case of an unknown cardinality constraint, there is no difference between policies that continue or stop packing after an item does not fit. Bernstein et al. [1] introduce a property on the objective function that they term accountability and that is more general than submodularity. They show that the optimal robustness factor for maximization of an accountable objective under an unknown cardinality constraint is between  $1/(1 + \phi) \approx 0.38$  where  $\phi$  is the golden ratio and  $0.46$ .

## 2 Preliminaries

### 2.1 Submodular Functions

Let  $N$  be a finite set. A function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  is called *monotone* if  $f(A) \leq f(B)$  for every  $A, B \in 2^N$  with  $A \subseteq B$ , is called *normalized* if  $f(\emptyset) = 0$ , and is called *submodular* if  $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$  for all  $A, B \in 2^N$ . For our purposes, it is without loss of generality to assume that  $f(\{j\}) > 0$  for all  $j \in N$  since an element  $j$  with  $f(\{j\}) = f(\emptyset)$ , by submodularity, has no influence on the value of  $f$  and, thus, can be removed from (1). It is well-known that a function  $f$  is submodular if and only if one of the following statements is satisfied:

$$\begin{aligned} f(A \cup \{u\}) - f(A) &\geq f(B \cup \{u\}) - f(B) \quad \text{for all } A \subseteq B \subseteq N, u \in N \setminus B, \\ f(A \cup \{u_1\}) + f(A \cup \{u_2\}) &\geq f(A \cup \{u_1, u_2\}) + f(A) \\ &\quad \text{for all } A \subset N, u_1, u_2 \in N \setminus A, u_1 \neq u_2. \end{aligned}$$

It is straightforward to verify that a submodular and monotone function satisfies the following inequality, see, e.g., Nemhauser et al. [18] for a reference

$$f(B) \leq f(A) + \sum_{u \in B \setminus A} (f(A \cup \{u\}) - f(A)) \quad \text{for all } A \subseteq B \subseteq N. \quad (3)$$

### 2.2 Curvature

The curvature of a normalized, monotone and submodular function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  is defined as

$$c = 1 - \min_{j \in N} \frac{f(N) - f(N \setminus \{j\})}{f(\{j\})}.$$

The following lemma summarizes a couple of inequalities that are valid for submodular functions with a given curvature that are easy to show yet useful for the remainder of the paper. For a proof, see Appendix A.

► **Lemma 1.** *For a normalized, monotonic, and submodular function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  with curvature  $c \in [0, 1]$ , the following inequalities are satisfied:*

- (i)  $(1 - c)f(\{j\}) \leq f(A \cup \{j\}) - f(A)$  for all  $A \subset N$  and all  $j \in N \setminus A$ ;
- (ii)  $f(A \cup B) \geq f(A) + (1 - c) \sum_{i \in B} f(\{i\})$  for all  $A, B \subset N$  with  $A \cap B = \emptyset$ .

### 2.3 Submodular Maximization under a Knapsack Constraint

An instance of the submodular maximization problem under a known knapsack constraint is given by a set of  $n$  items  $N = \{i_1, i_2, \dots, i_n\}$  where each item  $i \in N$  has a size  $s(i) \in \mathbb{R}_{>0}$ . We further have given a monotone, normalized and submodular function  $f : 2^N \rightarrow \mathbb{R}_{\geq 0}$  that assigns a value  $f(A)$  to every subset  $A \subseteq N$  of items, and a capacity  $C \in \mathbb{R}_{>0}$ . For a subset  $A \subseteq N$ , we write  $s(A) := \sum_{i \in A} s(i)$ . A solution to the problem is a set of items  $A \subseteq N$ . A solution  $A$  is called *feasible* if  $s(A) \leq C$ , and called *optimal* if  $f(A) \geq f(B)$  for every feasible solution  $B$ .

An instance of the submodular maximization problem under an unknown knapsack constraint is as above except that we do not know the capacity  $C \in \mathbb{R}_{>0}$ , i.e., we are again given a set of items  $N$ , their sizes  $s(i)$ ,  $i \in N$  and the submodular function  $f$ . A solution to this problem is a policy  $\Pi$  that governs the order in which items are added to the solution.

(a) Modified Greedy Algorithm MGREEDY.

---

```

1: $G_0 \leftarrow \emptyset; j \leftarrow 1$
2: $U \leftarrow \{i \in N \mid s(i) \leq C\}$
3: while $U \neq \emptyset$ do
4: $i_j \leftarrow \arg \max_{i \in U} \left\{ \frac{f(G_{j-1} \cup \{i\}) - f(G_{j-1})}{s(i)} \right\}$
5: if $s(G_{j-1} \cup \{i_j\}) \leq C$ then
6: $G_j \leftarrow G_{j-1} \cup \{i_j\}$
7: $U \leftarrow U \setminus \{i_j\}$
8: $j \leftarrow j + 1$
9: else
10: break
11: $k \leftarrow j - 1$
12: if $U = \emptyset$ then
13: return G_k
14: else
15: if $f(G_k) \geq f(\{i_{k+1}\})$ then
16: return G_k
17: else
18: return $\{i_{k+1}\}$

```

---

(b) Alternative Greedy Algorithm AGREEDY.

---

```

1: $G_0 \leftarrow \emptyset; j \leftarrow 1$
2: $U \leftarrow \{i \in N \mid s(i) \leq C\}$
3: while $U \neq \emptyset$ do
4: $i_j \leftarrow \arg \max_{i \in U} \left\{ \frac{f(G_{j-1} \cup \{i\}) - f(G_{j-1})}{s(i)} \right\}$
5: if $s(G_{j-1} \cup \{i_j\}) \leq C$ then
6: $G_j \leftarrow G_{j-1} \cup \{i_j\}$
7: $U \leftarrow U \setminus \{i_j\}$
8: $j \leftarrow j + 1$
9: else
10: break
11: $k \leftarrow j - 1$
12: if $U = \emptyset$ then
13: return G_k
14: else
15: if $f(G_k) \geq f(G_k \cup \{i_{k+1}\}) - f(G_k)$ then
16: return G_k
17: else
18: return $\{i_{k+1}\}$

```

---

■ **Figure 2** Greedy algorithms for maximizing a submodular function  $f$  over a knapsack constraint.

### 3 Submodular Knapsack Problem with Known Capacity

In this section, we analyze the approximation guarantee for two natural greedy algorithms that, for the sake of a better distinction, we call *modified greedy algorithm* (MGREEDY) and *alternative greedy algorithm* (AGREEDY). The modified greedy algorithm was proposed and analyzed by Wolsey [22] where he shows that it has an approximation ratio of  $1 - e^{-x} \approx 0.35$  where  $x$  is the unique root of the equation  $e^x = 2 - x$ . To the best of our knowledge, there is no better analysis of this algorithm for submodular functions with bounded curvature. The alternative greedy algorithm is a slight variation of this algorithm that we need in order to derive policies for the optimization problem with unknown knapsack constraints in Section 4.

Both algorithms first discard all items  $i$  that do not fit into an empty knapsack, i.e.,  $s(i) > C$ . Then, the algorithms start in iteration 0 with an empty solution  $G_0 = \emptyset$ . In every iteration  $j = 1, 2, \dots$ , both algorithms choose an item

$$i_j \in \arg \max \left\{ \frac{f(G_{j-1} \cup \{i\}) - f(G_{j-1})}{s(i)} \mid i \in N \setminus G_{j-1} \right\}$$

that is not yet contained in the solution  $G_{j-1}$  and maximizes the ratio of the increment of the objective function and the size of the item. If item  $i_j$  still fits the knapsack, i.e.,  $s(G_{j-1} \cup \{i_j\}) \leq C$ , then the item is added to the solution. Otherwise, the algorithm stops. Let  $k$  be the last index such that item  $i_k$  still fits into the knapsack.

Then, algorithm MGREEDY either returns the better of the solutions  $G_k$  and  $\{i_{k+1}\}$ , i.e., it either returns the maximum prefix of the greedy sequence that still fits into the knapsack, or the first item that did not fit into the knapsack anymore. The alternative greedy also either returns  $G_k$  or  $\{i_{k+1}\}$  but the rule when to return one of the solutions slightly differs. The item  $\{i_{k+1}\}$  is only returned if the increment  $f(G_k \cup \{i_{k+1}\}) - f(G_k)$  of adding it to  $G_k$  is larger than  $f(G_k)$ . In all other cases,  $G_k$  is returned.

Since MGREEDY always returns the better of the two solutions  $G_k$  and  $\{i_{k+1}\}$  while AGREEDY may also return  $G_k$  even though  $f(G_k) < f(\{i_{k+1}\})$ , it is clear that the solution returned by MGREEDY is always at least as good as the one returned by AGREEDY. Despite this fact, we are still interested in analyzing AGREEDY for two reasons. First, it turns out that AGREEDY is better suited in order to design robust packing policies for the problem with an unknown knapsack capacity. Second, it will turn out, that in the worst case, the approximation guarantees that we obtain for MGREEDY and AGREEDY are actually the same.

In the following, we fix an instance of the submodular maximization problem under a knapsack constraint with known capacity. Furthermore, we assume that the items are ordered  $N = \{i_1, i_2, \dots, i_n\}$  in the order as they would be considered by the greedy algorithms. We also set  $s_j = s(i_j)$  for all  $j \in \{1, \dots, n\}$ . We further let  $k$  be the maximal prefix of this ordering that still fits into the knapsack, i.e.,  $k = \max\{j \in \{1, \dots, n\} \mid \sum_{i=1}^j s(i) \leq C\}$ . Note that we order the items in this order beyond the  $(k+1)$ -st item when the algorithms stop. We further set  $G_j = \bigcup_{l=1, \dots, j} \{i_l\}$  and  $\delta_j = f(G_j) - f(G_{j-1})$  for all  $j \in \{1, \dots, n\}$ . We let MG denote the set of items returned by MGREEDY, and let AG denote the set of items returned by AGREEDY. Let OPT denote the set of items in an optimal solution. Additionally, let  $\chi_j$  be the indicator for  $i_j \in \text{OPT}$ , i.e.,  $\chi_j = 1$  if  $i_j \in \text{OPT}$ , and  $\chi_j = 0$ , otherwise. We further set  $S_j = s(\text{OPT} \cap G_j)$ .

Summarizing the above discussion, the following result is immediate.

► **Proposition 2.** *For every instance,  $f(\text{MG}) \geq f(\text{AG})$ .*

We first provide a lemma that bounds the increase of the value of the greedy solution from below in two different ways. The proofs use standard submodularity arguments as well as the property of the greedy sequence; for the proof see Appendix B.

► **Lemma 3.** *For all  $j \in \{1, \dots, k+1\}$ , we have*

$$\begin{aligned} \text{(i)} \quad \delta_j &\geq \frac{cs_j}{C} \left( f(\text{OPT}) - \sum_{m=1}^{j-1} \delta_m \right) + \frac{(1-c)s_j}{C - S_{j-1}} \left( f(\text{OPT}) - \sum_{m=1}^{j-1} \chi_m \delta_m \right), \\ \text{(ii)} \quad \delta_j &\geq \frac{s_j}{C - (1-c)s(G_{j-1})} \left( f(\text{OPT}) - \sum_{m=1}^{j-1} \delta_m \right). \end{aligned}$$

The following theorem bounds the value of every prefix of the greedy sequence  $f(G_j)$  in terms of  $f(\text{OPT})$ . For the proof, we use inductive arguments together with Lemma 3; see Appendix C.

► **Theorem 4.** *For all  $j \in \{1, \dots, k+1\}$  we have  $f(G_j) \geq \frac{1}{c} \left( 1 - \exp(-c \frac{s(G_j)}{C}) \right) f(\text{OPT})$ .*

For  $j = k+1$  we can derive from Theorem 4 that

$$f(\text{AG}) \geq \frac{1}{2} f(G_{k+1}) \geq \frac{1}{2c} \left( 1 - e^{-c} \right) f(\text{OPT}),$$

but we can improve the robustness factor with the ideas Wolsey [22] uses for MGREEDY in the general submodular case. Specifically, we obtain the following approximation guarantee. The main ideas of the proof are similar to that in Wolsey [22]; see Appendix D for the proof.

► **Theorem 5.** *Let  $c \in (0, 1]$ . For AGREEDY we have*

$$f(\text{AG}) \geq \frac{1-x}{2-(2-c)x} f(\text{OPT}),$$

where  $x$  is the unique root of  $\frac{1}{c} (1 - e^{-cz}) = \frac{1-z}{2-(2-c)z}$  for  $z \in [0, 1]$ .

The result of Theorem 5 coincides with the known robustness factors of MGREEDY for the cases  $c = 0$  and  $c = 1$ . For the limit  $c \rightarrow 0$  we have

$$\lim_{c \rightarrow 0} \frac{1}{c} (1 - e^{-cz}) = z,$$

and the equation simplifies to  $z = 1/2$  which is the known robustness factor for the modular case; see, e.g., the textbook by Korte and Vygen [11]. For the other extreme case of  $c = 1$  the equation simplifies to  $1 - e^{-z} = \frac{1-z}{2-z}$ , which was shown by Wolsey [22].

## 4 Submodular Knapsack Problem with Unknown Capacity

In this section we introduce an algorithm that generates a policy which is always at least as good as AGREEDY even though it does not know the capacity of the knapsack. For that purpose we introduce indispensable items in the first part of this section. They are defined similar to swap items defined by Disser et al. [5], which they used to achieve their  $1/2$ -optimal policy for the linear case.

As discussed by Kawase et al. [9], one major challenge when going from the case of a linear objective function to a submodular objective function is that the greedy order of items depends on the capacity of the knapsack. When an item is packed into the knapsack then other items that have a large overlap in terms of the objective with the packed item decrease in density. On the other hand, for another capacity where the first item is not packed since it does not fit they remain attractive. This issue makes it difficult to compare the outcome of a packing policy that does not know the capacity with the outcome of the MGREEDY algorithm as it was done in Disser et al. [5].

Kawase et al. [9] overcome this issue by introducing the concept of the *single-valuable items*  $i$  with the property  $f(\{i\}) \geq 2f(\text{OPT}(s(i)/2))$ , i.e., Kawase et al. do not compare items with the greedy solution at all and instead compare the value of an item directly with OPT. In their policy, the most valuable single-valuable item that fits in the knapsack is inserted first. Afterwards, they try to insert the rest of the items in their greedy order. This deterministic policy achieves a robustness factor of  $2(1 - 1/e)/21 \approx 0.06$ .

We use another idea to overcome the capacity-dependency of the greedy order. We define the concept of an *indispensable item*. These are items that the alternative greedy algorithm AGREEDY returns instead of the greedy solution. It turns out that the performance of this algorithm can be also obtained by a policy that does not know the capacity.

### 4.1 Indispensable Items

► **Definition 6.** An item  $i \in N$  is called *indispensable* if there exists a capacity  $C \in \mathbb{R}_{>0}$  for which AGREEDY returns  $\{i\} = \{i_{k+1}\}$  instead of the greedy solution  $G_k$ .

We say that an item  $i \in N$  is indispensable for capacity  $C \in \mathbb{R}_{\geq s(i)}$  if AGREEDY returns  $\{i\} = \{i_{k+1}\}$  instead of the greedy solution  $G_k$  for capacity  $C$ .

For ease of exposition, we assume in the following that there are no ties when an algorithm compares items by value, differences in value, or density. In practice this could be achieved by small perturbations of the values, or by using a lexicographic order that breaks ties in a systematic way. However, to avoid heavy notation, we assume that ties do not exist.

For a capacity  $C \in \mathbb{R}_{>0}$ , let  $G_C = (i_1, i_2, \dots, i_{n_C})$  be the greedy order of all items in  $N_C = \{i \in N \mid s(i) \leq C\}$  with  $n_C = |N_C|$ . The following lemma contains important properties of indispensable items that are key to our definition of robust policies.

► **Lemma 7.** *Let item  $i \in N$  be an indispensable item for capacity  $C \in \mathbb{R}_{\geq s(i)}$  and let  $G_C = (i_1, i_2, \dots, i_{n_C})$  be the greedy order for the given capacity with  $i = i_{k+1}$  for  $k \in \{0, \dots, n_C - 1\}$ . Then the following properties hold:*

- (i)  $k \geq 1$  and  $s_{k+1} > \sum_{j=1}^k s_j$ .
- (ii)  $i$  is and only is an indispensable item for capacities in the interval  $[C_1, C_2[$ , with

$$\begin{aligned} C_1 &= s_{k+1}, \\ C_2 &= \min \left\{ s(\{i_1, \dots, i_{k+1}\}), \right. \\ &\quad \left. \min \{ \tilde{C} > C \mid \text{the first } k+1 \text{ items in } G_{\tilde{C}} \text{ are not} \right. \\ &\quad \left. \text{the first } k+1 \text{ items of the greedy-order } G_{C_1} \} \right\}. \end{aligned}$$

- (iii) *Let  $\tilde{C}$  be the smallest capacity larger than  $s_{k+1}$ , such that the first  $k+1$  items in  $G_{\tilde{C}}$  are not the first  $k+1$  items in  $G_{s_{k+1}}$ . Then the first item in  $G_{\tilde{C}}$  that is larger than  $s_{k+1}$  is either the first item in  $G_{\tilde{C}}$  or an indispensable item for capacity  $\tilde{C}$ .*

**Proof.** In the following we denote the indispensable item  $i$  by  $i_{k+1}$ .

We start to show (i). Obviously,  $i_1$  cannot be an indispensable item for capacity  $C$  by definition. Therefore,  $1 \leq k \leq n_C - 1$ . Since  $i_{k+1}$  is an indispensable item, we have for all  $j \in \{1, 2, \dots, k\}$  that

$$f(G_k) < f(G_k \cup \{i_{k+1}\}) - f(G_k) \leq f(G_j \cup \{i_{k+1}\}) - f(G_j). \quad (4)$$

Additionally, we know for every greedy order that for all  $j \in \{1, 2, \dots, k\}$  it holds that

$$\frac{f(G_{j-1} \cup \{i_j\}) - f(G_{j-1})}{s_j} \geq \frac{f(G_{j-1} \cup \{i_{k+1}\}) - f(G_{j-1})}{s_{k+1}}. \quad (5)$$

Furthermore, we have

$$f(G_k) = \sum_{j=1}^k f(G_{j-1} \cup \{i_j\}) - f(G_{j-1}) \geq \sum_{j=1}^k \frac{s_j}{s_{k+1}} (f(G_{j-1} \cup \{i_{k+1}\}) - f(G_{j-1}))$$

where the first sum is telescopic and then we used inequality (5). We obtain

$$s_{k+1} \geq \sum_{j=1}^k s_j \frac{f(G_{j-1} \cup \{i_{k+1}\}) - f(G_{j-1})}{f(G_k)} > \sum_{j=1}^k s_j$$

by inequality (4).

We next show (ii). Let  $C$  be a capacity for which  $i_{k+1}$  is an indispensable item. Obviously,  $C \geq C_1 = s_{k+1}$ . By (i) we know that the size of every item  $i_j, j \in \{1, \dots, k\}$  is smaller than the size of  $i_{k+1}$ . Therefore, we have that the first  $k$  items of the greedy order  $G_C$  are identical to the first  $k$  items of the greedy order  $G_{\tilde{C}}$  for all capacities  $\tilde{C} \in [C_1, C]$  and thus,  $i_{k+1}$  is an indispensable item for all those capacities.

For all capacities  $\hat{C} > C$  we have that  $i_{k+1}$  is an indispensable item until there is a capacity for which the first  $k+1$  items of the greedy order change or  $i_{k+1}$  does no longer exceed the capacity, which is the case for  $s(i_1, \dots, i_{k+1})$ , if the greedy order does not change. Therefore,  $i_{k+1}$  is an indispensable item for all capacities  $\hat{C} \in [C, C_2[$ .

Assume that  $i_{k+1}$  is an indispensable item for a capacity higher or equal to  $C_2$  and therefore, it is the first item in the greedy order that exceeds the capacity. Then we have that the first  $k+1$  items of the greedy order have to have changed in comparison to  $G_C$ . But then, there is an item in front of  $i_{k+1}$  in the greedy order, which is larger than  $i_{k+1}$ . That contradicts property (i) for indispensable items.



■ **Algorithm 1** Determine indispensable items.

---

```

1: procedure ISINDISPENSABLE(N, i^*)
2: $G_0 \leftarrow \emptyset, j \leftarrow 1$
3: $U \leftarrow \{i \in N \mid s(i) \leq s(i^*)\}$
4: while $s(G_{j-1}) \leq s(i^*)$ do
5: $i_j \leftarrow \arg \max_{i \in U} \left\{ \frac{f(G_{j-1} \cup \{i\}) - f(G_{j-1})}{s(i)} \right\}$
6: if $i_j = i^*$ then
7: if $j \geq 2$ and $f(G_{j-1} \cup \{i_j\}) - f(G_{j-1}) > f(G_{j-1})$ then
8: return (True, G_{j-1})
9: else
10: return (False, \emptyset)
11: else
12: $G_j \leftarrow G_{j-1} \cup \{i_j\}$
13: $U \leftarrow U \setminus \{i_j\}$
14: $j \leftarrow j + 1$
15: return (False, \emptyset)

```

---

Finally, we show (iii). Let  $G_{\tilde{C}}$  be the greedy order for capacity  $\tilde{C}$  and let  $\tilde{G}_j$  denote the first  $j$  items in  $G_{\tilde{C}}$ . Further let  $\tilde{i}_{\tilde{k}+1}$  be the first item in  $G_{\tilde{C}}$  that has a larger size than  $i_{k+1}$ . It holds  $\tilde{k} \leq k$ ,  $G_j = \tilde{G}_j$  for all  $j \in \{1, \dots, \tilde{k}\}$  and  $s(\tilde{i}_{\tilde{k}+1}) = \tilde{C}$ . We proceed to prove that  $\tilde{i}_{\tilde{k}+1}$  is an indispensable item for  $\tilde{C}$ , if  $\tilde{k} \geq 1$ . Since  $s(\tilde{G}_{\tilde{k}}) = s(G_{\tilde{k}}) < s_{k+1} < s(\tilde{i}_{\tilde{k}+1})$  we have that  $\tilde{i}_{\tilde{k}}$  is the first item in the greedy order  $G_{\tilde{C}}$  that exceeds the capacity. Additionally, we have

$$\begin{aligned}
f(\tilde{G}_{\tilde{k}} \cup \{\tilde{i}_{\tilde{k}+1}\}) - f(\tilde{G}_{\tilde{k}}) &> f(\tilde{G}_{\tilde{k}} \cup \{i_{k+1}\}) - f(\tilde{G}_{\tilde{k}}) \\
&\geq f(G_k \cup \{i_{k+1}\}) - f(G_k) > f(G_k) \geq f(\tilde{G}_{\tilde{k}}).
\end{aligned}$$

The first inequality holds, since  $\tilde{i}_{\tilde{k}+1}$  is in front of  $i_{k+1}$  in the greedy order  $G_{\tilde{C}}$  and  $s(\tilde{i}_{\tilde{k}+1}) > s(i_{k+1})$ . Second and last inequality follow from submodularity and the fact that  $\tilde{G}_{\tilde{k}} = G_{\tilde{k}} \subseteq G_k$ . The third inequality holds, because  $i_{k+1}$  is an indispensable item. ◀

With Lemma 7 (ii) we can determine if an item  $i$  is an indispensable item by checking if it is indispensable for the capacity  $s(i)$ . Such a procedure is given in Algorithm 1. The algorithm builds the greedy order  $G_{s(i)}$  until the first item exceeds the capacity or item  $i$  is chosen. Only if  $i$  is the item that exceeds the capacity and fulfills the condition  $f(G_{j-1} \cup \{i\}) - f(G_{j-1}) > f(G_{j-1})$ , the algorithm returns **True** together with the greedy items in front of  $i$ , which will be helpful later on.

## 4.2 Robust Policy

The general idea of the adaptive policy is to choose a reasonable start item based on Lemma 7 (iii). We build a list of those items and then start to build our solution with the biggest item of the list that fits in the knapsack. Note that AGREEDY always returns the greedy solution for all capacities smaller than the size of the smallest indispensable item, thus we start the list with the smallest indispensable item and only add larger items to the list. By Lemma 7 (iii) we have for an indispensable item  $i_{k+1}$ , that it is part of the output of AGREEDY until there is a capacity for which there is a larger indispensable item or there is a new first item in the greedy order for this capacity. Therefore, we add exactly those items to the list.

■ **Algorithm 2** List of start items.

---

```

1: procedure STARTITEMLIST(N)
2: $i_1, \dots, i_{|N|} \leftarrow$ < items sorted non-decreasingly by size >
3: $F \leftarrow [], j \leftarrow 1$
4: while $j \leq |N|$ do
5: $U \leftarrow \{i \in N \mid s(i) \leq s(i_j)\}$
6: $i \leftarrow \arg \max \left\{ \frac{f(\{i\})}{s(i)} \mid i \in U \right\}$
7: $(isIndis, G) \leftarrow \text{ISINDISPENSABLE}(N, i)$
8: if $isIndis = \text{True}$ then
9: $F \leftarrow [i] + F$
10: else if $i = i_1$ and $F \neq []$ then
11: $F \leftarrow [i] + F$
12: $j \leftarrow j + 1$
13: return F

```

---

We create a list of start items as follows. It starts by ordering the items non-decreasingly by size. For every item  $i$  it is checked if the item is indispensable or if it is the first item in the greedy order for capacity  $s(i)$ . Indispensable items are always added to the list and items, which are first in the greedy order, are only added if there is already an item in the list. Thus, the smallest indispensable item is the first item added to the list. Note that it is not possible that two items of the same size are added to the list. Since the algorithm always adds an item to the start of the list, the size of items in the list is strictly decreasing. For a formal description; see Algorithm 2.

Finally, Algorithm 3 generates an adaptive policy for the submodular knapsack problem with unknown capacity. The algorithm consists of three main steps.

**Step 1.** If possible, insert the largest item from the list of starting items  $F$  that fits in the knapsack. Let this item be  $i_{k+1}$ .

**Step 2.** If  $i_{k+1}$  is an indispensable item, try to insert the first  $k$  items of the greedy order  $G_{s(i_{k+1})}$ .

**Step 3.** Try to pack all other items in their greedy order.

If AGREEDY returns an indispensable item, Step 1 guarantees that this indispensable item is also in the solution returned by Algorithm 3. For the case, when an indispensable item  $i_{k+1}$  is added to the solution in Step 1, but the capacity is higher or equal to  $s(\{i_1, \dots, i_{k+1}\})$ , such that  $i_{k+1}$  is not an indispensable for this capacity, we want to add the items  $\{i_1, \dots, i_k\}$ , contained in  $G_k$  to our solution (Step 2). In Step 3 we complete the solution. We try to add items to the knapsack in their greedy order. We will show in Theorem 8 that this policy generated by Algorithm 3 is always as good as AGREEDY.

► **Theorem 8.** *For a capacity  $C \in \mathbb{R}_{>0}$  let  $\Pi(C)$  be the policy generated from Algorithm 3 and let  $AG(C)$  be the output of AGREEDY. Then, we have  $f(\Pi(C)) \geq f(AG(C))$  for every capacity  $C \in \mathbb{R}_{>0}$ .*

**Proof.** Let  $F$  be the list of starting items created by Algorithm 2. If there are no indispensable items in  $N$ , then  $F$  and  $G_k$  are empty in Algorithm 3 and the algorithm tries in Step 3 to insert all items in their greedy order. Furthermore, AGREEDY always returns the greedy solution in this case. Assume that Algorithm 3 tries to pack an item in the knapsack that is not part of the greedy solution returned by AGREEDY, before the algorithm has packed all items of the greedy solution. Since this item was not considered by AGREEDY, it has to be larger than the knapsack capacity and thus, cannot be inserted by Algorithm 3. Therefore, we obtain  $f(\Pi(C)) \geq f(AG(C))$  for all capacities  $C \in \mathbb{R}_{>0}$  in this case.

---

**Algorithm 3** Robust policy.

---

```

1: $S \leftarrow \emptyset, G \leftarrow \emptyset$
2: $F \leftarrow \text{STARTITEMLIST}(N)$
3: $j \leftarrow 1$
4: while $S = \emptyset$ and $j \leq \text{len}(F)$ do
5: $i_{k+1} \leftarrow F[j]$
6: if i_{k+1} fits in the knapsack then
7: $(\sim, G_k) \leftarrow \text{ISINDISPENSABLE}(N, i_{k+1})$
8: $S = \{i_{k+1}\}$
9: else
10: $N \leftarrow \{i \in N \mid s(i) < s(i_{k+1})\}$
11: $j \leftarrow j + 1$
12: for $i \in G_k$ do
13: if $S \cup \{i\}$ fits in the knapsack then
14: $S \leftarrow S \cup \{i\}$
15: $N \leftarrow N \setminus \{i\}$
16: while $N \neq \emptyset$ do
17: $i_{\max} \leftarrow \arg \max \{ \frac{f(S \cup \{i\}) - f(S)}{s(i)} \mid i \in N \}$
18: if $S \cup \{i_{\max}\}$ fits into the knapsack then
19: $S \leftarrow S \cup \{i_{\max}\}$
20: $N \leftarrow N \setminus \{i_{\max}\}$
21: else
22: $N \leftarrow \{i \in N \mid s(i) < s(i_{\max})\}$
23: return S

```

---

Now we assume there is at least one indispensable item, implying that  $F$  is not empty. Let  $f_1, \dots, f_{|F|}$  be the items in  $F$  sorted increasingly by size. Consider the partition of all capacities in the intervals  $[I_j, I_{j+1})$  for  $j \in \{0, \dots, |F|\}$ , with  $I_0 = \min\{s(i) \mid i \in N\}$ ,  $I_{|F|+1} = s(N)$  and  $I_j = s(f_j)$  for  $j \in \{1, \dots, |F|\}$ .

First, we consider all capacities  $C \in [I_0, I_1)$ . Since, the capacities are smaller than any item in  $F$ , Algorithm 3 inserts no item from  $F$  and  $G_k$  remains empty. Therefore, Algorithm 3 again tries to insert all items by their greedy order and AGREEDY returns the greedy solution for all those capacities, because all indispensable items have a larger size. We have  $f(\Pi(C)) \geq f(\text{AG}(C))$  by the same argumentation as in the case where  $F$  is empty.

Second, we consider all capacities  $C \in [I_j, I_{j+1})$  for an arbitrary  $j \in \{1, \dots, |F|\}$ . Note that Algorithm 3 inserts the item  $f_j$  from  $F$  for all those capacities as the first item. We distinguish the cases where  $f_j$  is an indispensable item and where  $f_j$  is not an indispensable item. In the following we denote item  $f_j$  as  $i_{k+1}$ .

**Case 1:  $i_{k+1}$  is an indispensable item.** By Lemma 7 (ii) we have that  $i_{k+1}$  is packed by AGREEDY instead of the greedy solution  $G_k$  for all capacities  $C \in [C_1, C_2)$ . For all capacities  $C \in [C_1, C_2) \cap [I_j, I_{j+1})$  we have  $f(\Pi(C)) \geq f(\text{AG}(C))$ , since Algorithm 3 inserts the same indispensable item in step 1. Since  $C_1 = I_j$ , there are only two more cases to distinguish:  $C \in [C_2, I_{j+1})$  if  $C_2 < I_{j+1}$  and  $C \in [I_{j+1}, C_2)$  if  $C_2 > I_{j+1}$ .

**Subcase 1.1:**  $C \in [C_2, I_{j+1})$  if  $C_2 < I_{j+1}$ . Note that we have  $C_2 = s(\{i_1, \dots, i_{k+1}\})$ , because  $C_2 < s(i_1, \dots, i_{k+1})$  implies that an item with size  $C_2$  changed the greedy order of the first  $k+1$  items. Such an item would have been added to  $F$ , since it is an indispensable item or the first item in the greedy order by Lemma 7 (iii), but then  $C_2 = I_{j+1}$ .

For all capacities  $C \in [C_2, I_{j+1})$  we know by Lemma 7 (iii) that the first  $k+1$  items in the greedy order  $G_C$  are still identical to the first  $k+1$  items of  $G_{s_{k+1}}$ . We also know that there is no indispensable item for any capacity  $C \in [C_2, I_{j+1})$ . Therefore, AGREEDY returns the greedy solution. After Algorithm 3 packed the indispensable item  $i_{k+1}$ , the algorithm continues in step 2 to insert the first  $k$  greedy items of the greedy order  $G_{s(i_{k+1})}$ . We already discussed that all these items fit in the knapsack, since  $C \geq C_2 = s(\{i_1, \dots, i_{k+1}\})$  and that they are also the first  $k+1$  items packed by AGREEDY.

After the first  $k+1$  items, AGREEDY continues to pack more items greedily until the first item exceeds the capacity. Algorithm 3 also continues in step 3 to pack more items greedily and thus we have  $f(\Pi(C)) \geq f(\text{AG}(C))$  for all capacities  $C \in [C_2, I_{j+1})$ .

**Subcase 1.2:**  $C \in [I_{j+1}, C_2)$  if  $C_2 > I_{j+1}$ . We will show that this case cannot occur. Consider item  $f_{j+1} \in F$  with  $s(f_{j+1}) = I_{j+1}$ . It is either an indispensable item for capacity  $I_{j+1}$  or the first item in the greedy order  $G_{I_{j+1}}$ .

By Lemma 7 (ii) we know, that  $f_{j+1}$  is not allowed to change the first  $k+1$  items in the greedy order  $G_{s_{k+1}}$ , since then we would have  $C_2 = I_{j+1}$ . Thus,  $f_{j+1}$  cannot be the first item in the greedy order  $G_{I_{j+1}}$  and as an indispensable item for capacity  $I_{j+1}$  it has to appear behind the first  $k+1$  items of  $G_{s_{k+1}}$ . But this also gives a contradiction, since

$$s(f_{j+1}) > \sum_{j=1}^{k+1} s_j \geq C_2 > I_{j+1} = s(f_{j+1}). \quad (6)$$

The first inequality follows from Lemma 7 (i) and the second inequality from Lemma 7 (ii).

**Case 2:  $i_{k+1}$  is not an indispensable item** Then,  $i_{k+1}$  is the first item in the greedy order  $G_{s(i_{k+1})}$  and it is the first item in the greedy order  $G_C$  for every capacity  $C \in [I_j, I_{j+1})$  too, since a new first item would have been added to  $F$ . Additionally, we know that there is no indispensable item for all capacities  $C \in [I_j, I_{j+1})$ , since it also would have been added to  $F$ . Therefore, AGREEDY returns the greedy solution for all capacities  $C \in [I_j, I_{j+1})$  and the first item packed by Algorithm 3 is always the first item of the greedy solution returned by AGREEDY. In step 2 of Algorithm 3 no items are added to the knapsack, because  $G_k$  is empty, if  $i_{k+1}$  is not an indispensable item. Then, Algorithm 3 tries to insert items exactly by their greedy order in step 3 and this yields  $f(\Pi(C)) \geq f(\text{AG}(C))$  for all capacities  $C \in [I_j, I_{j+1})$ .

Thus, the statement holds for all capacities in the intervals  $[I_j, I_{j+1})$ ,  $j \in \{0, \dots, |F|\}$ . For capacities smaller than  $I_0$  no item can be packed and for capacities greater than  $I_{|F|+1}$  every item can be packed in the knapsack. This completes the proof.  $\blacktriangleleft$

From Theorem 5 and Theorem 8 we obtain the main result of this paper.

► **Theorem 9.** *The policy  $\Pi$  generated by Algorithm 3 has a robustness factor of  $\alpha = \frac{1-x}{2-(2-c)x}$  where  $x$  is the unique root in  $[0, 1]$  of the equation  $\frac{1}{c}(1 - e^{-cz}) = \frac{1-z}{2-(2-c)z}$ .*

## References

- 1 Aaron Bernstein, Yann Disser, Martin Groß, and Sandra Himburg. General bounds for incremental maximization. *Math. Program.*, 2020. doi:10.1007/s10107-020-01576-0.
- 2 Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. doi:10.1137/080733991.
- 3 Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discret. Appl. Math.*, 7(3):251–274, 1984. doi:10.1016/0166-218X(84)90003-9.
- 4 Gerard Cornuéjols, Marshall L. Fisher, and George L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Sci.*, 23:789–810, 1977. doi:10.1287/mnsc.23.8.789.
- 5 Yann Disser, Max Klimm, Nicole Megow, and Sebastian Stiller. Packing a knapsack of unknown capacity. *SIAM J. Discret. Math.*, 31(3):1477–1497, 2017. doi:10.1137/16M1070049.
- 6 Yann Disser, Max Klimm, and David Weckbecker. Fractionally subadditive maximization under an incremental knapsack constraint. In Jochen Könemann and Britta Peis, editors, *Approximation and Online Algorithms - 19th International Workshop (WAOA)*, volume 12982 of *Lecture Notes in Computer Science*, pages 206–223. Springer, 2021. doi:10.1007/978-3-030-92702-8\_13.
- 7 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.
- 8 Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. *An analysis of approximations for maximizing submodular set functions - II*, volume 8 of *Mathematical Programming Studies*, page 73–87. Springer, Berlin, Heidelberg, 1978. doi:10.1007/BFb0121195.
- 9 Yasushi Kawase, Hanna Sumita, and Takuro Fukunaga. Submodular maximization with uncertain knapsack capacity. *SIAM J. Discret. Math.*, 33(3):1121–1145, 2019. doi:10.1137/18M1174428.
- 10 David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory Comput.*, 11:105–147, 2015. doi:10.4086/toc.2015.v011a004.
- 11 Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer, Heidelberg, Berlin, 2018.
- 12 Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Trans. Intell. Syst. Technol.*, 2(4):32:1–32:20, 2011. doi:10.1145/1989734.1989736.
- 13 Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, 2008. URL: <https://dl.acm.org/citation.cfm?id=1390689>.
- 14 Jon Lee. Constrained maximum-entropy sampling. *Oper. Res.*, 46(5):655–664, 1998. doi:10.1287/opre.46.5.655.
- 15 Nicole Megow and Julián Mestre. Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science (ITCS)*, pages 495–504. ACM, 2013. doi:10.1145/2422436.2422490.
- 16 Alfredo Navarra and Cristina M. Pinotti. Online knapsack of unknown capacity: How to optimize energy consumption in smartphones. *Theor. Comput. Sci.*, 697:98–109, 2017. doi:10.1016/j.tcs.2017.07.029.
- 17 George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. doi:10.1287/moor.3.3.177.
- 18 George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978. doi:10.1007/BF01588971.

- 19 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004. doi:10.1016/S0167-6377(03)00062-2.
- 20 Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Math. Oper. Res.*, 42(4):1197–1218, 2017. doi:10.1287/moor.2016.0842.
- 21 Jan Vondrák. Submodularity and curvature: The optimal algorithm. *RIMS Kôkyûroku Bessatsu*, B23:253–266, 2010.
- 22 Laurence A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Math. Oper. Res.*, 7(3):410–425, 1982. doi:10.1287/moor.7.3.410.
- 23 Yuichi Yoshida. Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint. *SIAM J. Discret. Math.*, 33(3):1452–1471, 2019. doi:10.1137/16M1107644.

## A

 Proof of Lemma 1

**Proof.** We first show (i). Let  $A \subset N$  and  $j \in N \setminus A$  be arbitrary. We calculate

$$1 - c = \min_{i \in N} \frac{f(N) - f(N \setminus \{i\})}{f(\{i\})} \leq \frac{f(N) - f(N \setminus \{j\})}{f(\{j\})} \leq \frac{f(A \cup \{j\}) - f(A)}{f(\{j\})}$$

where for the equation, we used the definition of curvature, and for the last equation, we used submodularity.

To show (ii), we successively apply (i) on the elements in  $B$ . ◀

## B

 Proof of Lemma 3

**Proof.** Using Lemma 1 (ii) with  $A = \text{OPT}$  and  $B = G_{j-1} \setminus \text{OPT}$ , we obtain

$$f(\text{OPT}) + (1 - c) \sum_{i \in G_{j-1} \setminus \text{OPT}} f(\{i\}) - f(G_{j-1}) \leq f(\text{OPT} \cup G_{j-1}) - f(G_{j-1}). \quad (7)$$

We proceed to bound the term  $f(\text{OPT} \cup G_{j-1}) - f(G_{j-1})$  that appears on the right hand side of (7) via

$$\begin{aligned} f(\text{OPT} \cup G_{j-1}) - f(G_{j-1}) &\leq \sum_{i \in \text{OPT} \setminus G_{j-1}} f(G_{j-1} \cup \{i\}) - f(G_{j-1}) \\ &= \sum_{i \in \text{OPT} \setminus G_{j-1}} s(i) \frac{f(G_{j-1} \cup \{i\}) - f(G_{j-1})}{s(i)} \\ &\leq \left( \sum_{i \in \text{OPT} \setminus G_{j-1}} s(i) \right) \frac{f(G_{j-1} \cup \{i_j\}) - f(G_{j-1})}{s_j} \\ &= s(\text{OPT} \setminus G_{j-1}) \frac{f(G_j) - f(G_{j-1})}{s_j} = s(\text{OPT} \setminus G_{j-1}) \frac{\delta_j}{s_j}, \end{aligned}$$

## 49:16 Maximizing a Submodular Function Under an Unknown Knapsack Constraint

where we used (3) and the definition of the greedy sequence. To show (i), we bound the term  $(1-c) \sum_{i \in G_{j-1} \setminus \text{OPT}} f(\{i\}) - f(G_{j-1})$  on the left hand side of (7) by

$$\begin{aligned} (1-c) \sum_{i \in G_{j-1} \setminus \text{OPT}} f(\{i\}) - f(G_{j-1}) &\geq (1-c) \sum_{m=1}^{j-1} (1-\chi_m) \delta_m - f(G_{j-1}) \\ &= (1-c) \sum_{m=1}^{j-1} (1-\chi_m) \delta_m - \sum_{m=1}^{j-1} \delta_m \\ &= -(1-c) \sum_{m=1}^{j-1} \chi_m \delta_m - c \sum_{m=1}^{j-1} \delta_m, \end{aligned}$$

where we used submodularity for the inequality. Both bounds applied to inequality (7) yield

$$\begin{aligned} s(\text{OPT} \setminus G_{j-1}) \frac{\delta_j}{s_j} &\geq f(\text{OPT}) - (1-c) \sum_{m=1}^{j-1} \chi_m \delta_m - c \sum_{m=1}^{j-1} \delta_m \\ &= c \left( f(\text{OPT}) - \sum_{m=1}^{j-1} \delta_m \right) + (1-c) \left( f(\text{OPT}) - \sum_{m=1}^{j-1} \chi_m \delta_m \right). \end{aligned}$$

Multiplication with  $\frac{s_j}{s(\text{OPT} \setminus G_{j-1})}$  and applying  $s(\text{OPT} \setminus G_{j-1}) \leq C - S_{j-1} \leq C$  completes the proof of (i).

To prove (ii), we bound  $\sum_{i \in G_{j-1} \setminus \text{OPT}} f(\{i\})$  in a different way by

$$\begin{aligned} \sum_{i \in G_{j-1} \setminus \text{OPT}} f(\{i\}) &\geq \sum_{m=1}^{j-1} (1-\chi_m) \delta_m \\ &= \sum_{m=1}^{j-1} (1-\chi_m) s_m \frac{\delta_m}{s_m} \\ &\geq \left( \sum_{m=1}^{j-1} (1-\chi_m) s_m \right) \frac{\delta_j}{s_j} = s(G_{j-1} \setminus \text{OPT}) \frac{\delta_j}{s_j}, \end{aligned}$$

where for the first inequality we used submodularity and for the second inequality we used a property of the greedy sequence. Applied to inequality (7) together with the first bound yields

$$\begin{aligned} f(\text{OPT}) - f(G_{j-1}) &\leq s(\text{OPT} \setminus G_{j-1}) \frac{\delta_j}{s_j} - (1-c) s(G_{j-1} \setminus \text{OPT}) \frac{\delta_j}{s_j} \\ &= \left( s(\text{OPT} \setminus G_{j-1}) - (1-c) s(G_{j-1} \setminus \text{OPT}) \right) \frac{\delta_j}{s_j} \\ &\leq \left( s(\text{OPT}) - (1-c) s(G_{j-1}) \right) \frac{\delta_j}{s_j} \\ &\leq \left( C - (1-c) s(G_{j-1}) \right) \frac{\delta_j}{s_j}, \end{aligned}$$

as claimed. ◀



## C

 Proof of Theorem 4

**Proof.** Applying the bound of Lemma 3 (i) for all  $j$ , we obtain the following lower bound on  $G_j$ .

▷ **Claim 10.** For all  $j \in \{1, \dots, k+1\}$  and for all  $l \in \{0, \dots, j\}$  we have

$$\begin{aligned} f(G_j) &\geq f(\text{OPT}) - \left( \prod_{m=l+1}^j \left( 1 - \frac{cs_m}{C} \right) \right) \left( f(\text{OPT}) - \sum_{m=1}^l \delta_m \right) \\ &\quad + \frac{1-c}{c} \frac{C}{C-S_l} \left( 1 - \prod_{m=l+1}^j \left( 1 - \frac{cs_m}{C} \right) \right) \left( f(\text{OPT}) - \sum_{m=1}^l \chi_m \delta_m \right). \end{aligned} \quad (8)$$

For a fixed  $j \in \{1, \dots, k+1\}$  we proof the claim by induction over  $l$  starting with  $l = j$  and going down by one in the induction step.

For  $l = j$  both products in (8) are equal to 1 and the right side simplifies to  $\sum_{m=1}^j \delta_m = f(G_j)$ . For the induction step assume that the statement of the claim holds for  $l \in \{1, \dots, j\}$  and consider the statement for  $l-1$ . To shorten the notation we set

$$P_t := \prod_{m=t}^j \left( 1 - \frac{cs_m}{C} \right),$$

for  $t = 1, \dots, j+1$ . The induction hypothesis is

$$\begin{aligned} f(G_j) &\geq f(\text{OPT}) - P_{l+1} \left( f(\text{OPT}) - \sum_{m=1}^l \delta_m \right) \\ &\quad + \frac{1-c}{c} \frac{C}{C-S_l} \left( 1 - P_{l+1} \right) \left( f(\text{OPT}) - \sum_{m=1}^l \chi_m \delta_m \right). \end{aligned}$$

To apply Lemma 3 (i) for  $j = l$ , we rearrange  $\delta_l$  terms. Those terms read

$$\left( P_{l+1} - \chi_l \frac{1-c}{c} \frac{C}{C-S_l} \left( 1 - P_{l+1} \right) \right) \delta_l.$$

We can transform the factor of  $\delta_l$  to see that it is greater or equal to zero. Note that the factor  $1/c$  vanishes in  $1 - P_{l+1}$ .

$$\begin{aligned} &P_{l+1} - \chi_l \frac{1-c}{c} \frac{C}{C-S_l} \left( 1 - P_{l+1} \right) \\ &= (1-c) + c \left( 1 - \frac{1}{c} \left( 1 - P_{l+1} \right) \right) - \chi_l (1-c) \frac{C}{C-S_l} \frac{1}{c} \left( 1 - P_{l+1} \right) \\ &= c \left( 1 - \frac{1}{c} \left( 1 - P_{l+1} \right) \right) + (1-c) \left( 1 - \chi_l \frac{C}{C-S_l} \frac{1}{c} \left( 1 - P_{l+1} \right) \right) \geq 0. \end{aligned}$$

Applying

$$\delta_l \geq \frac{cs_l}{C} \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \delta_m \right) + \frac{(1-c)s_l}{C-S_{l-1}} \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \chi_m \delta_m \right),$$

yields

$$f(G_j) \geq f(\text{OPT}) - \alpha \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \delta_m \right) + \beta \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \chi_m \delta_m \right),$$

with

$$\begin{aligned}\alpha &= P_{l+1} - \frac{cs_l}{C} \left( P_{l+1} - \chi_l \frac{1-c}{c} \frac{C}{C-S_l} (1-P_{l+1}) \right) \\ &= \left( 1 - \frac{cs_l}{C} \right) P_{l+1} + \chi_l (1-c) \frac{s_l}{C-S_l} (1-P_{l+1}) = P_l + \chi_l (1-c) \frac{s_l}{C-S_l} (1-P_{l+1}),\end{aligned}$$

and

$$\begin{aligned}\beta &= \frac{1-c}{c} \frac{C}{C-S_l} (1-P_{l+1}) + \frac{(1-c)s_l}{C-S_{l-1}} \left( P_{l+1} - \chi_l \frac{1-c}{c} \frac{C}{C-S_l} (1-P_{l+1}) \right) \\ &= \frac{1-c}{c} \frac{C}{C-S_l} \left( 1 - \chi_l \frac{(1-c)s_l}{C-S_{l-1}} \right) (1-P_{l+1}) + \frac{(1-c)s_l}{C-S_{l-1}} P_{l+1} \\ &= \frac{1-c}{c} \frac{C}{C-S_l} \left( \frac{C-S_{l-1}-\chi_l s_l}{C-S_{l-1}} + \chi_l \frac{cs_l}{C-S_{l-1}} \right) (1-P_{l+1}) + \frac{(1-c)s_l}{C-S_{l-1}} P_{l+1} \\ &= \frac{1-c}{c} \frac{C}{C-S_{l-1}} (1-P_{l+1}) + \chi_l \frac{1-c}{c} \frac{C}{C-S_l} \frac{cs_l}{C-S_{l-1}} (1-P_{l+1}) + \frac{(1-c)s_l}{C-S_{l-1}} P_{l+1} \\ &= \frac{1-c}{c} \frac{C}{C-S_{l-1}} \left( 1 - (1 - \frac{cs_l}{C}) P_{l+1} \right) + \chi_l (1-c) \frac{C}{C-S_l} \frac{s_l}{C-S_{l-1}} (1-P_{l+1}) \\ &= \frac{1-c}{c} \frac{C}{C-S_{l-1}} (1-P_l) + \chi_l (1-c) \frac{C}{C-S_l} \frac{s_l}{C-S_{l-1}} (1-P_{l+1}) \\ &\geq \frac{1-c}{c} \frac{C}{C-S_{l-1}} (1-P_l) + \chi_l (1-c) \frac{s_l}{C-S_l} (1-P_{l+1}).\end{aligned}$$

Thus, we have

$$\begin{aligned}f(G_j) &\geq f(\text{OPT}) - P_l \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \delta_m \right) \\ &\quad + \frac{1-c}{c} \frac{C}{C-S_{l-1}} (1-P_l) \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \chi_m \delta_m \right) \\ &\quad + \chi_l (1-c) \frac{s_l}{C-S_l} (1-P_{l+1}) \left( \sum_{m=1}^{l-1} \delta_m - \sum_{m=1}^{l-1} \chi_m \delta_m \right) \\ &\geq f(\text{OPT}) - P_l \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \delta_m \right) \\ &\quad + \frac{1-c}{c} \frac{C}{C-S_{l-1}} (1-P_l) \left( f(\text{OPT}) - \sum_{m=1}^{l-1} \chi_m \delta_m \right),\end{aligned}$$

as claimed.

For  $j \in \{1, \dots, k+1\}$  and  $l = 0$  the statement of the claim simplifies to

$$\begin{aligned}f(G_j) &\geq f(\text{OPT}) - \left( \prod_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right) f(\text{OPT}) + \frac{1-c}{c} \left( 1 - \prod_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right) f(\text{OPT}) \\ &= \left( 1 + \frac{1-c}{c} \right) \left( 1 - \prod_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right) f(\text{OPT}) \\ &= \frac{1}{c} \left( 1 - \prod_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right) f(\text{OPT}).\end{aligned}$$

We derive the final statement of the theorem

$$\begin{aligned}
 f(G_j) &\geq \frac{1}{c} \left( 1 - \prod_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right) f(\text{OPT}) \\
 &\geq \frac{1}{c} \left( 1 - \left( \frac{1}{j} \sum_{m=1}^j \left( 1 - \frac{cs_m}{C} \right) \right)^j \right) f(\text{OPT}) \\
 &= \frac{1}{c} \left( 1 - \left( 1 - \frac{c}{jC} \sum_{m=1}^j s_m \right)^j \right) f(\text{OPT}) \\
 &= \frac{1}{c} \left( 1 - \left( 1 - \frac{cs(G_j)}{jC} \right)^j \right) f(\text{OPT}) \\
 &\geq \frac{1}{c} \left( 1 - \exp \left( -c \frac{s(G_j)}{C} \right) \right) f(\text{OPT}),
 \end{aligned}$$

where we used that the geometric mean is always smaller or equal to the arithmetic mean for non-negative values and that  $(1 + \frac{x}{j})^j \leq e^x$  for all  $j \geq 1$  and  $x \in \mathbb{R}$ . ◀

## D Proof of Theorem 5

**Proof.** We define  $z := \frac{s(G_k)}{C}$ . By Theorem 4 we have for  $j = k$  that

$$f(G_k) \geq \frac{1}{c} (1 - e^{-cz}) f(\text{OPT}). \quad (9)$$

Additionally, we have by Lemma 3 (ii) for  $j = k + 1$  that

$$\delta_{k+1} \geq \frac{s_{k+1}}{C - (1-c)s(G_k)} (f(\text{OPT}) - f(G_k)).$$

Solving for  $f(\text{OPT})$  and replacing  $s(G_k)$  by  $zC$  gives

$$f(\text{OPT}) \leq f(G_k) + \frac{C(1 - (1-c)z)}{s_{k+1}} \delta_{k+1}.$$

With  $C < s(G_k) + s_{k+1} = zC + s_{k+1}$  we have  $C < \frac{s_{k+1}}{1-z}$  and this yields

$$\begin{aligned}
 f(\text{OPT}) &\leq f(G_k) + \frac{1 - (1-c)z}{1-z} \delta_{k+1} \\
 &\leq f(\text{AG}) + \frac{1 - (1-c)z}{1-z} f(\text{AG}) = \frac{2 - (2-c)z}{1-z} f(\text{AG}),
 \end{aligned}$$

where the last inequality holds in both cases of AGREEDY. Together with (9) we get

$$f(\text{AG}) \geq \min_{z \in [0,1]} \max \left\{ \frac{1}{c} (1 - e^{-cz}), \frac{1-z}{2 - (2-c)z} \right\} f(\text{OPT}).$$

Note that  $\frac{1}{c} (1 - e^{-cz})$  is monotonically increasing and  $\frac{1-z}{2 - (2-c)z}$  is monotonically decreasing for  $z \in [0,1]$  and a fixed  $c \in (0,1)$  and that they have a unique intersection for  $z \in [0,1]$ . Therefore, the minimum is attained at this intersection. ◀



# Some Results on Approximability of Minimum Sum Vertex Cover

Aleksa Stanković   

Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

---

## Abstract

We study the Minimum Sum Vertex Cover problem, which asks for an ordering of vertices in a graph that minimizes the total cover time of edges. In particular,  $n$  vertices of the graph are visited according to an ordering, and for each edge this induces the first time it is covered. The goal of the problem is to find the ordering which minimizes the sum of the cover times over all edges in the graph.

In this work we give the first explicit hardness of approximation result for Minimum Sum Vertex Cover. In particular, assuming the Unique Games Conjecture, we show that the Minimum Sum Vertex Cover problem cannot be approximated within 1.014. The best approximation ratio for Minimum Sum Vertex Cover as of now is  $16/9$ , due to a recent work of Bansal, Batra, Farhadi, and Tetali.

We also revisit an approximation algorithm for regular graphs outlined in the work of Feige, Lovász, and Tetali, and show that Minimum Sum Vertex Cover can be approximated within 1.225 on regular graphs.

**2012 ACM Subject Classification** Theory of computation → Problems, reductions and completeness; Mathematics of computing → Approximation algorithms; Theory of computation → Approximation algorithms analysis

**Keywords and phrases** Hardness of approximation, approximability, approximation algorithms, Label Cover, Unique Games Conjecture, Vertex Cover

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.50

**Category** APPROX

**Funding** Research supported by the Approximability and Proof Complexity project funded by the Knut and Alice Wallenberg Foundation.

**Acknowledgements** Author thanks Johan Håstad for fruitful discussion, as well as Hans Oude Groeniger and anonymous reviewers for useful comments which improved the presentation of this work.

## 1 Introduction

In the Minimum Sum Vertex Cover problem, as an input we are given a graph  $G = (V, E)$ , and the goal is to find an ordering of vertices which minimizes the total cover time of edges in  $E$ . In particular, we visit vertices in  $|V|$  steps, one at each step, and an edge  $e$  is considered to be covered at the time  $t \in \{1, \dots, |V|\}$  if the first time one of its endpoints is visited by the ordering is  $t$ .

The Minimum Sum Vertex Cover (MSVC) problem was introduced by Feige, Lovász, and Tetali [9], as a special case of the Minimum Sum Set Cover problem, which was of primary interest in that work. The same work showed that MSVC can be approximated within a factor of 2 using linear programming. That work also studied MSVC on regular graphs, and observed that a greedy algorithm approximates the optimal value within a factor of  $4/3$ . In addition to this, it was shown that  $4/3$  factor can be improved using semidefinite programming to some non-explicit constant  $\beta$  smaller than  $4/3$ .



© Aleksa Stanković;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 50; pp. 50:1–50:16



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The 2-approximation algorithm was subsequently improved by Barenholz, Feige, and Peleg [7], who gave a 1.999946-approximation algorithm for this problem. This was then substantially improved by Bansal, Batra, Farhadi, and Tetali, who, using linear programming with fairly involved rounding procedure, showed that MSVC can be approximated within a factor of  $16/9$ . Furthermore, the same work gives a linear programming integrality gap matching the approximation ratio.

So far explicit hardness of approximation results for this problem have been lacking, and to the best knowledge of the author, the only inapproximability result [9] gives hardness of  $1 + \varepsilon$ , for some small non-explicit  $\varepsilon > 0$ , using a reduction from the Minimum Vertex Cover problem on bounded degree graphs [1, 4]. In this work we give the first explicit hardness for MSVC, which we state in the following theorem.

► **Theorem 1.** *Assuming the Unique Games Conjecture, Minimum Sum Vertex Cover is NP-hard to approximate within 1.014.*

We use the Unique Games Conjecture introduced by Khot [10] as our hardness assumption. This conjecture has been the central open problem in the hardness of approximation area since its introduction, and many already known (and optimal) hardness of approximation results rely on the validity of this conjecture [15, 2, 13, 6].

Furthermore, our hardness reduction outputs regular graphs, for which better approximation algorithms are known compared to the general case.

Further to this, we will also revisit the approximation algorithm of Feige, Lovász, and Tetali [9] for regular graphs. Our contribution can be described as follows. The algorithm for regular graphs outlined in [9] uses an approximation algorithm for a problem called Max- $k$ -VC in a “black box” manner. Max- $k$ -VC problem is the problem of finding  $k$  vertices in a graph that cover as many edges as possible. The approximation ratio of the algorithm for regular graphs in [9] depends on the approximation ratio  $\alpha$  for Max- $k$ -VC problem. Due to the developments since the publication of [9] on Max- $k$ -VC, a better value of  $\alpha$  can be achieved, and hence by using this value we can obtain stronger approximation. Furthermore, a certain bound<sup>1</sup> used in an argument outlined in [9] for the approximation algorithm on regular graphs is incorrect, which we show by giving a counterexample in the appendix. We correct this by proving the optimal bound, and observe that the rest of the argument still holds. Let us remark that the sharpness of the bound affects the approximation ratio, and hence finding the optimal bound is desirable in this case. In conclusion, we obtain the following result

► **Theorem 2.** *Minimum Sum Vertex Cover can be approximated within 1.225 on regular graphs.*

## 1.1 Techniques and Proof Ideas

In this section we give an overview of the proof and briefly discuss techniques used.

The starting point of our reduction are Unique Games, which we formally describe in Section 2. More precisely, we use regular Affine Unique Games as an input to our reduction. Regular Affine Unique Games are Unique Games in which the alphabet is understood as an additive group  $\mathbb{Z}_L$ , and the constraints are of form  $x_u - x_v = c_e$  for an edge  $e = (u, v)$ , while the word regular indicates that the constraint graph is regular. Interestingly, in this

---

<sup>1</sup> We do not discuss what this bound exactly is here, for the sake of clarity.

work the structure of Affine Unique Games actually helps us achieve better completeness and therefore a stronger inapproximability result. The property of Affine Unique Games that we use can be described as follows. Let us consider the completeness case, in which we have some assignment  $z$  of labels to the vertices in the Affine Unique Games, which satisfies almost all the constraints. Then, for any  $a \in \mathbb{Z}_L$ , the assignment  $z_a = z + a$  gives another assignment which satisfies almost all the constraints. Furthermore, if we let  $V_a, a \in \mathbb{Z}_L$ , to be the vertex subset in the label extended graph comprised of vertex labels “selected” by the map  $z_a$ , then the sets  $V_a$  are disjoint, and this gives us enough structure to find an ordering with a low sum set cover value.

Let us elaborate. Our reduction uses the same standard long code dictatorship testing as the celebrated paper of Khot, Kindler, Mossel, and O’Donnell [11], which among other results gave the optimal hardness of Max-Cut assuming the Unique Games Conjecture. This is the same reduction that appeared in [4, 5], and hence the graphs that are output by the reduction satisfy the same properties as outlined in these works, which turns out to be useful for studying soundness. In particular, in the soundness case, for each  $r \in (0, 1)$ , and each vertex subset of fractional size  $r$ , we have a lower bound  $b := b(r)$  on the number of edges with both endpoints in this subset. Therefore, no matter which order of visiting the vertices we choose, after  $t \in \{1, \dots, n\}$  steps, we have not covered edges which have both endpoints in vertices visited after the time  $t$ , and hence at the time  $t$  we have at least  $b(1 - t/n)$  uncovered edges. This gives us a lower bound of form  $\sum_{i=1}^n b(1 - i/n) \approx \int_0^1 b(x)dx$ .

In the completeness case, we are supposed to specify an ordering of the vertices in each of  $k \in \mathbb{N}$  long codes. Given this ordering, in the first pass we would pick first vertex in each of the  $k$  long codes, after which we would pick the second vertex in each long code, etc. The order in which we visit  $k$  long codes will not be impactful. Hence, it is very important to pick order of visiting vertices in each long code well. This is where the affine structure of Unique Games proves to be useful. In the case we have only one good labeling (as it is the case with “classical” Unique Games), an obvious observation is that we can take first all vertices with 0 in the coordinate fixed by a good labelling  $z = z_0$ , and then all vertices with 1 in the same coordinate. However, there are many vertices in a long code which have 0 in the coordinate fixed by a good labelling, and hence many orderings can be chosen. Therefore, the question is which order should one pick the vertices with in this subset? Since in Affine Unique Games we have a second satisfying assignment, namely  $z_1$ , there is a natural ordering among these. We iterate through vertices that have 0 in the coordinate fixed by  $z_1$ , and after visiting the whole subgraph, visit vertices that have 1 in the coordinate fixed by  $z_1$ . We can repeat this idea and visit smaller and smaller subgraphs, the last of which will consist only of two vertices and for which we will use  $z_{L-1}$ .

The idea of using multiple good assignments in reductions from Unique Games already appeared in [8], but it is still fairly uncommon. Hence, it would be interesting to see whether it would be useful for some other problems as well.

As we mentioned in the introduction, the output of the hardness reduction is a weighted graph, and we need to remove its weights. The idea for this is simple: we replace each vertex  $v$  with  $m$  new vertices which we group in a set  $A_v$ , for  $m$  is sufficiently large. We then replace each edge  $e = (u, v)$  by sampling edges between  $A_u$  and  $A_v$  at a correct density. This graph indeed looks like the initial graph and is almost regular, however, proving that it preserves soundness and completeness properties, and making it exactly regular, requires some effort. Due to the size limitation, we defer the details of this part to the full version of this paper.



## 1.2 Organization

In Section 2 we introduce the notation used in this work, recall some well known facts, and formally introduce the Minimum Sum Vertex Cover problem. Then, in Section 3, we give our hardness reduction which outputs weighted graphs and discuss how it can be used to show Theorem 1.

Then, in Section 4 we show how Minimum Sum Vertex Cover on regular graphs can be approximated within a factor of 1.225, by recalling the algorithm from [9] and making necessary changes.

## 2 Preliminaries

For  $n \in \mathbb{N}$  we use  $[n]$  to denote  $[n] = \{1, 2, \dots, n\}$ . In this paper we work with undirected (multi)graphs  $G = (V, E)$ . For a set  $S \subseteq V$  of vertices we use  $S^c$  to denote its complement  $S^c = V \setminus S$ , and write  $U \sqcup V$  for a disjoint union of sets  $U$  and  $V$ .

The initial graph output by our reduction will be edge weighted. The weights of edges are given by a function  $w: E \rightarrow [0, 1]$ . For a subset  $K \subseteq E$  we interpret  $w(K)$  as the sum of weights of edges in  $K$ . Furthermore, we will typically *normalize* the weights so that  $w(E) = 1$ . For  $S, T \subseteq V$ , we write  $w(S, T)$  for the total weight of edges from  $E$  which have one endpoint in  $S$ , and other in  $T$ . Note that, since we work with undirected graphs, the order of endpoints is not important, and therefore  $w(S, T) = w(T, S)$ . We remark that the sets  $S, T$ , do not need to be disjoint. We also use  $N(S, T)$  to denote the set of all edges with one endpoint in  $S$  and other endpoint in  $T$ . For a vertex  $v \in V$ , we use  $N(v)$  to denote the set of its neighbours.

The following definition will be useful in discussing properties of our reduction.

► **Definition 3.** A graph  $G$  is  $(r, h)$ -dense if every subset  $S \subseteq V$  with  $w(S) = r$  satisfies  $w(S, S) \geq h$ .

Minimum Sum Vertex Cover is arguably more natural in an unweighted setting, i.e., the setting in which the weights of all edges are equal. Let us now introduce the Min Sum Vertex Cover problem for unweighted graphs.

► **Definition 4.** Consider an unweighted graph  $G = (V, E)$ , and let  $n = |V|$ . For an ordering of vertices represented as a bijection  $\sigma: [n] \leftrightarrow V$ , and an edge  $(u, v) = e \in E$ , let us denote with  $c_{\sigma, e}$  the “time” at which edge  $e$  is covered, that is

$$c_{\sigma, e} = \min(\sigma^{-1}(u), \sigma^{-1}(v)).$$

Then the Sum Vertex Cover under scheduling  $\sigma$ , which we denote by  $\text{SVC}_G(\sigma)$ , is given as

$$\text{SVC}_G(\sigma) = \frac{1}{|E|} \sum_{e \in E} c_{\sigma, e}. \quad (1)$$

The value of Min Sum Vertex Cover is the minimal value of  $\text{SVC}_G(\sigma)$  over all possible permutations  $\sigma$ , that is

$$\text{MSVC}(G) = \min_{\sigma: [n] \leftrightarrow V} \text{SVC}_G(\sigma). \quad (2)$$

We have normalized the expression for  $\text{SVC}_G(\sigma)$  by  $\frac{1}{|E|}$  for the sake of writing convenience. This does not affect our results, since the normalization factor will be cancelled out when studying approximation ratios. We can also reformulate the expression (1), stating the value

of Sum Vertex Cover under scheduling  $\sigma$ , as follows. At the time  $t \in [n]$ , the total number of edges not covered<sup>2</sup> is  $w(\sigma([t])^c, \sigma([t])^c)$ , and let us assign them the cost of 1 at that time. The cost  $c_{\sigma,e}$  of an edge  $e$  under  $\sigma$  is exactly the number of times  $t$  the edge was not covered, and hence we can write

$$\text{SVC}_G(\sigma) = \frac{1}{|E|} \sum_{t=1}^n w(\sigma([t])^c, \sigma([t])^c). \quad (3)$$

We remark that this allows us to define Minimum Sum Vertex Cover for edge weighted graphs, by replacing  $\frac{1}{|E|}$  above with  $\frac{1}{w(E)}$ , i.e., for weighted graphs we have

$$\text{SVC}_G(\sigma) = \frac{1}{w(E)} \sum_{t=1}^n w(\sigma([t])^c, \sigma([t])^c).$$

We can also discuss Minimum Sum Vertex Cover for weighted graphs in the sense of definitions (1) and (2) by letting

$$\text{SVC}_G(\sigma) = \frac{1}{w(E)} \sum_{e \in E} w_e c_{\sigma,e}.$$

As mentioned in the introduction, we can extend this definition in a natural way to include vertex weights. However, we have not found vertex weights to be useful for hardness reduction, and hence we omit further discussing this for the sake of simplicity.

In order to state the quantities appearing in our result, it is necessary to introduce some more notation. We use  $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$  to denote the density function of a standard normal random variable, and  $\Phi(x) = \int_{-\infty}^x \phi(y) dy$  to denote its cumulative distribution function (CDF). We also work with bivariate normal random variables, and to that end introduce the following function.

► **Definition 5.** Let  $\rho \in [-1, 1]$ , and consider two jointly normal random variables  $X, Y$ , with mean 0, and covariance matrix  $\text{Cov}(X, Y) = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$ . We define  $\Gamma_\rho: [0, 1]^2 \rightarrow [0, 1]$  as

$$\Gamma_\rho(x, y) = \Pr [X \leq \Phi^{-1}(x) \wedge Y \leq \Phi^{-1}(y)].$$

We also write  $\Gamma_\rho(x) = \Gamma_\rho(x, x)$ .

The hardness result stated in this paper is based on the Unique Games Conjecture. In order to state this conjecture, we first introduce Unique Games.

► **Definition 6.** A Unique Games instance  $\Lambda = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \Pi, [L])$  consists of an unweighted bipartite multigraph  $(\mathcal{U} \sqcup \mathcal{V}, \mathcal{E})$ , a set  $\Pi = \{\pi_e: [L] \rightarrow [L] \mid e \in \mathcal{E} \text{ and } \pi_e \text{ is a bijection}\}$  of permutation constraints, and a set  $[L]$  of labels. The value of  $\Lambda$  under the assignment  $z: \mathcal{U} \sqcup \mathcal{V} \rightarrow [L]$  is the fraction of edges satisfied, where an edge  $e = (u, v)$ ,  $u \in \mathcal{U}$ ,  $v \in \mathcal{V}$ , is satisfied if  $\pi_e(z(u)) = z(v)$ . We write  $\text{Val}_z(\Lambda)$  for the value of  $\Lambda$  under  $z$ , and  $\text{Opt}(\Lambda)$  for the maximum possible value over all assignments  $z$ .

Let us remark that we require Unique Games instance graph  $(\mathcal{U}, \mathcal{V}, \mathcal{E})$  to be regular. Since Unique Games belong to the class of problems known as Constraint Satisfaction Problems (CSPs), without loss of generality we can assume regularity, as shown in [16].

The Unique Games Conjecture [10] can be stated as follows ([12], Lemma 3.4).

<sup>2</sup> We interpret  $\sigma([t])$  as  $\sigma([t]) = \{\sigma(i) \mid i \in [t]\}$ .

► **Conjecture 7** (Unique Games Conjecture). *For every constant  $\gamma > 0$  there is a sufficiently large  $L \in \mathbb{N}$ , such that for a Unique Games instance  $\Lambda = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \Pi, [L])$  with a regular bipartite graph  $(\mathcal{U} \sqcup \mathcal{V}, \mathcal{E})$ , it is NP-hard to distinguish between*

- $\text{Opt}(\Lambda) \geq 1 - \gamma$ ,
- $\text{Opt}(\Lambda) \leq \gamma$ .

The starting point of hardness result in this work are Affine Unique Games, which are a type of Unique Games defined as follows.

► **Definition 8.** *An Affine Unique Games instance  $\Lambda = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \Pi, [L])$  is a Unique Games Instance  $\Lambda$  in which all permutation constraints  $\pi_e$  are affine constraints. Furthermore, the alphabet  $[L]$  is identified with an additive group  $\mathbb{Z}_L$ , and for each  $\mathcal{E} \ni e = (u, v)$  we have  $\pi_e(x) = x - c_e$ , where  $c_e \in \mathbb{Z}_L$  is a constant.*

We remark that approximating Affine Unique Games is equally hard as approximating Unique Games, in the sense stated by the lemma below which was proved in [11].

► **Lemma 9** (Affine Unique Games Hardness). *Assuming the Unique Games Conjecture, the following statement holds. For every constant  $\gamma > 0$ , there is a sufficiently large  $L \in \mathbb{N}$ , such that for an Affine Unique Games instance  $\Lambda = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \Pi, [L])$  with a regular bipartite graph  $(\mathcal{U} \sqcup \mathcal{V}, \mathcal{E})$ , it is NP-hard to distinguish between*

- $\text{Opt}(\Lambda) \geq 1 - \gamma$ ,
- $\text{Opt}(\Lambda) \leq \gamma$ .

### 3 Hardness Reduction

In this section we state and prove our main result. In Section 3.1 we give a reduction from Affine Unique Games to weighted graphs which satisfy properties sufficient for showing hardness of approximating Min Sum Vertex Cover.

#### 3.1 Reduction from Unique Games to Weighted Graphs

We remark that we use the same type of reduction as in [11, 4, 5], with the only difference being that we now use Affine Unique Games as the starting point, and compared to [5] we are here interested only<sup>3</sup> in the unbiased setting ( $q = 1/2$ ). The main challenge lies in proving completeness, since we will reuse the soundness property of the reduction in the aforementioned results.

Before giving the full proof of the result, we will sketch the ideas behind studying the completeness case now. Consider having a labelling  $z$  which satisfies almost all the edges. Let us describe what happens locally on two vertices  $u, v$ , with a common neighbour  $w$ , which are chosen such that  $(u, w)$  and  $(v, w)$  edges are satisfied by  $z$ . For the sake of simplicity, let us assume that the affine constraints on  $e_1 = (u, w)$ , and  $e_2 = (v, w)$ , are trivial, that is,  $c_{e_1} = c_{e_2} = 0$ , so that the labels  $x_u$  and  $x_v$  are matched by  $z$  if and only if  $x_u = x_v$ . Then, we replace both  $u$  and  $v$  with  $2^L$  strings of length  $L$ . Let us call the sets of strings which replaced  $u$  and  $v$  as  $R$  and  $S$ , respectively. We drop indices  $u, v$  here for the sake of readability. Hence, we have

$$S = \{(s_1, \dots, s_L) \mid s_i \in \{0, 1\}, i \in [L]\}, \quad R = \{(r_1, \dots, r_L) \mid r_i \in \{0, 1\}, i \in [L]\}.$$

<sup>3</sup> We remark that one could also consider using a reduction with biased bits, i.e., the reduction from [5] with  $q \neq 1/2$ . However, this does not yield better inapproximability.

Edges between  $S$  and  $R$  are created as follows. The reduction first fixes some negative correlation parameter  $\rho \in (-1, 0)$ , samples  $L$  times pairs of unbiased,  $\rho$  correlated bits,  $(s_i, r_i), i = 1, \dots, L$ , and then adds an edge between  $s = (s_1, \dots, s_L) \in S$  and  $r = (r_1, \dots, r_L) \in R$ . Let us use  $\nu$  to denote the probability distribution of two  $\rho$  correlated, unbiased bits, i.e.,

$$\nu(0, 0) = \nu(1, 1) = \frac{1 + \rho}{4}, \quad \nu(0, 1) = \nu(1, 0) = \frac{1 - \rho}{4},$$

and study Minimum Sum Vertex Cover on this graph. We will upper bound the value of MSVC on this graph  $G_L$  by<sup>4</sup> some  $T_L$ , by exhibiting an ordering  $\sigma_L$ . Actually, we build our ordering for vertices in  $G_L$  by using the ordering on  $G_{L-1}$ , which is a graph that would have been created with an alphabet size  $L - 1$ . In particular, we observe that the induced subgraph of  $G_L$  obtained by fixing  $s_1 = r_1 = 0$  is isomorphic to  $G_{L-1}$ . Hence, if we use  $\sigma_{L-1}$  to visit vertices in this subgraph, edges with both endpoints in it will be visited by the time  $T_{L-1}$  on average. Since the total weight of edges in this subgraph is  $\nu(0, 0)$ , the cost of covering edges in this subgraph is at most

$$\nu(0, 0) \cdot T_{L-1}.$$

We have spent  $2^L$  steps in visiting this subgraph. Observe that we also covered the edges between strings  $s, r$ , which have  $(s_1, r_1) \in \{(0, 1), (1, 0)\}$ . In particular, we will show that they are covered by the time  $2^L/2$  on average, which intuitively can be seen by observing that we visit  $G_{L-1}$  in  $2^L$  steps, and an average edge will be visited in half that time. This gives us a cost

$$(\nu(0, 1) + \nu(1, 0)) \cdot 2^{L-1}.$$

Finally, the subgraph with  $s, r$  such that  $s_1 = r_1 = 1$  is also isomorphic to  $G_{L-1}$ , and once again use the ordering  $\sigma_{L-1}$  to traverse it in  $2^L$  steps. In this case, we have a delay of  $2^L$  due to visiting vertices with  $r_1 = 0$  or  $s_1 = 0$ , and hence the edges are covered by the time  $2^L + T_{L-1}$ , and their total cost is

$$\nu(1, 1) \cdot (2^L + T_{L-1}).$$

Hence, we have that

$$T_L \leq \nu(0, 0) \cdot T_{L-1} + (\nu(0, 1) + \nu(1, 0)) \cdot 2^{L-1} + \nu(1, 1) \cdot (2^L + T_{L-1}).$$

Letting  $t_L = T_L/2^{L+1}$  and replacing the values of  $\nu$  yields

$$t_L \leq \frac{1 + \rho}{4} t_{L-1} + \frac{1}{4},$$

which is a recurrence relation, and solving it shows that  $t_L \rightarrow \frac{1}{3-\rho}$ , regardless of  $t_1$ . Hence, for sufficiently large  $L$  we should expect to get MSVC close to  $\frac{1}{3-\rho}$ .

With this intuition in mind, we now state and prove the theorem which gives the hardness reduction from Affine Unique Games to weighted graphs.

► **Theorem 10.** *For any  $\varepsilon > 0, \rho \in (-1, 0), \gamma > 0$ , there is a sufficiently large alphabet size  $L \in \mathbb{N}$  and a reduction from regular Affine Unique Games instances  $\Lambda = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \Pi, [L])$  to weighted multigraphs  $G = (V, E)$  with the following properties:*

<sup>4</sup> Without loss of generality, we assume that weights of edges sum up to 1 here.

- Completeness: If  $\text{Opt}(\Lambda) \geq 1 - \gamma$ , then  $\text{MSVC}(G) \leq \left(\frac{1}{3-\rho} + \varepsilon + 3\gamma\right) |V|$ .
- Soundness: If  $\text{Opt}(\Lambda) \leq \gamma$ , then for every  $r \in [0, 1]$ ,  $G$  is  $(r, \Gamma_\rho(r) - \varepsilon)$ -dense. Moreover, the running time of the reduction is polynomial in  $|\mathcal{U}|, |\mathcal{V}|, |\mathcal{E}|$ , and exponential in  $L$ . If we use  $D$  to denote the degree of the regular Unique Games instance, then the weights of edges in  $G$  belong to the set  $\left\{\left(\frac{1+\rho}{4}\right)^i \left(\frac{1-\rho}{4}\right)^{L-i}\right\}_{i=0}^L$ . The size of  $|V|$  is at least  $2^L$ . Finally, the output graph  $G$  is also regular, in the following two senses. First, if we consider  $G$  as an unweighted graph, every vertex is of degree  $D^2$ . The graph  $G$  is also regular in the weighted sense, i.e., the value  $W_E(u, N(u))$  is uniform across all  $u \in V$ , and it equals  $D^2 2^{-L+1}$ .

**Proof.** Let  $\nu: \{0, 1\}^2 \rightarrow [0, 1]$  be the probability distribution over correlated uniformly distributed bits with negative correlation coefficient  $\rho < 0$ . In other words, we have

$$\nu(0, 0) = \nu(1, 1) = \frac{1+\rho}{4}, \quad \nu(0, 1) = \nu(1, 0) = \frac{1-\rho}{4}.$$

Let us now describe how the multigraph  $G$  can be constructed from  $\Lambda$ . We define the vertex set of  $G$  to be  $V = \mathcal{V} \times \{0, 1\}^L = \{(v, x) \mid v \in \mathcal{V}, x \in \{0, 1\}^L\}$ . In particular, for every vertex  $v \in \mathcal{V}$  we create  $2^L$  vertices of  $G$ , which we identify with  $L$ -bit strings in  $\{0, 1\}^L$ . We also write  $v^x$  for a vertex  $(v, x)$  of the graph  $G$ . The edges of  $G$  are constructed in the following way. For every  $u \in \mathcal{U}$ , and for every two  $v_1, v_2 \in N(u)$ , we create an edge between vertices  $v_1^x, v_2^y$  with weight

$$\nu^{\otimes L}(x \circ \pi_{e_1}, y \circ \pi_{e_2}), \quad \text{where } e_1 = (u, v_1), \quad e_2 = (u, v_2).$$

Expressed formally, the edge set  $E$  is

$$E = \{(e_1^x, e_2^y) \mid e_1 = (u, v_1), e_2 = (u, v_2), u \in \mathcal{U}, v_1, v_2 \in \mathcal{V}, x, y \in \{0, 1\}^L\}.$$

The number of vertices in  $G$  is  $|\mathcal{V}|2^L$ , and the number of edges is  $|\mathcal{V}|D^2 2^L$ , so the construction is indeed polynomial in  $|\mathcal{U}|, |\mathcal{V}|$  and  $|\mathcal{E}|$ , and exponential in  $L$ . Also, since  $\mathcal{V} \neq \emptyset$  we have  $|V| \geq 2^L$ , and the weights of the edges indeed belong to the set specified in the statement of the theorem. Finally, the total weight of edges incident upon each vertex  $v^x$  is the same for any  $v^x$ , and since  $W_E(E) = D^2 |\mathcal{V}|$ , we have that  $w_E(v^x, N(v^x)) = 2D^2 |\mathcal{V}| \frac{1}{|V|} = D^2 2^{-L+1}$  for all  $v^x \in V$ .

We are using the same reduction<sup>5</sup> as the one used in Theorem 3.1. from [5], and the only difference is that we are starting from Affine Unique Games instead of (general) Unique Games. Since we are using the same reduction and Affine Unique Games are subsumed by the Unique Games, our graph  $G$  satisfies the same soundness property as the one expressed by Theorem 3.1. in [5], and this is exactly the soundness property stated above. Hence, we only need to show completeness.

For the completeness case let us assume  $\text{Opt}(\Lambda) \geq 1 - \gamma$ . Therefore, there is a labelling  $z: \mathcal{U} \sqcup \mathcal{V} \rightarrow \mathbb{Z}_L$  such that  $\text{Val}_z(\Lambda) \geq 1 - \gamma$ . In particular, there is  $\hat{\mathcal{E}} \subseteq \mathcal{E}$ ,  $|\hat{\mathcal{E}}| \geq (1 - \gamma)|\mathcal{E}|$ , such that for each  $e = (u, v) \in \hat{\mathcal{E}}$  we have  $z(u) - z(v) = c_e$ . Let us use  $\hat{E} \subseteq E$  to denote the set

$$\hat{E} := \{(e_1^x, e_2^y) \in E \mid e_1, e_2 \in \hat{\mathcal{E}}\}.$$

Observe that  $|\hat{E}| \geq |E| \cdot (1 - 2\gamma)$ . Since the complement of  $\hat{E}$  is of small fractional size, i.e., smaller than  $2\gamma$ , in the analysis we will focus on cover times of edges in  $\hat{E}$ , and we will

<sup>5</sup> This is the same as the Max-Cut hardness reduction in [11]. Same reduction and soundness result also appeared in [4], albeit with biased bits.

trivially upper bound the cover time of edges in  $\hat{E}^c$  by  $|V|$ . In particular, let us denote with  $\hat{G}$  the graph  $\hat{G} = (V, \hat{E})$  and find an ordering  $\sigma$  such that  $SVC_{\hat{G}}(\sigma) \leq (\frac{1}{3-\rho} + \varepsilon + \gamma)|V|$ . As discussed this would then give us the stated completeness

$$SVC_G(\sigma) \leq \left( \frac{1}{3-\rho} + \varepsilon + 3\gamma \right) |V|,$$

by bounding the cover time of edges in  $\hat{E}^c$  by  $|V|$ .

Before explaining how  $\sigma$  is constructed, let us first introduce some notation. We use  $z_1, \dots, z_L: \mathcal{U} \rightarrow \mathbb{Z}_L$  to denote the mappings defined by

$$z_i(u) := z(u) + i, \quad \text{for } i \in [L].$$

Let us then define sets  $F_i^0, F_i^1 \subseteq V$ , as the sets in which, for every  $v \in \mathcal{V}$ , inside the long code  $(v, x)$  we fix the  $z_i(v)$ -th coordinate to 0 or 1, respectively. In particular, we have

$$F_i^0 = \{(v, x) \in V \mid x_{z_i(v)} = 0\}, \quad F_i^1 = \{(v, x) \in V \mid x_{z_i(v)} = 1\}.$$

Intuitively, the sets  $F_i^0$  (or  $F_i^1$ ) for a fixed  $i$  fix the values at the coordinates in which labels “agree”. Then, we use the sets  $F_i^0$  and  $F_i^1$  to construct ordering inductively. First, we define the ordering on  $C_{L-1} = F_1^0 \cap F_2^0 \cap \dots \cap F_{L-1}^0$ , then using this ordering we define ordering on  $C_{L-2} = F_1^0 \cap F_2^0 \cap \dots \cap F_{L-2}^0$ , and so on until we construct an ordering on  $C_1 = F_1^0$  and finally on  $C_0$  which we define to be  $C_0 := V$ . As we are defining orderings on  $C_i, i = 0, \dots, L-1$ , we will be expressing an upper bound  $T_i$  for the average time edges  $E_i$  with both endpoints endpoints in  $C_i$  are covered by the ordering. Before discussing our ordering, let us make an observation that  $|C_i| = 2 \cdot |C_{i+1}|$ , since  $C_i$  has one more free coordinate for each  $v \in \mathcal{V}$ .

We discuss the ordering for  $C_{L-1}$  first. Before that, let us remark that the particular ordering and the cost of covering edges in  $C_{L-1}$  will be inconsequential for the final value that we get in this theorem. The main reason we discuss this case here is because we believe it will be a good preparation for discussing the inductive step that will follow. Let us first normalize the weights of edges in  $E_{L-1}$  so that they sum up to 1. In the first step, we iterate through  $v \in \mathcal{V}$  in a random order<sup>6</sup>, and pick  $(v, x) \in C_{L-1}$  such that<sup>7</sup>  $x_{z_L(v)} = 0$ . Then, we iterate through  $v \in \mathcal{V}$  in a random order and pick the remaining vertex at each  $(v, x)$ , i.e., the vertex with  $x_{z_L(v)} = 1$ . Let us upper bound the average time an edge  $e \in E_{L-1}$  with both endpoints in  $C_{L-1}$  is visited by this schedule. Observe that we spent  $\frac{1}{2}|C_{L-1}|$  time in the first step, and  $\frac{1}{2}|C_{L-1}|$  in the second step. Thus, if an edge with both endpoints in  $C_{L-1}$  has at least one endpoint with a label 0 at  $x_{z_L(v)}$ , then this point will be picked in the first step on average by the time  $\frac{1}{4}|C_{L-1}|$ . Otherwise, if the edge  $e$  has both endpoints  $v_1^x, v_2^y$  picked in the second step, i.e.  $x_{z_L(v_1)} = 1, y_{z_L(v_2)} = 1$ , then it will be picked on average by the time  $\frac{3}{4}|C_{L-1}|$ . Since the weight of edges from  $E_{L-1}$  picked in the first step is  $\nu(0, 0) + \nu(0, 1) + \nu(1, 0) = \frac{3-\rho}{4}$ , and the weight of the remaining edges that we consider is  $\nu(1, 1) = \frac{1+\rho}{4}$ , the average cover time is

$$T_{L-1} = \frac{3-\rho}{4} \cdot \frac{1}{4}|C_{L-1}| + \frac{1+\rho}{4} \cdot \frac{3}{4}|C_{L-1}| = \frac{3+\rho}{8}|C_{L-1}|.$$

We observe that this also shows that there is an ordering  $\sigma_{L-1}$  which covers an edge in  $E_{L-1}$  on average by the time  $T_{L-1}$ .

<sup>6</sup> As we have said, the value obtained in the first step is not relevant as it will be seen later. Hence, we can also choose to visit  $v \in \mathcal{V}$  in any fixed order in which case we can also use a trivial upper bound of  $|V_{L-1}|$  on  $T_{L-1}$ .

<sup>7</sup> Due to symmetry it is not important whether we pick  $x_{z_L(v)} = 0$  or  $x_{z_L(v)} = 1$  in the first iteration, as long as we keep that choice fixed.

## 50:10 Some Results on Approximability of Minimum Sum Vertex Cover

Let us now fix  $i = 0, \dots, L - 2$ , and assume that we have an ordering  $\sigma_{i+1}$  of vertices in  $C_{i+1}$  such that the edges in  $E_{i+1}$  are covered by the time  $T_{i+1}$  on average, and let us use this procedure to construct an ordering of the vertices in  $C_i$  and derive a suitable upper bound on  $T_i$ . We assume that the weights of the edges  $E_i$  are normalized so that they sum up to 1. The ordering in  $C_i$  works as follows. First, using  $\sigma_{i+1}$  we visit vertices in  $C_{i+1} = C_i \cap F_i^0$ . The total weight of the edges with both endpoints in  $C_i \cap F_i^0$  is  $\nu(0, 0)$ , and they are covered by  $\sigma_{i+1}$  until  $T_i$  on average. Hence, the cost for these edges is

$$T_{i+1} \cdot \nu(0, 0). \quad (4)$$

Furthermore, during this pass, we have also visited all the edges with one endpoint in  $C_i \cap F_i^0$  and another endpoint in  $C_i \cap F_i^1$ , and their total weight is  $\nu(0, 1) + \nu(1, 0)$ . Also, these covers are disjoint (each one of these edges will be visited only once in the first pass). Since the starting Unique Games instance was regular and we removed at most  $2\gamma$  edges, the edges will be covered by the time

$$\frac{1 + 2\gamma}{2} |C_{i+1}| \quad (5)$$

at most. Hence, the cost for these edges is

$$(\nu(0, 1) + \nu(1, 0)) \frac{1 + 2\gamma}{2} |C_{i+1}|. \quad (6)$$

Finally, we pass through the vertices in  $C_i \cap F_i^1$ . The graph induced by this vertex set is actually isomorphic to  $C_{i+1} = C_i \cap F_i^0$ , and hence we can once again use the ordering  $\sigma_{i+1}$ . Then, the edges in this graph are visited on average by the time

$$|C_{i+1}| + T_{i+1},$$

where the  $|C_{i+1}|$  term is due to the delay coming from the first pass. Hence, the cost of these edges is at most

$$\nu(1, 1)(|C_{i+1}| + T_{i+1}). \quad (7)$$

Adding up (4), (6) and (7) we get that

$$T_i \leq \frac{1 + \rho}{4} T_{i+1} + \frac{1 - \rho}{2} \frac{1 + 2\gamma}{2} |C_{i+1}| + \frac{1 + \rho}{4} \cdot (|C_{i+1}| + T_{i+1}). \quad (8)$$

If we let  $t_i = T_i / |C_i|$  and divide both sides by  $|C_i| = 2|C_{i+1}|$ , we can write (8) as

$$t_i \leq \frac{1 + \rho}{8} t_{i+1} + \frac{1 - \rho}{4} \frac{1 + 2\gamma}{2} + \frac{1 + \rho}{4} \left( \frac{1 + t_{i+1}}{2} \right),$$

which can be simplified to

$$t_i \leq \frac{1}{4} + \frac{1 + \rho}{4} t_{i+1} + \frac{1 - \rho}{4} \gamma. \quad (9)$$

Let us show that  $t_i \leq \frac{1}{3 - \rho} + \gamma + 2^{-L+i}$  as follows. Let us define  $r_i = t_i - \gamma - \frac{1}{3 - \rho}$ . By substituting  $t_i = \frac{1}{3 - \rho} + \gamma + r_i$  into (9) we obtain

$$\frac{\gamma}{2} + r_i \leq \frac{1 + \rho}{4} r_{i+1}. \quad (10)$$



Since  $\rho \in (-1, 0)$  and  $\gamma > 0$  we have

$$r_i \leq \frac{1}{2} r_{i+1}. \quad (11)$$

Hence, since by calculation for  $T_{L-1}$  we have  $r_{L-1} \leq \frac{1}{2}$ , which with (11) implies that  $r_i \leq 2^{-L+i}$ , and therefore  $t_0 \leq \frac{1}{3-\rho} + \gamma + 2^{-L}$ . By letting  $L$  be large enough so that  $2^{-L} \leq \varepsilon$  and recalling that  $t_0 = T_0/|V|$  we get

$$T_0 \leq \left( \frac{1}{3-\rho} + \gamma + \varepsilon \right) |V|,$$

which is what we wanted to prove.  $\blacktriangleleft$

This reduction outputs a weighted graph. In the full version of this paper we will show how this weighted graph can be transformed into an unweighted graph with *essentially* the same properties using a polynomial time reduction. For now, let us briefly discuss how the soundness and completeness properties stated in the theorem above are useful for studying Min Sum Vertex Cover.

For completeness, we will get that  $\text{MSVC}(G) \leq \left( \frac{1}{3-\rho} + \varepsilon + 3\gamma \right) |V|$ . On the other hand, in the soundness case we have that for any ordering  $\sigma$  and given any  $\eta > 0$  we have

$$\begin{aligned} \text{SVC}_G(\sigma) &= \sum_{t=1}^{|V|} w(\sigma([t])^c, \sigma([t])^c) \geq \sum_{t=1}^{n-\lceil \eta n \rceil} w(\sigma([t])^c, \sigma([t])^c) \geq \sum_{t=1}^{n-\lceil \eta n \rceil} \Gamma_\rho(1 - t/n) - \varepsilon \\ &= \left( \int_0^{1-\eta} \Gamma_\rho(1-r) dr - \varepsilon \right) \cdot |V| + O(1) = \left( \int_\eta^1 \Gamma_\rho(r) dr - \varepsilon \right) \cdot |V| + O(1). \end{aligned}$$

Hence, by letting  $\eta \rightarrow 0, \gamma \rightarrow 0, \varepsilon \rightarrow 0, |V| \rightarrow \infty$ , we get an inapproximability ratio of

$$\frac{\int_0^1 \Gamma_\rho(r) dr}{\frac{1}{3-\rho}}.$$

This expression is minimized for  $\rho \approx -0.52$ , for which the inapproximability ratio is approximately 1.014.

Numerical simulations show that the best ratio we can get with these techniques is 1.014, and it is obtained for  $\rho = -0.52$ .

## 4 Approximating Min Sum Vertex Cover on Regular Graphs

In this section we will revisit an approximation algorithm for Minimum Sum Vertex Cover on regular graphs introduced in [9], in Theorem 11. The authors in that work did not explicitly state the approximation ratio obtained by that algorithm, since their primary interest was showing that 4/3-approximation achieved by the greedy algorithm can be beaten by more advanced techniques.

We will here give an explicit constant, also taking into account progress in the approximation of the so called Max- $k$ -VC problem, which is used in that approach, and for which better algorithms exist since the publication of the aforementioned article.

Before discussing the algorithm, let us define the Max- $k$ -VC problem. In this problem a graph  $G = (V, E)$  is given as an input, and the goal is to find  $S \subseteq V, |S| = k$ , such that  $w(S, V)$  is as big as possible. Austrin, Benabbas and Georgiou [3] show that Max-2-Sat with a bisection constraint, that is, Max-2-Sat in which admissible assignments have exactly half

## 50:12 Some Results on Approximability of Minimum Sum Vertex Cover

of the variables set to 1, and the other half to 0, can be approximated within  $\alpha_{LLZ} \approx 0.9401$ . Let us remark that  $\alpha_{LLZ}$  is the optimal<sup>8</sup> approximation ratio for Max-2-Sat problem [14, 2]. Since this problem subsumes Max- $k$ -VC when  $k = n/2$ , we can approximate Max- $n/2$ -VC within  $\alpha_{LLZ} \approx 0.9401$ .

Let us also recall the following two facts for regular graphs:

- The greedy algorithm on regular graphs covers edges on average by the time  $\frac{1}{3}|V|$ ,
- The optimal solution covers an edge on average by the time at least  $\frac{1}{4}|V|$ .

Let us now discuss the algorithm introduced in [9]. We will closely follow the argument outlined there. Let  $\varepsilon > 0$  be some constant that we will fix later. In case the optimal solution covers an edge by the time  $(\frac{1}{4} + \varepsilon)|V|$  later, the greedy algorithm approximates the optimal value within a factor of

$$\frac{1/3}{\frac{1}{4} + \varepsilon} = \frac{4}{3 + \frac{3}{4}\varepsilon}.$$

Otherwise, the optimal solution covers an edge on average at the time  $(\frac{1}{4} + \delta)|V|$ , for some  $\delta \in (0, \varepsilon)$ . In this case, we have the following lemma.

► **Lemma 11.** *Let  $G = (V, E)$  be a regular graph, let  $n := |V|$ , and let the optimal solution of Minimum Sum Vertex Cover be  $(\frac{1}{4} + \delta)|V|$ . Then the optimal solution covers at least  $(1 - \sqrt{\delta})$  fraction of edges in the first  $n/2$  steps.*

**Proof.** Let us denote the degree of the graph with  $D \in \mathbb{N}$ , and with  $m$  the number of edges  $m = nD/2$ . We argue by contradiction, and assume that the optimal solution covers less than  $(1 - \sqrt{\delta})$  fraction of edges in the first  $n/2$  steps. Let us use  $u_i, i = 1, \dots, n$ , to denote the number of uncovered edges at the time step  $i$ , and let  $s := u_{n/2}$ . Then by assumption  $s > \sqrt{\delta}m$ . Furthermore, the value of the minimum sum vertex cover is  $\frac{1}{m} \sum_{i=1}^n u_i$ . Let us show that  $\frac{1}{m} \sum_{i=1}^n u_i > (\frac{1}{4} + \delta)n$  yielding a contradiction to the assumption that the optimal solution of Minimum Sum Vertex Cover is  $(\frac{1}{4} + \delta)n$ .

Let us use  $c_i = u_i - u_{i-1}$  to denote the number of additionally covered edges at step  $i$ . Since we are considering the optimal solution to MSVC, the sequence  $c_i$  is non-increasing (otherwise changing the order would yield a smaller solution). Furthermore, let us use  $c$  to denote  $c_{n/2}$ .

Now, assuming that  $c_{n/2} = c$ ,  $u_{n/2} = s$ , let us calculate the smallest possible value of MSVC. We know that after  $i$  steps, we can cover at most  $i \cdot D$  edges (this happens if all the edges chosen are disjoint). Furthermore, since we assumed that after  $n/2$  steps we leave  $s$  edges uncovered, and since  $c = c_{n/2}$  and  $c$  is non-increasing, we have that at the step  $i$  we leave at least  $s + (n/2 - i) \cdot c$  edges uncovered. This shows that

$$u_i \geq \max \left( \frac{nD}{2} - iD, s + (n/2 - i) \cdot c, 0 \right), i \in [n].$$

In particular, the right hand side is a maximum of three linear functions, and therefore, the following scenario for covering the edges will lower bound  $u_i$ . In the first  $t$  fraction of steps, edges get covered at the optimal rate (at each step we cover  $D$  new edges), where  $t$  is a parameter calculated later. Then, after  $t$  fraction of steps, we cover  $c$  edges, until we cover all the edges<sup>9</sup>.

<sup>8</sup> Assuming the Unique Games Conjecture

<sup>9</sup> In the last step we might cover less than  $c$  edges, but we will ignore this case for the sake of simplicity.

Since we spend  $t$  fraction of time covering  $D$  edges in each step, the cost of edges covered in this time is

$$\frac{1}{m} \sum_{i=1}^{t \cdot n} \frac{Dn}{2} - iD \geq nt(1-t).$$

The remaining time  $x \cdot n$ , for some  $x \in (0, 1)$ , is spent on covering  $c$  edges at each step. Since after  $x$  fraction of steps we covered all the edges, we have

$$m = t \cdot n \cdot D + x \cdot n \cdot c,$$

and since  $m = nD/2$  we have that

$$x = \frac{D}{2} \cdot \frac{1-2t}{c}. \quad (12)$$

Hence, the average cost of edges incurred in the remaining time is

$$\frac{1}{m} x \cdot n \cdot (m - t \cdot n \cdot D) \frac{1}{2} = xn \left( \frac{1}{2} - t \right),$$

which with (12) yields the cost

$$\frac{D}{2} \cdot \frac{1-2t}{c} n \left( \frac{1}{2} - t \right) = n \frac{D}{4c} (1-2t)^2.$$

Hence, the total cost is

$$nt(1-t) + n \frac{D}{4c} (1-2t)^2 \quad (13)$$

Let us now calculate the value of  $t$  in terms of  $s$  and  $c$ . We use the fact that after  $t$  fraction of steps we covered  $t \cdot n \cdot D$  edges, and after  $n/2$  steps we covered  $m - s$  edges. Since we are covering  $c$  edges at each step  $i \in [tn, n/2]$  we have that

$$m - s = tnD + \left( \frac{n}{2} - tn \right) \cdot c,$$

and from here we get

$$t = \frac{m - s - \frac{n \cdot c}{2}}{n \cdot D - n \cdot c} = \frac{1}{2} - \frac{s}{n(D-c)}.$$

Replacing this in (13) we get that the total cost is at least

$$n \left( \frac{1}{4} - \frac{s^2}{n^2(D-c)^2} + \frac{D}{4c} \left( 2 \frac{s}{n(D-c)} \right)^2 \right)$$

which reduces to

$$n \left( \frac{1}{4} + \frac{s^2}{n^2} \frac{1}{(D-c)c} \right)$$

Now, by our contradiction hypothesis we have  $s > \sqrt{\delta}m$ , and  $\frac{1}{(D-c)c} \geq 4 \frac{1}{D^2}$  (since  $c \in (1, D)$ ), we have that the total cover time is strictly greater than

$$n \left( \frac{1}{4} + \frac{\delta m^2}{n^2} \frac{4}{D^2} \right) = n \left( \frac{1}{4} + \delta \right),$$

which contradicts the fact that the optimal solution to MSVC on the graph  $G$  has value  $n(\frac{1}{4} + \delta)$ . This concludes our proof.  $\blacktriangleleft$

In [9] it is claimed that in the setup of Lemma 11, the optimal solution covers  $(1 - \delta)$  fraction of edges. However, this is not correct, and we illustrate that with a counterexample provided in the appendix. Nevertheless, this does not greatly change the conclusion in [9], as using the correct version of the lemma just replaces one unspecified constant below  $4/3$  by another unspecified constant below  $4/3$ .

Let us now fix  $k = n/2$ , and use the Max- $k$ -VC algorithm. This will give us a set  $S \subseteq V$  such that  $w(S, V) \geq \alpha_{LLZ}(1 - \sqrt{\delta})$ . We next consider the following ordering of vertices in  $V$  and calculate Minimum Sum Vertex Cover for it. We first pick vertices from  $S$  in a random order, and then take the remaining vertices in random order as well. Then, the vertices in  $N(S, S)$  are covered by the time  $|V|/4$  in expectation, while the remaining vertices are covered by the time  $\frac{|V|}{2} + \frac{1}{3} \frac{|V|}{2}$ . Hence, we can find an ordering for which Sum Vertex Cover has value

$$w(S, V) \frac{|V|}{4} + w(S^c, S^c) \left( \frac{|V|}{2} + \frac{|V|}{6} \right) \leq \alpha_{LLZ}(1 - \sqrt{\delta}) \frac{|V|}{4} + \left( 1 - \alpha_{LLZ}(1 - \sqrt{\delta}) \right) \cdot \frac{2|V|}{3}.$$

We can simplify this expression as

$$\left( \frac{-5\alpha_{LLZ} + 5\alpha_{LLZ}\sqrt{\delta}}{12} + \frac{2}{3} \right) |V|.$$

Hence, we get an approximation ratio of

$$\frac{\frac{-5\alpha_{LLZ} + 5\alpha_{LLZ}\sqrt{\delta}}{12} + \frac{2}{3}}{\frac{1}{4} + \delta}.$$

In conclusion, for fixed  $\varepsilon$  we have that the approximation ratio is given as

$$\max \left( \frac{4}{3 + \frac{3}{4}\varepsilon}, \sup_{\delta \in (0, \varepsilon]} \frac{\frac{-5\alpha_{LLZ} + 5\alpha_{LLZ}\sqrt{\delta}}{12} + \frac{2}{3}}{\frac{1}{4} + \delta} \right).$$

Optimizing over different values of  $\varepsilon$  gives us that the approximation ratio of this algorithm is approximately 1.225.

---

## References

- 1 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998. doi:10.1145/278298.278306.
- 2 Per Austrin. Balanced max 2-sat might not be the hardest. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 189–197, 2007. doi:10.1145/1250790.1250818.
- 3 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 277–294, 2013. doi:10.1137/1.9781611973105.21.
- 4 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory Comput.*, 7(1):27–43, 2011. doi:10.4086/toc.2011.v007a003.
- 5 Per Austrin and Aleksa Stankovic. Global cardinality constraints make approximating some max-2-csps harder. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA.*, volume 145 of *LIPIcs*, pages 24:1–24:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.24.

- 6 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 453–462. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.23.
- 7 Uri Barenholz, Uriel Feige, and David Peleg. Improved approximation for min-sum vertex cover. *Technical report, MCS06-07, Computer Science and Applied Mathematics*, 2006.
- 8 Vaggos Chatziafratis, Neha Gupta, and Euiwoong Lee. Inapproximability for local correlation clustering and dissimilarity hierarchical clustering. *CoRR*, abs/2010.01459, 2020. arXiv:2010.01459.
- 9 Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004. doi:10.1007/s00453-004-1110-5.
- 10 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775, 2002. doi:10.1145/509907.510017.
- 11 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.
- 12 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within  $2 - \varepsilon$ . In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, page 379, 2003. doi:10.1109/CCC.2003.1214437.
- 13 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. doi:10.1016/j.jcss.2007.06.019.
- 14 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-sat and MAX DI-CUT problems. In *Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings*, pages 67–82, 2002. doi:10.1007/3-540-47867-1\_6.
- 15 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 245–254, 2008. doi:10.1145/1374376.1374414.
- 16 Aleksa Stankovic. On regularity of max-csps and min-csps. *Inf. Process. Lett.*, 176:106244, 2022. doi:10.1016/j.ipl.2022.106244.

## A Addressing Argument in Theorem 11 in [9]

Let  $\delta > 0$ . We construct a 2-regular graph  $G = (V, E)$  with  $|V| = n, |E| = m$ , such that the minimum sum vertex cover has value

$$\left(\frac{1}{4} + \delta\right)n,$$

while any set  $S \subseteq V, |S| = \frac{n}{2}$ , satisfies  $|w(S, V)| \leq (1 - \sqrt{\delta})$ . This shows that the factor  $(1 - \sqrt{\delta})$  in Lemma 11 can not be replaced by a sharper  $(1 - \delta)$ , as it was done in [9].

Let  $t = (\frac{1}{2} - 3\sqrt{\delta})n$  and  $s = 2\sqrt{\delta} \cdot n$ , where  $n \in \mathbb{N}$  is chosen such that  $t, s \in \mathbb{N}$  (we also approximate  $\sqrt{\delta}$  by a rational number), and such that  $t$  is even. Then, we construct the graph  $G$  by taking  $t/2$  disjoint copies of  $K_{2,2}$  and  $s$  disjoint copies of  $K_3$ . Let  $V_1$  be the set composed of only “the left sides” of  $t/2$  disjoint copies of  $K_{2,2}$ ,  $V_2$  the set composed of only “the right sides” of  $t/2$  disjoint copies of  $K_{2,2}$ , and let  $V_3$  be composed of the vertices from  $s$  disjoint copies of  $K_3$ .

Then, the optimal solution for MSVC will work in the following three stages:

- *Stage 1:* Pick vertices from  $V_1$  in any order.
- *Stage 2:* Pick one vertex from each  $K_3$  in  $V_3$ .
- *Stage 3:* Pick another vertex from each  $K_3$  in the set  $V_3$ .

## 50:16 Some Results on Approximability of Minimum Sum Vertex Cover

It is clear that this is the minimum sum vertex cover. The total cost is then split into the following costs:

- Edges covered in the first stage. In this case we pick  $t \cdot 2$  edges, and an edge is picked on average at the time  $t/2$ , so the cost is

$$\frac{t}{2} \cdot 2 \cdot t = t^2.$$

- Edges covered in the second stage. We pick  $s \cdot 2$  edges, and each edge is picked on average at the time  $t + s/2$ , where the factor  $t$  exists because this step happens after the first stage. We have the cost of

$$2 \cdot s(t + s/2) = 2 \cdot s \cdot t + s^2.$$

- Edges covered in the third stage. We pick  $s$  edges, and each edge is picked on average at the time  $t + s + s/2$ , where the factor  $t + s$  exists because this step happens after the second stage. We have the cost of

$$s(t + s + s/2) = st + s^2 + s^2/2.$$

Hence, the total cost is

$$t^2 + 2 \cdot s \cdot t + s^2 + st + s^2 + s^2/2 = t^2 + 3st + \frac{5s^2}{2}.$$

Now, recalling that  $t = (\frac{1}{2} - 3\sqrt{\delta})n$  and  $s = 2\sqrt{\delta} \cdot n$ , we have that the cost is

$$\begin{aligned} n^2 \cdot \left( \frac{1}{4} - 3\sqrt{\delta} + 9\delta \right) + 3 \cdot 2\sqrt{\delta} \cdot n \cdot \left( \frac{1}{2} - 3\sqrt{\delta} \right) \cdot n + \frac{5 \cdot 4\delta \cdot n^2}{2} \\ = \frac{1}{4}n^2 - 3\sqrt{\delta}n^2 + 9\delta n^2 + 3\sqrt{\delta}n^2 - 18\delta n^2 + 10\delta n^2 \\ = \frac{1}{4}n^2 + \delta n^2. \end{aligned}$$

Now, since our graph is 2-regular graph on  $n = 2t + 3s$  vertices, we have that  $m = n$ , and we can write the total cost as

$$n \left( \frac{1}{4} + \delta \right),$$

as claimed. It remains to show that for any set  $S \subseteq V$  with  $|S| = n/2$  we have

$$|E(S, V)| \leq (1 - \sqrt{\delta}) \cdot m.$$

It is obvious that the worst case  $S$  is exactly the set of vertices picked in the first  $n/2$  steps in the algorithm above. In this case, the number of edges not covered is  $s/2$ , since after  $n/2$  steps we are left with one edge uncovered in exactly half of the  $K_3$  triangles. Hence, the number of uncovered edges is

$$\frac{s}{2} = \sqrt{\delta} \cdot n = \sqrt{\delta} \cdot m,$$

and hence



$$|E(S, V)| \leq (1 - \sqrt{\delta}) \cdot m,$$

as required.

# $(1 + \epsilon)$ -Approximate Shortest Paths in Dynamic Streams

Michael Elkin  

Ben-Gurion University of the Negev, Beer-Sheva, Israel

Chhaya Trehan  

London School of Economics & Political Science, UK

---

## Abstract

Computing approximate shortest paths in the dynamic streaming setting is a fundamental challenge that has been intensively studied. Currently existing solutions for this problem either build a sparse multiplicative spanner of the input graph and compute shortest paths in the spanner offline, or compute an exact single source BFS tree. Solutions of the first type are doomed to incur a stretch-space tradeoff of  $2\kappa - 1$  versus  $n^{1+1/\kappa}$ , for an integer parameter  $\kappa$ . (In fact, existing solutions also incur an extra factor of  $1 + \epsilon$  in the stretch for weighted graphs, and an additional factor of  $\log^{O(1)} n$  in the space.) The only existing solution of the second type uses  $n^{1/2-O(1/\kappa)}$  passes over the stream (for space  $O(n^{1+1/\kappa})$ ), and applies only to unweighted graphs.

In this paper we show that  $(1 + \epsilon)$ -approximate single-source shortest paths can be computed with  $\tilde{O}(n^{1+1/\kappa})$  space using just *constantly* many passes in unweighted graphs, and polylogarithmically many passes in weighted graphs. Moreover, the same result applies for multi-source shortest paths, as long as the number of sources is  $O(n^{1/\kappa})$ . We achieve these results by devising efficient dynamic streaming constructions of  $(1 + \epsilon, \beta)$ -spanners and hopsets.

On our way to these results, we also devise a new dynamic streaming algorithm for the 1-sparse recovery problem. Even though our algorithm for this task is slightly inferior to the existing algorithms of [26, 11], we believe that it is of independent interest.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming models; Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Theory of computation  $\rightarrow$  Shortest paths; Theory of computation  $\rightarrow$  Sparsification and spanners

**Keywords and phrases** Shortest Paths, Dynamic Streams, Approximate Distances, Spanners, Hopsets

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.51

**Category** APPROX

**Related Version** *Extended Version*: <https://arxiv.org/abs/2107.13309> [20]

**Funding** *Michael Elkin*: This research was supported by ISF grant 2344/19.

## 1 Introduction

Processing massive graphs is an important algorithmic challenge. This challenge is being met by intensive research effort. One of the most common theoretical models for addressing this challenge is the *semi-streaming* model of computation [22, 2, 36]. In this model, edges of an input  $n$ -vertex graph  $G = (V, E)$  arrive one after another, while the storage capacity of the algorithm is limited. Typically it should be close to linear in the number of *vertices*,  $n$ . One usually allows space of  $\tilde{O}(n)$ , though it is often relaxed to  $n^{1+o(1)}$ , sometimes to  $O(n^{1+\rho})$ , for an arbitrarily small constant parameter  $\rho > 0$ , or even to  $O(n^{1+\eta_0})$ , for some fixed constant  $\eta_0$ ,  $0 < \eta_0 < 1$ . Generally, the model allows several passes over the stream, and the objective is to keep both the number of passes and the space complexity of the algorithm in check.



© Michael Elkin and Chhaya Trehan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 51; pp. 51:1–51:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



The model comes in two main variations. In the first one, called *static* or *insertion-only* model [22], the edges can only arrive, and never get deleted. If the algorithm employs multiple passes, then the streams of edges observed on these passes may be permutations of one another, but are otherwise identical. In the more general *dynamic* (also known as *turnstile*) streaming setting [2], edges may either arrive or get deleted.

## 1.1 Distances in the Streaming Model

An important thread of the literature on dynamic streaming algorithms for graph problems is concerned with computing *distances* and constructing *spanners* and *hopsets*. This is also the topic of the current paper. For a pair of parameters  $\alpha \geq 1$ ,  $\beta \geq 0$ , given an undirected graph  $G = (V, E)$ , a subgraph  $G' = (V, H)$  of  $G$  is said to be an  $(\alpha, \beta)$ -*spanner* of  $G$ , if for every pair  $u, v \in V$  of vertices, it holds that  $d_{G'}(u, v) \leq \alpha \cdot d_G(u, v) + \beta$ , where  $d_G$  and  $d_{G'}$  are the distance functions of  $G$  and  $G'$ , respectively. A spanner with  $\beta = 0$  is called a *multiplicative* spanner and one with  $\alpha = 1$  is called an *additive* spanner. There is another important variety of spanners called *near-additive* spanners for which  $\beta \geq 0$  and  $\alpha = 1 + \epsilon$ , for an arbitrarily small  $\epsilon > 0$ . The near-additive spanners are mostly applicable to *unweighted* graphs, even though there are some recent results about weighted near-additive spanners [14].

Spanners are very well-studied from both combinatorial and algorithmic viewpoints. It is well-known that for any parameter  $\kappa = 1, 2, \dots$ , and for any  $n$ -vertex graph  $G = (V, E)$ , there exists a  $(2\kappa - 1)$ -spanner with  $O(n^{1+1/\kappa})$  edges, and this bound is nearly-tight unconditionally, and completely tight under Erdős-Simonovits girth conjecture [38, 4]. The parameter  $2\kappa - 1$  is called the *stretch* parameter of the spanner. Also, for any pair of parameters,  $\epsilon > 0$  and  $\kappa = 1, 2, \dots$ , there exists  $\beta = \beta_{EP} = \beta(\kappa, \epsilon)$ , so that for every  $n$ -vertex undirected graph  $G = (V, E)$ , there exists a  $(1 + \epsilon, \beta)$ -spanner with  $O_{\kappa, \epsilon}(n^{1+1/\kappa})$  edges [18]. The additive term  $\beta = \beta_{EP}$  in [18] behaves as  $\beta(\kappa, \epsilon) \approx \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa}$ , and this bound is the state-of-the-art. A lower bound of  $\Omega(\frac{1}{\epsilon \cdot \log \kappa})^{\log \kappa}$  for it was shown in [1].

Given an  $n$ -vertex weighted undirected graph  $G = (V, E, \omega)$  and two parameters  $\epsilon > 0$  and  $\beta = 1, 2, \dots$ , a graph  $G' = (V, H, \omega')$  is called a  $(1 + \epsilon, \beta)$ -*hopset* of  $G$ , if for every pair of vertices  $u, v \in V$ , we have

$$d_G(u, v) \leq d_{G \cup G'}^{(\beta)}(u, v) \leq (1 + \epsilon) \cdot d_G(u, v).$$

Here  $d_{G \cup G'}^{(\beta)}(u, v)$  stands for  $\beta$ -bounded distance (See Definition 3) between  $u$  and  $v$  in  $G \cup G'$ . The parameter  $\beta$  is called the *hopbound* of the hopset  $G'$ .

Just like spanners, hopsets are a fundamental graph-algorithmic construct. They are extremely useful for computing approximate shortest distances and paths in various computational settings, in which computing shortest paths with a limited number of hops is significantly easier than computing them with no limitation on the number of hops. A partial list of these settings includes streaming, distributed, parallel and centralized dynamic models. Cohen [10] showed that for any undirected weighted  $n$ -vertex graph  $G$ , and parameters  $\epsilon > 0$ ,  $\rho > 0$ , and  $\kappa = 1, 2, \dots$ , there exists a  $(1 + \epsilon, \beta_C)$ -hopset with  $\tilde{O}(n^{1+1/\kappa})$  edges, where  $\beta_C = \left(\frac{\log n}{\epsilon}\right)^{O(\frac{\log \kappa}{\rho})}$ . Elkin and Neiman [16] improved Cohen's result, and constructed hopsets with *constant* hopbound. Specifically, they showed that for any  $\epsilon > 0$ ,  $\kappa = 1, 2, \dots$ , and any  $n$ -vertex weighted undirected graph, there exists a  $(1 + \epsilon, \beta_{EN})$ -hopset with  $\tilde{O}(n^{1+1/\kappa})$  edges, and  $\beta_{EN} = \beta_{EP} \approx \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa}$ . The lower bound of [1],  $\beta = \Omega(\frac{1}{\epsilon \cdot \log \kappa})^{\log \kappa}$  is applicable to hopsets as well. Generally, hopsets (see [10, 28, 16]) are closely related to near-additive spanners. See a recent survey [17] for an extensive discussion on this relationship.

Most of the algorithms for computing (approximate) distances and shortest paths in the streaming setting compute a sparse spanner, and then employ it for computing exact shortest paths and distances in it offline, i.e., in the post-processing, after the stream is over [23, 12, 7, 21, 15, 3, 32, 24, 25]. Specifically, in the dynamic streaming model, algorithms for computing approximately shortest paths (with space  $\tilde{O}(n^{1+1/\kappa})$ , for a parameter  $\kappa = 1, 2, \dots$ ), can be divided into two categories. The algorithms in the first category build a sparse multiplicative  $(2\kappa - 1)$ -spanner, and they provide a *multiplicative* stretch of at least  $2\kappa - 1$  [3, 32, 24, 25]. Moreover, due to existential lower bounds for spanners, this approach is doomed to provide stretch of at least  $\frac{4}{3}\kappa$  [35]. The algorithms in the second category compute *exact single source* shortest paths in *unweighted* graphs, but they employ  $n^{1/2-O(1/\kappa)}$  passes [9, 13]. In the current paper, we partially fill in the gap between these two extremes, and devise a dynamic streaming  $(1 + \epsilon)$ -approximate SSSP algorithm with space  $\tilde{O}(n^{1+1/\kappa})$  that uses  $(\frac{\kappa}{\epsilon})^{\kappa(1+o(1))}$  passes. Table 1 summarizes the existing algorithms for the problem of computing approximate shortest paths in streaming setting on unweighted graphs, and compares them to our results.

■ **Table 1** Prior Work on the Problem.

| Citation     | Model   | Stretch                      | Space                                               | No. of Passes                                                   | Technique                             |
|--------------|---------|------------------------------|-----------------------------------------------------|-----------------------------------------------------------------|---------------------------------------|
| [23]         | Static  | $(2\kappa + 1)$              | $\tilde{O}(n^{1+1/\kappa})$                         | 1                                                               | Multiplicative Spanner                |
| [12, 7]      | Static  | $(2\kappa - 1)$              | $\tilde{O}(n^{1+1/\kappa})$                         | 1                                                               | Multiplicative Spanner                |
| [21, 15]     | Static  | $(1 + \epsilon, \beta_{EN})$ | $\tilde{O}(n^{1+1/\kappa})$                         | $\beta_{EN}$                                                    | Near-Additive Spanner                 |
| [29]         | Static  | $(1 + \epsilon)$             | $n^{1+o(1)} \cdot O(\frac{\log \Lambda}{\epsilon})$ | $2^{O(\sqrt{\log n \log \log n})} = n^{o(1)}$                   | Hopsets                               |
| [16]         | Static  | $(1 + \epsilon)$             | $\tilde{O}(n^{1+\rho})$ , for $\rho > 0$            | $(\frac{\log n}{\epsilon \cdot \rho})^{\frac{1}{\rho}(1+o(1))}$ | Hopsets                               |
| [13]         | Static  | Exact                        | $O(n \cdot p)$ , for $1 \leq p \leq n$              | $O(n/p)$                                                        | Hopsets                               |
| [3]          | Dynamic | $(2\kappa - 1)$              | $\tilde{O}(n^{1+1/\kappa})$                         | $\kappa$                                                        | Multiplicative Spanner                |
| [24]         | Dynamic | $(2\kappa - 1)$              | $\tilde{O}(n^{1+1/\kappa})$                         | $\lfloor \kappa/2 \rfloor + 1$                                  | Multiplicative Spanner                |
| [3]          | Dynamic | $O(\kappa^{\log_2 5})$       | $\tilde{O}(n^{1+1/\kappa})$                         | $O(\log \kappa)$                                                | Multiplicative Spanner                |
| [25]         | Dynamic | $O(\kappa^{\log_2 3})$       | $\tilde{O}(n^{1+1/\kappa})$                         | $O(\log \kappa)$                                                | Multiplicative Spanner                |
| [9]          | Dynamic | Exact                        | $\tilde{O}(n + p^2)$ , for $1 \leq p \leq n$        | $\tilde{O}(n/p)$                                                | Exact BFS                             |
| (This Paper) | Dynamic | $(1 + \epsilon)$             | $\tilde{O}(n^{1+1/\kappa})$                         | Constant (unweighted graphs)<br>and polylog (weighted graphs)   | Near-Additive Spanners<br>and Hopsets |

The algorithms of [23, 12, 7] apply to unweighted graphs, but they can be extended to weighted graphs by running many copies of them in parallel, one for each weight scale. Let  $\Lambda = \Lambda(G)$  denote the *aspect ratio* of the graph, i.e.,  $\Lambda = \frac{\max_{u,v \in V} d_G(u,v)}{\min_{u,v \in V} d_G(u,v)}$ . Also, let  $\epsilon \geq 0$  be a slack parameter. Then by running  $O(\frac{\log \Lambda}{\epsilon})$  copies of the algorithm for unweighted graphs and taking the union of their outputs as the ultimate spanner, one obtains a one-pass static streaming algorithm for  $2(1 + \epsilon)\kappa$ -spanner with  $\tilde{O}(n^{1+\frac{1}{\kappa}} \cdot (\log \Lambda)/\epsilon)$  edges [19].

In [21], the authors devised static streaming algorithms for building  $(1 + \epsilon, \beta_{EZ})$ -spanners where,  $\beta_{EZ} = \beta_{EZ}(\epsilon, \rho, \kappa) = \left(\frac{\log \kappa}{\epsilon \cdot \rho}\right)^{O(\frac{\log \kappa}{\rho})}$ , for any parameters  $\epsilon, \rho > 0$  and  $\kappa = 1, 2, \dots$ . This result was improved in [15], where a static streaming algorithm with similar properties, but with  $\beta = \beta_{EN} = \left(\frac{\log \kappa \rho + 1/\rho}{\epsilon}\right)^{\log \kappa \rho + 1/\rho}$  was devised. The algorithms of [21, 15] directly give rise to  $\beta$ -pass static streaming algorithms with space  $\tilde{O}(n^{1+\rho})$  for  $(1 + \epsilon, \beta)$ -APASP (*All Pairs Almost Shortest Paths*) in unweighted graphs where  $\beta(\rho) \approx (1/\rho)^{(1/\rho)(1+o(1))}$ . They can also be used for producing purely multiplicative  $(1 + \epsilon)$ -approximate shortest paths and distances in  $O(\beta/\epsilon)$  passes and  $\tilde{O}(n^{1+\rho})$  space from up to  $n^{\rho(1-o(1))}$  designated sources to all other vertices. There are also a number of additional *not* spanner-based static streaming algorithms for computing approximate shortest paths. Henzinger et al. [29] and Elkin and Neiman [16] devised  $(1 + \epsilon)$ -approximate *single-source* shortest paths (henceforth, SSSP) algorithms for weighted graphs, that are based on *hopsets*.

Recently Chang et al. [9] devised a *dynamic streaming* algorithm for this problem in *unweighted* graphs. Their algorithm uses  $\tilde{O}(n/p)$  passes (for parameter  $1 \leq p \leq n$  as above) and space  $\tilde{O}(n + p^2)$  for the SSSP problem, and space  $\tilde{O}(|S|n + p^2)$  for the  $S \times V$  approximate shortest path computation.

Ahn et al. [3] devised the first *dynamic streaming* algorithm for computing approximate distances. Their algorithm computes a  $(2\kappa - 1)$ -spanner (for any  $\kappa = 1, 2, \dots$ ) with  $\tilde{O}(n^{1+1/\kappa})$  edges (and the same space complexity) in  $\kappa$  passes over the stream. This bound was recently improved by [24]. Their algorithm computes a spanner with the same properties using  $\lceil \kappa/2 \rceil + 1$  passes. Ahn et al. [3] also devised an  $O(\log \kappa)$ -pass algorithm for building  $O(\kappa^{\log_2 5})$ -spanner with size and space complexity  $\tilde{O}(n^{1+1/\kappa})$ . This bound was recently improved by [25], whose algorithm produces  $O(\kappa^{\log_2 3})$ -spanner with the same pass and space complexities, and the same size.

Another dynamic streaming algorithm was devised by Kapralov and Woodruff [32]. It produces a  $(2^\kappa - 1)$ -spanner with  $\tilde{O}(n^{1+1/\kappa})$  edges (and space usage) in two passes. Kapralov et al. [25] improved the stretch parameter of the spanner to  $2^{\frac{\kappa+3}{2}} - 3$ , with all other parameters the same as in the results of [32]. Kapralov et al. [25] also devised a general tradeoff in which the number of passes can be between 2 and  $\kappa$ , and the stretch of the spanner decreases gracefully from exponential in  $\kappa$  (where the number of passes is 2) to  $2\kappa - 1$  (when the number of passes is  $\kappa$ ). They have also devised a single pass algorithm with stretch  $\tilde{O}(n^{\frac{2}{3}(1-1/\kappa)})$ . As was mentioned above, most of these spanner-based algorithms provide a solution for  $(2\kappa - 1)$ -APASP for unweighted graphs with space  $\tilde{O}(n^{1+1/\kappa})$  and the number of passes equal to that of the spanner-construction algorithm. Like their static streaming counterparts [23, 12, 7], they can be extended to weighted graphs, at the price of increasing their stretch by a factor of  $1 + \epsilon$  (for an arbitrarily small parameter  $\epsilon > 0$ ), and their space usage by a factor of  $O\left(\frac{\log \Delta}{\epsilon}\right)$ .

## 1.2 Our Results

In the current paper, we present the first dynamic streaming algorithm for SSSP with stretch  $1 + \epsilon$ , space  $\tilde{O}(n^{1+1/\kappa})$ , and *constant* (as long as  $\epsilon$  and  $\kappa$  are constant) number of passes for unweighted graphs. For weighted graphs, our number of passes is *polylogarithmic* in  $n$ . Specifically, the number of passes of our SSSP algorithm is  $\left(\frac{\kappa}{\epsilon}\right)^{\kappa(1+o(1))}$  for unweighted graphs, and  $\left(\frac{(\log n)\kappa}{\epsilon}\right)^{\kappa(1+o(1))}$  for weighted ones. Moreover, within the same complexity bounds, our algorithm can compute  $(1 + \epsilon)$ -approximate  $S \times V$  shortest paths from  $|S| = n^{1/\kappa}$  designated sources. Moreover, in *unweighted* graphs, *all* pairs almost shortest paths with stretch  $(1 + \epsilon, \left(\frac{\kappa}{\epsilon}\right)^\kappa)$  can also be computed within the same space and number of passes. (That is, paths and distances with multiplicative stretch  $1 + \epsilon$  and additive stretch  $\left(\frac{\kappa}{\epsilon}\right)^\kappa$ .) Note that our multiplicative stretch  $(1 + \epsilon)$  is dramatically better than  $(2\kappa - 1)$ , exhibited by algorithms based on multiplicative spanners [3, 32, 24, 25], but this comes at a price of at least exponential increase in the number of passes. Nevertheless, our number of passes is *independent of  $n$* , for unweighted graphs, and depends only polylogarithmically on  $n$  for weighted ones.

## 1.3 Technical Overview

We devise algorithms for building near-exact hopsets and near-additive spanners in dynamic streaming model. These structures help us compute almost shortest paths in the input graph. The following theorem summarizes the resource usage and properties of our hopset construction.

► **Theorem 1** (Theorem 9 in Section 3.4). *For any  $n$ -vertex graph  $G(V, E, \omega)$  with aspect ratio  $\Lambda$ ,  $2 \leq \kappa \leq (\log n)/4$ ,  $1/\kappa \leq \rho \leq 1/2$  and  $0 < \epsilon < 1$ , our dynamic streaming algorithm computes w.h.p. a  $(1 + \epsilon, \beta)$  hopset  $H$  with expected size  $O(n^{1+1/\kappa} \cdot \log n)$  and the hopbound  $\beta$  given by,  $\beta = O\left(\frac{(\log \kappa \rho + 1/\rho) \log n}{\epsilon}\right)^{\log \kappa \rho + 1/\rho}$ . It does so by making  $O(\beta \cdot (\log \kappa \rho + 1/\rho))$  passes through the stream and using  $O(n \cdot \log^3 n \cdot \log \Lambda)$  bits of space in the first pass and  $O(\frac{\beta}{\epsilon} \cdot \log^2 1/\epsilon \cdot n^{1+\rho} \cdot \log^5 n)$  bits of space (respectively  $O(\frac{\beta^2}{\epsilon} \cdot \log^2 1/\epsilon \cdot n^{1+\rho} \cdot \log^5 n)$  bits of space for path-reporting hopset) in each of the subsequent passes.*

The hopset is then used (in Section 4) to compute almost shortest paths in weighted graphs.

In the extended version, we also show a similar algorithm for constructing near-additive spanners. This result is summarized in the following theorem.

► **Theorem 2.** *For any unweighted graph  $G(V, E)$  on  $n$  vertices, parameters  $0 < \epsilon < 1$ ,  $\kappa \geq 2$ , and  $\rho > 0$ , our dynamic streaming algorithm computes a  $(1 + \epsilon, \beta)$ -spanner with  $O_{\epsilon, \kappa, \rho}(n^{1+1/\kappa})$  edges, in  $O(\beta)$  passes using  $O(n^{1+\rho} \log^4 n)$  space with high probability, where  $\beta$  is given by,  $\beta = \left(\frac{\log \kappa \rho + 1/\rho}{\epsilon}\right)^{\log \kappa \rho + 1/\rho}$ .*

Our algorithms for spanner and hopset construction extend the results of [15, 16] from the static streaming setting to dynamic streaming one. The algorithms of [15, 16], like their predecessor, the algorithm of [18], are based on the superclustering-and-interconnection (henceforth, SAI) approach. Our algorithms in the current paper also fall into this framework. Algorithms that follow the SAI approach proceed in phases, and in each phase they maintain a partial partition of the vertex set  $V$  of the graph. Some of the clusters of  $G$  are selected to create superclusters around them. This is the superclustering step. Clusters that are not superclustered into these superclusters are then *interconnected* with their nearby clusters. The main challenge in implementing this scheme in the dynamic streaming setting is in the interconnection step. Indeed, the superclustering step requires a single and rather shallow BFS exploration, and implementing depth- $d$  BFS in unweighted graphs in  $d$  passes over the dynamic stream can be done in near-linear space (See, e.g., [3, 9]). For the weighted graphs, we devise a routine for performing an approximate Bellman-Ford exploration up to a given hop-depth  $d$ , using  $d$  passes and  $\tilde{O}(n)$  space.

On the other hand, the interconnection step requires implementing simultaneous BFS (Bellman-Ford for the weighted case) distance explorations originated at multiple sources. A crucial property that enabled [15, 16] to implement it in the static streaming setting is that one can argue that with high probability, not too many distance explorations traverse any particular vertex. Let us denote by  $N$ , an upper bound on the number of explorations (traversing any particular vertex). In the dynamic streaming setting, however, at any point of the stream, there may well be much more than  $N$  explorations that traverse a specific vertex  $v \in V$ , based on the stream of updates observed so far. Storing data about all these explorations would make the space requirement of the algorithm prohibitively large.

To resolve this issue (and a number of related similar issues), we incorporate a *sparse recovery* routine into our algorithms. Sparse recovery is a fundamental and well-studied primitive in the dynamic streaming setting [26, 11, 30, 5]. It is defined for an input which is a stream of (positive and negative) updates to an  $n$ -dimensional vector  $\vec{a} = (a_1, a_2, \dots, a_n)$ . In the *strict* turnstile setting, which is sufficient for our application, ultimately each coordinate  $a_i$  (i.e., at the end of the stream) is non-negative, even though negative updates are allowed and intermediate values of coordinates may be negative. In the *general* turnstile model coordinates of the vector  $\vec{a}$  may be negative at the end of the stream as well. The *support* of  $\vec{a}$ , denoted  $\text{supp}(\vec{a})$ , is defined as the set of its non-zero coordinates. For a parameter

$s$ , an  $s$ -sparse recovery routine returns the vector  $\vec{a}$ , if  $|\text{supp}(\vec{a})| \leq s$ , and returns failure otherwise. (It is typically also allowed to return failure with some small probability  $\delta > 0$ , given to the routine as a parameter, even if  $|\text{supp}(\vec{a})| \leq s$ .)

Most of sparse recovery routines are based on 1-sparse recovery, i.e., the case  $s = 1$ . The first 1-sparse recovery algorithm was devised by Ganguly [26], and it applies to the strict turnstile setting. The space requirement of the algorithm of [26] is  $O(\log n)$ . The result was later extended to the general turnstile setting by Cormode and Firmani [11] (See also [37]). We devise an alternative streaming algorithm for this basic task in the strict turnstile setting. The space complexity of our algorithm is  $O(\log n)$ , like that of [26]. The processing time-per-item of [26]’s algorithm is however  $O(1)$ , instead of  $\text{polylog}(n)$  of our algorithm.<sup>1</sup> Nevertheless, we believe that our new algorithm for this task is of independent interest.

In the current paper we analyze our algorithm in terms of the aspect ratio  $\Lambda$  of the input graph, given by  $\Lambda = \frac{\max_{u,v \in V} d_G(u,v)}{\min_{u,v \in V} d_G(u,v)}$ . (All dependencies are polylogarithmic in  $\Lambda$ .) In the extended version [20], we also show that [34]’s weight reduction (see also [16]) can be implemented in the dynamic streaming model. As a result, we replace all appearances of  $\log \Lambda$  in the hopset’s size, hopbound and number of passes of our construction by  $O(\log n)$ . However, the space complexity of our algorithm still mildly depends on  $\log \Lambda$ . Specifically, it is  $\tilde{O}(n^{1+\rho}) + \tilde{O}(n) \cdot \log \Lambda$ . In all existing dynamic streaming algorithms for computing multiplicative spanners or computing approximate shortest paths in weighted graphs [3, 32, 24, 25], both the spanner’s size and the space requirements are  $\tilde{O}(n^{1+1/\kappa} \cdot \log \Lambda)$ . But using our weight reduction in conjunction with these algorithms, one can produce spanners of size  $\tilde{O}(n^{1+1/\kappa})$  (without dependence on  $\log \Lambda$ ), using space  $\tilde{O}(n^{1+1/\kappa}) + \tilde{O}(n) \cdot \log \Lambda$ . However, the number of passes increases by an additive term of 1. Completely eliminating the dependence on  $\log \Lambda$  from these results is left as an open problem.

## 1.4 Outline

The rest of the paper is organized as follows. Section 2 provides necessary definitions and concepts. Section 3 presents an algorithm for constructing hopsets with constant hopbound. All the missing proofs and a more thorough analysis of our hopset construction are available in the extended version [20]. Section 4 shows how we use the algorithm of Section 3 to compute  $(1 + \epsilon)$ -approximate shortest paths in weighted graphs. The spanner construction algorithm is very similar to the hopset construction algorithm. The spanner construction algorithm and details on its usage in computing approximate shortest paths in unweighted graphs are available in the extended version [20] of this paper. The subroutine for performing a limited depth Bellman-Ford exploration in the input graph is described in Appendix B. Appendix appears after the references.

## 2 Preliminaries

### 2.1 Streaming Model

In the *dynamic streaming* model of computation, the set of vertices  $V$  of the input graph is known in advance and the edge set  $E$  is revealed one at a time. The edges can be added as well as removed. For a weighted input graph, the stream  $S$  arrives as a sequence of

<sup>1</sup> If the algorithm knows in advance the dimension  $n$  of the vector  $\vec{a}$  and is allowed to compute during preprocessing, before seeing the stream, a table of size  $n$ , then our algorithm can also have  $O(1)$  processing time per update. This scenario occurs in dynamic streaming graph algorithms, including those discussed in the current paper.

edge updates  $S = \langle s_1, s_2, \dots \rangle$ , where  $s_t = (e_t, eSign_t, eWeight_t)$ , where  $e_t$  is the edge being updated and  $eWeight_t$  is its weight. The  $eSign_t \in \{+1, -1\}$  value of an update indicates whether the edge  $e_t$  is to be added or removed. A value of  $+1$  indicates addition and a value of  $-1$  indicates removal.

For a weighted undirected graph  $G = (V, E, \omega)$ , we assume that the edge weights are scaled so that the minimum edge weight is 1. Let  $maxW$  denote the maximum edge weight  $\omega(e)$ ,  $e \in E$ . For a non-edge  $(u, v) \notin E$ , we define  $\omega((u, v)) = \infty$ .

► **Definition 3.** Given a weighted graph  $G(V, E, \omega)$ , a positive integer parameter  $t$ , and a pair  $u, v \in V$  of distinct vertices, a  **$t$ -bounded  $u$ - $v$  path** in  $G$  is a path between  $u$  and  $v$  that contains no more than  $t$  edges (also known as hops), and  **$t$ -bounded distance** between  $u$  and  $v$  in  $G$  denoted  $d_G^{(t)}(u, v)$  is the length of a shortest  $t$ -bounded  $u$ - $v$  path in  $G$ .

## 2.2 Samplers, Hash Functions and Vertex Encodings

The main technical tool in our algorithms is a space-efficient sampling technique which enables us to sample a single vertex or a single edge from an appropriate subset of the vertex set or the edge set of the input graph, respectively. Algorithms for sampling from a dynamic stream are inherently randomized and often use hash functions as a source of randomness. Appendix A is devoted to hash functions.

## 2.3 Vertex Encodings

We assume that the vertices have unique IDs from the set  $\{1, \dots, n\}$ . The maximum possible ID (which is  $n$ ) of a vertex in the graph is denoted by  $maxVID$ . The binary representation of the ID of a vertex  $v$  can be obtained by performing a name operation  $name(v)$ . For an integer  $k \geq 1$ ,  $[k]$  denotes the set  $\{1, 2, \dots, k\}$ .

We also need standard definitions of convex combination, convex hull and a convexly independent set. We will use the following CIS-based encoding for the vertices of the graph: **CIS Encoding Scheme  $\nu$** : We assign a unique code in  $\mathbb{Z}^2$  to every vertex  $v \in V$ . The encoding scheme works by generating a set of  $n$  convexly independent integer vectors in  $\mathbb{Z}^2$ . Specifically, our encoding scheme uses as its range, the extremal points of the convex hull of  $Ball_2(R) \cap \mathbb{Z}^2$ , where  $Ball_2(R)$  is a two-dimensional disc of radius  $R$  centered at origin. A classical result by [31], later refined by [6], states that the number of extremal points of the convex hull of a set of integer points of a disc of radius  $R$  is  $\Theta(R^{2/3})$ . We set  $R = \Theta(n^{3/2})$  to allow for all the possible  $n = \Theta(R^{2/3})$  vertices to be encoded in  $O(\log n)$  bits. The encoding of any vertex  $v$  can be obtained by performing an encoding operation denoted by  $\nu(v)$ .

We prove the following lemma in extended version which will be useful in Section 3.2 to detect if the sampling procedure succeeded in sampling exactly one vertex from a desired subset of the set  $V$ .

► **Lemma 4.** Let  $c_1, c_2, \dots, c_n$  be non-negative integer coefficients of a linear combination of a set  $P = \{p_1, p_2, \dots, p_n\}$  of  $n$  convexly independent points in  $\mathbb{Z}^2$  such that  $\frac{\sum_{j=1}^n c_j \cdot p_j}{\sum_{j=1}^n c_j} = p_i$ , for some  $p_i \in P$ . Then  $c_j = 0$  for every  $j \neq i$ , and  $c_i \neq 0$ .

## 3 Hopsets with Constant Hopbound in Dynamic Streaming Model

We adapt the insertion-only streaming algorithm of [16] for hopset construction to work in the dynamic streaming setting. The algorithm is based on the SAI (Superclustering and Interconnection) approach. (See [16] for more details.) The main ingredient of both

the superclustering and interconnection steps is a set of Bellman-Ford explorations (*B-F explorations*, henceforth) up to a given distance in the input graph from a set of chosen vertices. The insertion-only streaming algorithm of [16] identifies all the edges spanned by  $\Theta(\beta)$  iterations of certain B-F explorations up to a distance  $\delta$  from a set of chosen vertices, by making  $\Theta(\beta)$  passes through the stream. Other parts of the hopset construction, such as identifying the vertices of the graph from which to perform B-F explorations and subsequently adding edges corresponding to certain paths traversed by these explorations to the hopset, are performed offline.

We devise a technique to perform a given number of iterations of a B-F exploration from a set of chosen vertices and up to a given distance in the graph in the dynamic streaming setting, and as in [16], perform the rest of the work offline. The difference however is that in the dynamic streaming setting, we do not perform an exact and deterministic B-F exploration (as in [16]). A randomized algorithm for performing an approximate B-F exploration originated at a subset of source vertices in a weighted graph, that succeeds whp, is described in Appendix B. We use this algorithm as a subroutine in the superclustering step of our main algorithm. The interconnection step is more challenging and involves performing multiple simultaneous B-F explorations in a weighted graph, each from a separate source vertex. Here, each vertex in the graph needs to identify all the B-F explorations it is a part of, and to find its (approximate) distance to the source of each such exploration. Due to the dynamic nature of the stream, a given vertex may find itself on a lot more explorations than it finally ends up belonging to. This can be dealt with by combining a delicate encoding/decoding scheme for the IDs of exploration sources with a space-efficient sampling technique.

In the following section, we provide an overview of our hopset construction algorithm.

### 3.1 Overview

Our hopset construction algorithm takes as input an  $n$ -vertex weighted undirected graph  $G = (V, E, \omega)$  with aspect ratio  $\Lambda$ , and parameters  $0 < \epsilon' < 1/10$ ,  $\kappa = 1, 2, \dots$  and  $1/\kappa < \rho < 1/2$ , and produces as output a  $(1 + \epsilon', \beta')$ -hopset of  $G$ . The hopbound parameter  $\beta'$  is a function of  $\epsilon'$ ,  $\Lambda$ ,  $\kappa$ ,  $\rho$  and is given by  $\beta' = O\left(\frac{\log \Lambda}{\epsilon'} \cdot (\log \kappa \rho + 1/\rho)\right)^{\log \kappa \rho + 1/\rho}$ .

Let  $k = 0, 1, \dots, \lceil \log \Lambda \rceil - 1$ . Given two parameters  $\epsilon > 0$  and  $\beta = 1, 2, \dots$ , a set of weighted edges  $H_k$  on the vertex set  $V$  of the input graph is said to be a  $(1 + \epsilon, \beta)$ -hopset for the scale  $k$  or a *single-scale hopset*, if for every pair of vertices  $u, v \in V$  with  $d_G(u, v) \in (2^k, 2^{k+1}]$  we have that

$$d_G(u, v) \leq d_{G_k}^{(\beta)}(u, v) \leq (1 + \epsilon) \cdot d_G(u, v),$$

where  $G_k = (V, E \cup H_k, \omega_k)$  and  $\omega_k(u, v) = \min\{\omega(u, v), \omega_{H_k}(u, v)\}$ , for every edge  $(u, v) \in E \cup H_k$ . Let  $\epsilon > 0$  be a parameter that will be determined later in the sequel. (See Section 3.3.) Set also  $\ell = \lfloor \log \kappa \rho \rfloor + \lceil \frac{\kappa+1}{\kappa \rho} \rceil - 1$ . Let  $\beta = (1/\epsilon)^\ell$ .

The algorithm constructs a separate  $(1 + \epsilon, \beta)$ -hopset  $H_k$  for every scale  $(2^0, 2^1], (2^1, 2^2], \dots, (2^{\lceil \log \Lambda \rceil - 1}, 2^{\lceil \log \Lambda \rceil}]$  one after another. For  $k \leq \lceil \log \beta \rceil - 1$ , we set  $H_k = \phi$ . We can do so because for such a  $k$ , it holds that  $2^{k+1} \leq \beta$ , and for every pair of vertices  $u, v$  with  $d_G(u, v) \leq 2^{k+1}$ , the original graph  $G$  itself contains a shortest path between  $u$  and  $v$  that contains at most  $\beta$  edges. (We remark that after rescaling, we will have  $\beta' = \beta$ . See Section 3.3.) In other words,  $d_G(u, v) = d_G^{(\beta)}(u, v)$ . Denote  $k_0 = \lfloor \log \beta \rfloor$  and  $k_\Lambda = \lceil \log \Lambda \rceil - 1$ . We construct a hopset  $H_k$  for every  $k \in [k_0, k_\Lambda]$ .



During the construction of the hopset  $H_k$  for some  $k \geq k_0$ , we need to perform explorations from certain vertices in  $V$  up to distance  $\delta \leq 2^{k+1}$  in  $G$ . An exploration up to a given distance from a certain vertex in  $G$  may involve some paths with up to  $n - 1$  hops. This can take up to  $O(n)$  passes through the stream. We overcome this problem by using the hopset edges  $H^{(k-1)} = \bigcup_{k_0 \leq j \leq k-1} H_j$  for constructing hopset  $H_k$ . The hopset  $H_k$  has to take care of all pairs of vertices  $u, v$  with  $d_G(u, v) \in (2^k, 2^{k+1}]$ , whereas the edges in  $E \cup H^{(k-1)}$  provide a  $(1 + \epsilon_{k-1})$ -approximate shortest path with up to  $\beta$  hops, for every pair  $u, v$  with  $d_G(u, v) \leq 2^k$ . The value of  $\epsilon_{k-1}$  will be specified later in the sequel. (See Section 3.3.) Denote by  $G^{(k-1)}$  the graph obtained by adding the edge set  $H^{(k-1)}$  to the input graph  $G$ . Instead of conducting explorations from a subset  $S \subseteq V$  up to distance  $\delta \leq 2^{k+1}$  in the input graph  $G$ , we perform  $2\beta + 1$  iterations of B-F algorithm on the graph  $G^{(k-1)}$  up to distance  $(1 + \epsilon_{k-1}) \cdot \delta$ . The following lemma from [16] provides a justification for this approach.

► **Lemma 5** ([16]). *For  $u, v \in V$  with  $d_G(u, v) \leq 2^{k+1}$ , the following holds:*

$$d_{G^{(k-1)}}^{(2\beta+1)}(u, v) \leq (1 + \epsilon_{k-1}) \cdot d_G(u, v) \quad (1)$$

### 3.2 Constructing $H_k$

We now proceed to the construction of the hopset  $H_k$  for the scale  $(2^k, 2^{k+1}]$ , for some  $k \in [k_0, k_\Lambda]$ . We start by initializing the hopset  $H_k$  as an empty set and proceed in phases  $0, 1, \dots, \ell$ . All phases of our algorithm except for the last one consist of two steps, a superclustering step and an interconnection step. In the last phase, we go directly to the interconnection step. Throughout the algorithm, we build clusters of nearby vertices. The input to phase  $i \in [0, \ell]$  is a set of clusters  $P_i$ , a distance threshold parameter  $\delta_i$  and a degree parameter  $\deg_i$ . For phase 0, the input  $P_0$  is a partition of the vertex set  $V$  into singleton clusters. Every cluster created by our algorithm has a designated *center* vertex. We denote by  $r_C$  the center of cluster  $C$ . In particular, each singleton cluster  $C = \{v\}$  is centered around  $v$ . For a cluster  $C$ , we define  $\text{Rad}(C) = \max\{d_{G^{(k-1)}}(r_C, v) \mid v \in C\}$ . For a set of clusters  $P_i$ ,  $\text{Rad}(P_i) = \max_{C \in P_i} \{\text{Rad}(C)\}$ .

The degree threshold parameter  $\deg_i$  of phase  $i$  is used to define the sampling probability with which the centers of clusters in  $P_i$  are selected to grow superclusters around them. We partition the phases of the algorithm into two stages based on how the degree parameter grows in each stage. The two stages are the *exponential growth stage* and the *fixed growth stage*. In the *exponential growth stage*, which consists of phases  $0, 1, \dots, i_0 = \log\lceil \kappa\rho \rceil$ , we set  $\deg_i = n^{\frac{2^i}{\kappa}}$ . In the *fixed growth stage*, which consists of phases  $i_0 + 1, i_0 + 2, \dots, i_1 = i_0 + \lceil \frac{\kappa+1}{\kappa\rho} \rceil$ , we set  $\deg_i = n^\rho$ . Observe that for every index  $i$ , we have  $\deg_i \leq n^\rho$ .

The distance threshold parameter increases by a factor of  $1/\epsilon$  in every phase. The sequence of the distance threshold parameters for the centralized construction as defined in [16] is given by  $\alpha = \alpha^{(k)} = \epsilon^\ell \cdot 2^{k+1}$ ,  $\delta_i = \alpha(1/\epsilon)^i + 4R_i$ , where  $R_0 = 0$  and  $R_{i+1} = R_i + \delta_i = \alpha(1/\epsilon)^i + 5R_i$  for  $i \geq 0$ . Here  $\alpha$  can be perceived as a unit of distance. To adjust for the fact that explorations are performed on the graph  $G^{(k-1)}$ , and not on the input graph  $G$ , we multiply all the distance thresholds  $\delta_i$  by a factor of  $1 + \epsilon_{k-1}$ , the stretch guarantee of the graph  $G^{(k-1)}$ . We further modify this sequence to account for the fact that our B-F explorations in the dynamic stream are not exact and incur a multiplicative error. Throughout the construction of  $H_k$ , we set the multiplicative error of every approximate B-F Exploration we perform to  $1 + \chi$ , for a parameter  $\chi > 0$  which will be determined later (in Section 3.3). Therefore we multiply all the distance thresholds by a factor of  $1 + \chi$ . We define  $R'_i = (1 + \chi) \cdot (1 + \epsilon_{k-1})R_i$  and  $\delta'_i = (1 + \chi) \cdot (1 + \epsilon_{k-1})\delta_i$  for every  $i \in [0, \ell]$ . In the centralized setting,  $R_i$  serves as an upper bound on the radii of the input clusters of phase  $i$ . As a result of rescaling,  $R'_i$  becomes the new upper bound on the radii of input clusters of phase  $i$ .

**Superclustering.** The phase  $i$  begins by sampling each cluster  $C \in P_i$  independently at random with probability  $1/\deg_i$ . Let  $S_i$  denote the set of sampled clusters. We now have to conduct (approximate) distance exploration up to depth  $\delta'_i$  in  $G^{(k-1)}$  rooted at the set  $CS_i = \bigcup_{C \in S_i} \{r_C\}$ . By Lemma 5, this can be achieved by  $2\beta + 1$  iterations of B-F algorithm on the graph  $G^{(k-1)}$ . For this, we invoke the approximate B-F exploration algorithm of Appendix B on graph  $G^{(k-1)}$  with set  $CS_i$  as the set  $S$  of source vertices and parameters  $\eta = 2\beta + 1$ ,  $\zeta = \chi$ . We slightly modify the algorithm of Appendix B and then invoke the modified version. In the modified version, at the end of each pass through the stream, for every vertex  $v \in V$ , we scan through the edges incident to  $v$  in the set  $H^{(k-1)}$  and update its distance estimate  $\hat{d}(v)$  as:

$$\hat{d}(v) = \min\{\hat{d}(v), \min_{(v,w) \in H^{(k-1)}} \{\hat{d}(w) + \omega_{H^{(k-1)}}(v, w)\}\}.$$

The parent of  $v$ ,  $\hat{p}(v)$  is also updated accordingly. Note that this modification does not affect the space complexity, stretch guarantee or the success probability of the algorithm of Appendix B. This provides us with a  $(1 + \chi)$ -approximation of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, CS_i)$ , for all  $v \in V$ . Hence, by Theorem 14, an invocation of modified version of approximate B-F algorithm of Appendix B during the the superclustering step of phase  $i$  generates whp, an approximate B-F exploration of the graph  $G^{(k-1)}$ , rooted at the set  $CS_i \subseteq V$  in  $2\beta + 1$  passes. It outputs for every  $v \in V$  an estimate  $\hat{d}(v)$  of its distance to set  $CS_i$  such that:

$$d_{G^{(k-1)}}^{(2\beta+1)}(v, CS_i) \leq \hat{d}(v) \leq (1 + \chi) \cdot d_{G^{(k-1)}}^{(2\beta+1)}(v, CS_i). \quad (2)$$

Moreover, the set of parent variables  $\hat{p}(v)$  of every  $v \in V$  with  $\hat{d}(v) < \infty$  spans a forest  $F$  of  $G^{(k-1)}$  rooted at the set of sampled centers  $CS_i$ .

For every cluster center  $r_{C'}$ ,  $C' \in P_i \setminus S_i$ , such that  $\hat{d}(r_{C'}) \leq \delta'_i$ , the algorithm adds an edge  $(r_C, r_{C'})$  of weight  $\hat{d}(r_{C'})$  to the hopset  $H_k$ , where  $r_C$  is the root of the tree in  $F$  to which  $r_{C'}$  belongs. We also create a supercluster rooted at  $r_C$  which contains all the vertices of  $C'$  as above. Note that if  $d_G(r_C, r_{C'}) \leq \delta_i$ , then by equations (1) and (2),  $\hat{d}(r_{C'}) \leq (1 + \chi) \cdot (1 + \epsilon_{k-1})d_G(r_C, r_{C'}) = \delta'_i$ . Therefore, edge  $(r_C, r_{C'})$  will be added to the hopset and the cluster  $C'$  will be superclustered into a cluster centered at  $r_C$ .

**Interconnection.** Next we describe the interconnection step of each phase  $i \in \{0, 1, \dots, \ell\}$ . Let  $U_i$  be the set of clusters of  $P_i$  that were not superclustered in phase  $i$ . Let  $CU_i = \bigcup_{C \in U_i} \{r_C\}$ . In the interconnection step of phase  $i \geq 0$ , we want to connect every cluster  $C \in U_i$  to every other cluster  $C' \in U_i$  that is close to it. To do this, we want to perform  $2\beta + 1$  iterations of a  $(1 + \chi)$ -approximate B-F exploration from every cluster center  $r_C \in CU_i$  separately in  $G^{(k-1)}$ . These explorations are, however, conducted to a bounded depth (in terms of number of hops), and to bounded distance. Specifically, the hop-depth of these explorations will be at most  $2\beta + 1$ , while the distance to which they are conducted is roughly  $\delta_i/2$ . For every cluster center  $r_{C'}$ ,  $C' \in U_i$  within distance  $\delta_i/2$  from another center  $r_C$  in  $G$ , we want to add an edge  $e = (r_C, r_{C'})$  of weight at most  $(1 + \chi) \cdot d_{G^{(k-1)}}^{(2\beta+1)}(r_C, r_{C'})$  to the hopset  $H_k$ . To do so, we turn to the stream to find an estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, r_C)$  for every  $v \in V$  and every center  $r_C \in U_i$ . We cannot afford to invoke the algorithm of Appendix B multiple times in parallel to conduct a separate exploration from every center  $r_C$  in  $CU_i$ , due to space constraints. As shown in [16] (See Lemmas 3.2 and 3.3 of [16]), the following lemma holds in the interconnection step of our (single-scale) hopset construction:

► **Lemma 6 ([15]).** *For any vertex  $v \in V$ , the expected number of explorations that visit  $v$  in the interconnection step of phase  $i$  is at most  $\deg_i$ . Moreover, for any constant  $c'_1$ , every vertex  $v$  is explored by at most  $c'_1 \cdot \ln n \cdot \deg_i$  explorations in phase  $i$  with probability at least  $1 - 1/n^{c'_1-1}$ .*

Specifically, if one conducts B-F explorations to depth at most  $\delta'_i/2$  in  $G^{(k-1)}$  to hop-depth at most  $2\beta + 1$ , then, whp, every vertex is traversed by at most  $O(\deg_i \ln n)$  explorations. Therefore, we devise a randomized technique to efficiently identify for every  $v \in V$ , the sources of all the explorations it gets visited by in phase  $i$ . Moreover, for every vertex  $v \in V$  with a non-empty subset  $U_i^v \subseteq U_i$  of explorations that visit  $v$ , we find for every cluster  $C \in U_i^v$ , an estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, r_C)$ .

Throughout the interconnection step of phase  $i$ , we maintain for every vertex  $v \in V$ , a set  $LCurrent_v$  (called *estimates list of  $v$* ) of sources of B-F explorations that visited  $v$  so far. Each element of  $LCurrent_v$  is a tuple  $(s, \hat{d}(v, s))$ , where  $s$  is the center of some cluster in  $U_i$ , and  $\hat{d}(v, s)$  is the current estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$ . For any center  $s' \in CU_i$ , for which we do not yet have a tuple in  $LCurrent_v$ ,  $\hat{d}(v, s')$  is implicitly defined as  $\infty$ . Initially, the estimates lists of all the vertices are empty, except for the centers of clusters in  $U_i$ . The estimates list of every center  $r_C \in CU_i$  is initialized with a single element  $(r_C, 0)$  in it. The interconnection step of phase  $i$  is carried out in  $2\beta + 1$  sub-phases. Next, we describe the purpose of each of the  $2\beta + 1$  sub-phases of the interconnection step and the way they are carried out.

**Sub-phase  $p$  of the interconnection step.** Denote  $\zeta' = \frac{\chi}{2 \cdot (2\beta+1)}$ . Our goal is to ensure that by the end of sub-phase  $p$ , for every vertex  $v \in V$  and every exploration source  $s \in CU_i$  with a  $p$ -bounded path to  $v$  in  $G^{(k-1)}$ , there is a tuple  $(s, \hat{d}(v, s))$  in the estimates list  $LCurrent_v$  such that

$$d_{G^{(k-1)}}^{(p)}(v, s) \leq \hat{d}(v, s) \leq (1 + \zeta')^p \cdot d_{G^{(k-1)}}^{(p)}(v, s).$$

To accomplish this, in every sub-phase  $p$ , we search for every vertex  $v \in V$ , a *better* (smaller than the current value of  $\hat{d}(v, s)$ ) estimate (if exists) of its  $(2\beta + 1)$ -bounded distance to every source  $s \in CU_i$ , by keeping track of edges  $e = (u, v)$  incident to  $v$  in  $G^{(k-1)}$ . In each of the  $2\beta + 1$  sub-phases, we make two passes through the stream. For a given vertex  $v \in V$ , an exploration source  $s \in CU_i$  is called an *update candidate* of  $v$  in sub-phase  $p$ , if a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  is available in sub-phase  $p$  through some edge  $e = (u, v)$  on the stream. (Recall that the current estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s')$  for some source  $s' \in CU_i$  for which we do not yet have an entry in  $LCurrent_v$  is  $\infty$ .) We go through the edge set  $H^{(k-1)}$  offline at the end of every sub-phase and update all our estimates lists with the best available estimates in  $H^{(k-1)}$ .

In the first pass of sub-phase  $p$ , we identify for every  $v \in V$ , all of  $v$ 's update candidates in sub-phase  $p$ . All of these update candidates are added to a list called the *update list* of  $v$ , denoted  $LUpdate_v$ . Each element of  $LUpdate_v$  is a tuple  $(s, range, r)$ , where  $s$  is the ID of an exploration source in  $CU_i$  for which a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  is available,  $range$  is the distance range  $I = (low, high]$  in which the better estimate is available, and  $r$  is the number of vertices  $u \in \Gamma_G(v)$ , such that  $\hat{d}(u, s) + \omega(u, v) \in range$ . The second pass of sub-phase  $p$  uses the update list of every vertex  $v \in V$  to find a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$ , for every update candidate  $s$  in  $LUpdate_v$ . The new better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  for every source  $s$  in  $LUpdate_v$  is then used to update the estimates list  $LCurrent_v$  of  $v$ .

**First pass of sub-phase  $p$  of phase  $i$ .** By Lemma 6,, the number of update candidates of  $v$  in any sub-phase of interconnection step of phase  $i$  is at most  $c'_1 \cdot \ln n \cdot \deg_i$  whp. (Recall that all the explorations are restricted to distance at most  $\delta'_i/2$ .) We denote  $\mathcal{N}_i = c'_1 \cdot \ln n \cdot \deg_i$

## 51:12 $(1 + \epsilon)$ -Approximate Shortest Paths in Dynamic Streams

and  $\mu_i = 16 \cdot c_4 \cdot \mathcal{N}_i \cdot \ln n$ , where  $c_4 \geq 1$  is a sufficiently large positive constant. At a high level, in the first pass of every sub-phase, we want to recover, for every vertex  $v \in V$ , a vector (containing sources of explorations that visit  $v$  in sub-phase  $p$ ) with at most  $\mathcal{N}_i$  elements in its support. In other words, we want to perform an  $s$ -sparse recovery for every vertex  $v \in V$ , where  $s = \mathcal{N}_i$ . We do so by performing multiple simultaneous invocations of a procedure called *FindNewCandidate*, for every  $v \in V$ . The pseudocode for procedure *FindNewCandidate* is given in Algorithm 1. The procedure *FindNewCandidate* enables us to sample an update candidate  $s$  of  $v$  (if exists), with a better (than the current) estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  in a specific distance range.

■ **Algorithm 1** Pseudocode for procedure *FindNewCandidate*.

---

```

1: Procedure FindNewCandidate(v, h, I)
2: ▷ Initialization
3: $slots \leftarrow \emptyset$ ▷ An array with $\lambda = \lceil \log n \rceil$ elements indexed from 1 to λ .
 ▷ Each element of slots is a tuple $(sCount, sNames)$. For a given index $1 \leq \tau \leq \lambda$, fields $sCount$
 and $sNames$ of $slots[\tau]$ can be accessed as $slots[\tau].sCount$ and $slots[\tau].sNames$, respectively.
 ▷ $slots[\tau].sCount$ counts the update candidates u seen by v with hash values $h(u) \in [2^\tau]$. It is
 set to 0 initially.
 ▷ $slots[\tau].sNames$ is an encoding of the names of candidate sources seen by v with hash values
 in $[2^\tau]$. It is set to ϕ initially.
 ▷ Update Stage
4: while (there is some update $(e_t, eSign_t, eWeight_t)$ in the stream) do
5: if (e_t is incident on v and some $u \in V$) then
6: for each $(s, \hat{d}(u, s)) \in LCurrent_u$ do
7: if $((\hat{d}(u, s) + eWeight_t) \in I$ and
8: $\hat{d}(u, s) + eWeight_t < \hat{d}(v, s))$ then
9: $\tau \leftarrow \lceil \log h(s) \rceil$
10: repeat ▷ Update $slots[\tau]$ for all $\lceil \log h(s) \rceil \leq \tau \leq \lambda$
11: $slots[\tau].sCount \leftarrow slots[\tau].sCount + eSign_t$
12: $slots[\tau].sNames \leftarrow slots[\tau].sNames + \nu(s) \cdot eSign_t$
13: ▷ The function ν is described in Section 2.3.
14: ▷ The addition in line 12 is a vector addition.
15: $\tau = \tau + 1$
16: until $\tau > \lambda$
 ▷ Recovery Stage
17: if ($slots$ vector is empty) then
18: return (ϕ, ϕ)
19: else if $(\exists \text{ index } \tau \text{ s.t. } \frac{slots[\tau].sNames}{slots[\tau].sCount} = \nu(s) \text{ for some } s \text{ in } V)$ then
20: return $(s, slots[\tau].sCount)$
21: else
22: return (\perp, \perp)

```

---

For every vertex  $v \in V$ , we divide the possible range of better estimates of  $v$ 's  $(2\beta + 1)$ -bounded distances to its update candidates, into sub-ranges on a geometric scale. We then invoke the procedure *FindNewCandidate* repeatedly in parallel to perform an  $\mathcal{N}_i$ -sparse recovery for  $v$  on every sub-range. Specifically, we divide the search space of potential better estimates,  $[1, \delta'_i/2]$ , into sub-ranges  $I_j = ((1 + \zeta')^j, (1 + \zeta')^{j+1}]$ , for  $j \in \{0, 1, \dots, \gamma\}$ , where  $\gamma = \lceil \log_{1+\zeta'} \delta'_i/2 \rceil - 1$ . For  $j = 0$ , we make the sub-range  $I_0 = [(1 + \zeta')^0, (1 + \zeta')^1]$  closed to include the value 1. Note that we are only interested in distances at most  $\delta'_i/2$ . Therefore we restrict our search for distance estimates to the range  $[1, \delta'_i/2]$ .

In more detail, we make for for each  $v \in V$  and for each sub-range  $I_j$ ,  $\mu_i = \Theta(\mathcal{N}_i \cdot \ln n)$  attempts in parallel. In a specific attempt for a given vertex  $v$  and a given sub-range  $I_j$ , we make a single call to procedure *FindNewCandidate* which samples an update candidate

$s$  (if exists) of  $v$  with a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  in the sub-range  $I_j$ . Henceforth, we will refer to an update candidate  $s$  of a vertex  $v$  with a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  in a given distance range  $I$ , as the *update candidate of  $v$  in the range  $I$* . A single call to procedure *FindNewCandidate* succeeds only with a constant probability. Hence multiple parallel calls are required to boost the probability of success. In the extended version we show that making  $\mu_i$  parallel attempts ensures that a specific update candidate  $s$  for some input vertex  $v$  and a distance subrange  $I_j$  gets sampled whp (assuming that it exists).

The procedure *FindNewCandidate* takes as input the ID of a vertex, a hash function  $h$  chosen at random from a family of pairwise independent hash functions and an input range  $I = (low, high]$ . (The input range may be closed as well.) A successful invocation of *FindNewCandidate* for an input vertex  $v$  and a distance range  $I$  returns a tuple  $(s, c_s)$ , where  $s$  is the ID of an update candidate of  $v$  in the range  $I$ , and  $c_s$  is the number of edges  $(v, u) \in E$  such that  $\hat{d}(u, s) + \omega(v, u) \in I$ . If there is no update candidate of  $v$  in the input range  $I$ , procedure *FindNewCandidate* returns a tuple  $(\phi, \phi)$ . If there are update candidates of  $v$  in the input range, but procedure *FindNewCandidate* fails to isolate an ID of such a candidate, it returns  $(\perp, \perp)$ . Before we start making our attempts in parallel, we sample uniformly at random a set of functions  $H_p$  ( $|H_p| = \mu_i$ ) from a family of pairwise independent hash functions  $h : \{1, \dots, maxVID\} \rightarrow \{1, \dots, 2^\lambda\}$ , where  $\lambda = \lceil \log maxVID \rceil = \lceil \log n \rceil$ . Then, for every vertex  $v \in V$  and every distance sub-range  $I_j$ ,  $j \in \{0, 1, \dots, \gamma\}$ , we make  $\mu_i$  parallel calls to procedure *FindNewCandidate*( $v, h, I_j$ ), one call for each  $h \in H_p$ .

**Procedure FindNewCandidate.** Procedure *FindNewCandidate* uses the input hash function  $h$  to sample for the input vertex  $v$ , an update candidate of  $v$  in the input range  $I$ . Let  $d_v^{(I)}$  be the number of update candidates of  $v$  in the input range  $I$ . If we knew the exact value of  $d_v^{(I)}$ , we could sample every new update candidate witnessed by  $v$  with probability  $1/d_v^{(I)}$  to extract exactly one of them in expectation. However, all we know about  $d_v^{(I)}$  is that it is at most  $deg_i$  in expectation (Lemma 6) and at most  $O(deg_i \cdot \ln n)$  whp. We therefore sample every new update candidate seen by  $v$  on a range of probabilities. We use an array *slots* of  $\lambda$  elements, indexed by *slot-levels* from 1 to  $\lambda = \lceil \log n \rceil$ , to implement sampling on a range of probabilities. We want a given update candidate  $s$  to be sampled into slot-level  $\tau$  with probability  $1/2^{\lambda-\tau}$ . When  $d_v^{(I)} \approx 2^{\lambda-\tau}$ , with a constant probability there is exactly one exploration source that gets mapped to *slots*[ $\tau$ ].

Each element of *slots* is a tuple  $(sCount, sNames)$ . The field *sCount* of element at slot-level  $\tau$  counts the new update candidates seen by  $v$  with hash values in  $[2^\tau]$ . It is set to 0 initially. The field *sName* of element at slot-level  $\tau$  is an encoding of the names of candidate sources seen by  $v$  with hash values in  $[2^\tau]$ . It is set to  $\phi$  initially.

The update stage of the procedure for an input vertex  $v$  and an input distance range  $I$  (See lines 4-16 of Algorithm 1) proceeds as follows. For every update  $(e_t, eSign_t, eWeight_t)$  to an edge  $e_t$  incident to  $v$  and some vertex  $u$ , we look at every exploration source  $s$  in the estimates list  $LCurrent_u$  of  $u$ , (see line 6 of Algorithm 1) and check whether the distance estimate of  $v$  to  $s$  via edge  $e_t = (v, u)$  is better than the current value of  $\hat{d}(v, s)$ , and whether it falls in the input distance range  $I$ . (See line 8 of Algorithm 1.) If this is the case, then, we sample  $s$  (on a range of probabilities) by updating the elements of *slots* array from slot-levels  $\lceil \log h(s) \rceil$  to  $\lambda$ . (See Lines 11 and 12.) We use the CIS-based encoding scheme  $\nu$  described in Section 2.3 to encode the names of the exploration sources we sample, and use Lemma 4 to check (See line 19 of Algorithm 1), if we have successfully isolated the ID of a single update candidate in the desired distance range.

We need to make sure that for some  $1 \leq \tau \leq \lambda$ , only one exploration source will get mapped to  $\text{slots}[\tau]$ . By Corollary 13 (see Appendix A), only one exploration source gets mapped to  $\text{slots}[\tau]$  for  $\tau = \lambda - \lceil \log d_v^{(I)} \rceil - 1$ , with at least a constant probability. Therefore, a single call to *FindNewCandidate* succeeds with at least a constant probability. For a vertex  $v \in V$ , if there are no update candidates of  $v$  in sub-phase  $p$ , all the calls to procedure *FindNewCandidate* in all the attempts return  $(\phi, \phi)$ . For every such vertex, we do not need to add anything to its update list  $LU\text{update}_v$ . At the end of the first pass, if no invocation of procedure *FindNewCandidate* returns as error, we extract for every vertex  $v \in V$  and every distance range  $I_j$  ( $j \in \{0, 1, \dots, \gamma\}$ ), all the distinct update candidates of  $v$  in the range  $I_j$  sampled by  $\mu_i$  attempts made for  $v$  and sub-range  $I_j$ . For a given update candidate  $s$  of  $v$ , let  $j = j_{v,s}$  be the smallest index in  $\{0, 1, \dots, \gamma\}$ , such that a tuple  $(s, c_s)$  (for some  $c_s > 0$ ) is returned by a call to procedure *FindNewCandidate* $(v, h, I_j)$ . We add a tuple  $(s, I_j, c_s)$  to the list of update candidates  $LU\text{update}_v$  of  $v$ . Recall that the set  $LU\text{update}_v$  of vertex  $v$  contains tuples  $(s, \text{range}, r_s)$ , where  $s$  is the ID of an update candidate of  $v$ ,  $\text{range}$  is the distance range in which a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  lies, and  $r$  is the number of edges  $(u, v) \in \Gamma_G(v)$  such that  $\hat{d}(u, s) + \omega(u, v) \in \text{range}$ .

**Second pass of sub-phase  $j$  of phase  $i$ .** The second pass of sub-phase  $p$  starts with the update lists  $LU\text{update}_v$  of every  $v \in V$ . We find for every tuple  $(s, \text{range}, r)$  in  $LU\text{update}_v$ , a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  in the sub-range  $\text{range}$ , by invoking procedure *GuessDistance* (described in Appendix B.1)  $O(\log n)$  times. We sample uniformly at random a set of  $c_1 \log_{7/8} n = O(\log n)$  pairwise independent hash functions  $H'_p$  from the family  $h : \{1, \dots, \max\text{VID}\} \rightarrow \{1, 2, \dots, 2^\lambda\}$  ( $\lambda = \lceil \log n \rceil$ ), to be used by invocations of procedure *GuessDistance*.

Note that the current estimate  $\hat{d}(v, s)$  of input vertex  $v$ 's distance to its update candidate  $s$  is either available in its estimates list  $LC\text{urrent}_v$  or is implicitly set to  $\infty$ . The latter happens if  $v$  has not yet been visited by the exploration rooted at source  $s$ . At the end of the second pass, we have the results of all the invocations of procedure *GuessDistance*, for a given vertex  $v$  corresponding to the tuple  $(s, \text{range}, r) \in LU\text{update}_v$ . We update the corresponding tuple  $(s, \hat{d}(v, s))$  in the estimates list  $LC\text{urrent}_v$  of  $v$  with the minimum value returned by any invocation of *GuessDistance* for vertex  $v$ . If an entry corresponding to  $s$  is not present in the estimates list  $LC\text{urrent}_v$  at this stage (i.e.,  $\hat{d}(v, s) = \infty$  as above), then we add a new tuple to the estimates list of  $v$ . Finally, the updates lists of all the vertices are cleared to be re-used in the next sub-phase. At the end of second pass of sub-phase  $p$ , we go through the edges of the lower level hopsets and check for each  $v \in V$  whether a better estimate of  $d_{G^{(k-1)}}^{(2\beta+1)}(v, s)$  for any source  $s \in CU_i$  is available through one of the hopset edges. If this is the case, then we update the estimates lists accordingly.

Finally, after  $2\beta + 1$  sub-phases of the interconnection step of phase  $i$ , we go through the estimates list of every center  $r_C \in CU_i$  to check for every center  $r'_C \in CU_i$ , whether, there is a tuple  $(r'_C, \hat{d}(r_C, r'_C)) \in LC\text{urrent}_{r_C}$  and  $\hat{d}(r_C, r'_C) \leq \delta'_i/2$ . Then, for every such center  $r'_C$  found, we add an edge  $(r_C, r'_C)$  of weight  $\hat{d}(r_C, r'_C)$  into hopset  $H_k$ .

Next, we analyze the properties of our final hopset  $H$ .

### 3.3 Size, Stretch and Hopbound Analysis

**Size:** The size of our hopset  $H$  is the same as that of the insertion-only algorithm of [16], since we follow the same criteria (as in [16]), when deciding which cluster centres to connect via a hopset edge during our construction. Thus, the overall size of the hopset produced by our construction is  $O(n^{1+1/\kappa} \cdot \log \Lambda)$  in expectation.

**Stretch and Hopbound.** Recall that  $\epsilon_k$  is the value such that the graph  $G^{(k)}$  (which is a graph obtained by adding the edges of hopset  $H^{(k)} = \bigcup_{k_0 \leq j \leq k} H_j$  to the input graph  $G$ ) provides stretch at most  $1 + \epsilon_k$ . Also, recall that  $k_0 = \lfloor \log \beta \rfloor$  and  $k_\Lambda = \lceil \log \Lambda \rceil$ . Write  $c_5 = 2$ . We need the following lemma from [16] regarding the stretch of a single scale hopset  $H_k$ ,  $k \in [k_0, k_\Lambda]$  produced by the insertion-only algorithm. We refer the reader to Lemma 3.10 and preamble of Theorem 3.11 of [16] for the proof.

► **Lemma 7** ([16]). *Let  $x, y \in V$  be such that  $2^k \leq d_G(x, y) \leq 2^{k+1}$ , then it holds that*

$$d_{G \cup H_k}^{(h_\ell)}(x, y) \leq (1 + \epsilon_{k-1})(1 + 16 \cdot c_5 \cdot \ell \cdot \epsilon) d_G(x, y), \quad (3)$$

and  $h_\ell = O(\frac{1}{\epsilon})^\ell$  is the hopbound. (Here  $c_5$  is a fixed constant.)

**Rescaling.** Define  $\epsilon'' = 16 \cdot c_5 \cdot \ell \cdot \epsilon$ . Therefore, the stretch of a single scale hopset  $H_k$ ,  $k \in [k_0, k_\Lambda]$ , produced by the insertion-only algorithm of [16] becomes  $(1 + \epsilon_{k-1})(1 + \epsilon'')$ . After rescaling, the hopbound  $h_\ell$  becomes  $O(\frac{\ell}{\epsilon''})^\ell$ . Recall that  $\ell = \ell(\kappa, \rho) = \lfloor \log(\kappa\rho) \rfloor + \lceil \frac{\kappa+1}{\rho\kappa} \rceil - 1 \leq \log(\kappa\rho) + \lceil 1/\rho \rceil$ , is the number of phases of our single-scale hopset construction. It follows that the hopbound of the insertion-only algorithm is

$$\beta_{EN} = O\left(\frac{\log \kappa\rho + 1/\rho}{\epsilon''}\right)^{\log \kappa\rho + 1/\rho}. \quad (4)$$

Observe that for  $k = k_0$ , graph  $G^{(k-1)}$  is the input graph  $G$  itself, since  $H_k$  for all  $k < k_0$  is an empty set. (See Section 3.1 for details.) Therefore,  $1 + \epsilon_{k-1}$  for  $k = k_0$  is equal to 1. It follows therefore that the stretch  $1 + \epsilon_k = 1 + \epsilon_{k_{EN}}$ , of the insertion-only algorithm follows the following sequence:  $1 + \epsilon_{k_0_{EN}} = (1 + \epsilon'')$  and for the higher scales,  $1 + \epsilon_{k+1_{EN}} = (1 + \epsilon'') \cdot (1 + \epsilon_{k_{EN}})$ .

The stretch of our single scale hopset construction (Section 3.2) for any scale  $(2^k, 2^{k+1}]$ ,  $k_0 \leq k \leq k_\Lambda$  is  $(1 + \chi)$  times the stretch of the corresponding hopset produced by the insertion-only algorithm. We set  $\chi = \epsilon''$ . Incorporating the additional stretch incurred by our algorithm into the stretch analysis of [16], we get the following lemma about the stretch of our dynamic streaming algorithm.

► **Lemma 8.** *For  $k \in [k_0, k_\Lambda]$ , we have*

$$\begin{aligned} 1 + \epsilon_{k_0} &= (1 + \epsilon'')^2 \\ 1 + \epsilon_k &= (1 + \epsilon'')^2 (1 + \epsilon_{k-1}) \text{ for } k > k_0 \end{aligned}$$

Observe that Lemma 8 implies that the overall stretch of our hopset  $H$  is at most  $(1 + \epsilon'')^{2 \log \Lambda}$ . Recall that the desired stretch of our hopset construction is  $1 + \epsilon'$  (see Section 3.1), where  $\epsilon' > 0$  is an input parameter of our algorithm. We set  $\epsilon'' = \frac{\epsilon'}{4 \cdot \log \Lambda}$ , and it follows that our overall stretch is  $\left(1 + \frac{\epsilon'}{4 \log \Lambda}\right)^{2 \log \Lambda} \leq 1 + \epsilon'$ . It follows that  $\epsilon = \frac{\epsilon'}{64 \cdot c_5 \cdot \ell \cdot \log \Lambda}$ .

Plugging in  $\epsilon'' = \frac{\epsilon'}{4 \cdot \log \Lambda}$  in (4), we get the following expression for the hopbound of our dynamic streaming hopset:

$$\beta' = O\left(\frac{\log \Lambda}{\epsilon'} (\log \kappa\rho + 1/\rho)\right)^{\log \kappa\rho + 1/\rho}. \quad (5)$$

Also recall that we had defined  $\beta = (\frac{1}{\epsilon})^\ell$  for using  $2\beta + 1$  as the hop-depth of our explorations. After the two rescaling steps as above, we get that  $\beta = \beta'$ .



### 3.4 Summary

A detailed analysis of the resource usage of the hopset construction algorithm is provided in extended version. In the extended version we also generalize our result to path-reporting hopsets, and integrate it with a weight reduction that eliminates most of the dependencies on  $\log \Lambda$ . We summarize the results below.

► **Theorem 9.** *For any  $n$ -vertex graph  $G(V, E, \omega)$  with aspect ratio  $\Lambda$ ,  $2 \leq \kappa \leq (\log n)/4$ ,  $1/\kappa \leq \rho \leq 1/2$  and  $0 < \epsilon' < 1$ , our dynamic streaming algorithm computes whp, a  $(1 + \epsilon', \beta')$  hopset  $H$  with expected size  $O(n^{1+1/\kappa} \cdot \log n)$  and the hopbound  $\beta'$  given by*

$$\beta' = O\left(\frac{(\log \kappa \rho + 1/\rho) \log n}{\epsilon'}\right)^{\log \kappa \rho + 1/\rho}$$

*It does so by making  $O(\beta' \cdot (\log \kappa \rho + 1/\rho))$  passes through the stream and using  $O(n \cdot \log^3 n \cdot \log \Lambda)$  bits of space in the first pass and  $O(\frac{\beta'}{\epsilon'} \cdot \log^2 1/\epsilon' \cdot n^{1+\rho} \cdot \log^5 n)$  bits of space (respectively  $O(\frac{\beta'^2}{\epsilon'} \cdot \log^2 1/\epsilon' \cdot n^{1+\rho} \cdot \log^5 n)$  bits of space for path-reporting hopset) in each of the subsequent passes.*

## 4 $(1 + \epsilon)$ -Approximate Shortest Paths

Consider the problem of computing  $(1 + \epsilon)$ -approximate shortest paths (henceforth  $(1 + \epsilon)$ -ASP) for all pairs in  $S \times V$ , for a subset  $S$ ,  $|S| = s$ , of designated source vertices, in a weighted undirected  $n$ -vertex graph  $G = (V, E, \omega)$  with aspect ratio  $\Lambda$ . Let  $\epsilon, \rho > 0$  be parameters, and assume that  $s = O(n^\rho)$ . Our dynamic streaming algorithm for this problem computes a path-reporting  $(1 + \epsilon, \beta)$ -hopset  $H$  of  $G$  with  $\beta = O(\frac{\log n}{\epsilon \rho})^{1/\rho}$  using our hopset construction algorithm, with  $\kappa = 1/\rho$ . Once the hopset  $H$  has been computed, we conduct  $(1 + \epsilon)$ -approximate Bellman-Ford explorations in  $G \cup H$  to depth  $\beta$  from all the sources of  $S$ . (See the algorithm from Appendix B.) By Theorem 14, this requires  $O(\beta)$  passes of the stream, and space  $O(|S| \cdot n \cdot \text{poly}(\log n, \log \Lambda))$ , and results in  $(1 + \epsilon)$ -approximate distances  $d_{G \cup H}^{(\beta)}(s, v)$ , for all  $(s, v) \in S \times V$ . (Note that following every pass over  $G$ , we do an iteration of Bellman-Ford over the hopset  $H$  offline, as  $H$  is stored by the algorithm.) In addition, for every pair  $(s, v) \in S \times V$ , we also get the parent of  $v$  on the exploration rooted at source  $s$ . We compute the path  $\pi_{G \cup H}(s, v)$  between  $s$  and  $v$  in graph  $G \cup H$  from these parent pointers. The path-reporting property of our hopset  $H$  enables us to replace any hopset edge  $e = (x, y) \in H$  on the path  $\pi_{G \cup H}(s, v)$  with a corresponding path  $\pi_G(x, y)$  in  $G$ . In the extended version we also argue that these replacements can be performed using  $\tilde{O}(n^{1+\rho})$  space. By definition of the hopset, we have  $d_G(s, v) \leq d_{G \cup H}^{(\beta)}(s, v) \leq (1 + \epsilon) \cdot d_G(s, v)$ , and the estimates  $\hat{d}(s, v)$  computed by our approximate Bellman-Ford algorithm satisfy  $d_{G \cup H}^{(\beta)}(s, v) \leq \hat{d}(s, v) \leq (1 + \epsilon) \cdot d_{G \cup H}^{(\beta)}(s, v)$ . Thus, we have,  $d_G(s, v) \leq \hat{d}(s, v) \leq (1 + \epsilon)^2 \cdot d_G(s, v)$ . By rescaling  $\epsilon' = 3\epsilon$ , we obtain  $(1 + \epsilon)$ -approximate  $S \times V$  paths, the total space complexity of the algorithm is  $O(n^{1+\rho} \cdot \text{poly}(\log n, \log \Lambda))$ , and the number of passes is  $\text{poly}(\log n)$ . We derive the following theorem:

► **Theorem 10.** *For any parameters  $\epsilon, \rho > 0$ , and any  $n$ -vertex undirected weighted graph  $G = (V, E, \omega)$  with polynomial in  $n$  aspect ratio, and any set  $S \subseteq V$  of  $n^\rho$  distinguished sources,  $(1 + \epsilon)$ -ASP for  $S \times V$  can be computed in dynamic streaming setting in  $\tilde{O}(n^{1+\rho})$  space and  $\log^{\frac{1}{\rho} + O(1)} n = \text{polylog}(n)$  passes.*

## References

- 1 Amir Abboud and Greg Bodwin. The  $4/3$  additive spanner exponent is tight. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 351–361, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897555.
- 2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467, USA, 2012. Society for Industrial and Applied Mathematics.
- 3 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'12, pages 5–14, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213556.2213560.
- 4 Ingo Althofer, Gautam Das, David Dobkin, and Deborah A Joseph. Generating sparse spanners for weighted graphs. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1989.
- 5 Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1190–1197. SIAM, 2010. doi:10.1137/1.9781611973075.95.
- 6 Antal Balog and Imre Bárány. On the convex hull of the integer points in a disc. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 162–165, New York, NY, USA, 1991. Association for Computing Machinery. doi:10.1145/109648.109666.
- 7 Surender Baswana. Streaming algorithm for graph spanners – single pass and constant processing time per edge. *Information Processing Letters*, 106(3):110–114, 2008. doi:10.1016/j.ipl.2007.11.001.
- 8 J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. doi:10.1016/0022-0000(79)90044-8.
- 9 Yi-Jun Chang, Martin Farach-Colton, Tsan-sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 34:1–34:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.34.
- 10 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. In *STOC '94*, 1994.
- 11 Graham Cormode and D. Firmani. A unifying framework for  $\epsilon$ -sampling algorithms. *Distributed and Parallel Databases*, 32:315–335, 2013.
- 12 Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algorithms*, 7(2), March 2011. doi:10.1145/1921659.1921666.
- 13 Michael Elkin. Distributed exact shortest paths in sublinear time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 757–770. ACM, 2017. doi:10.1145/3055399.3055452.
- 14 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Improved weighted additive spanners. *CoRR*, abs/2008.09877, 2020. arXiv:2008.09877.
- 15 Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Trans. Algorithms*, 15(1), November 2018. doi:10.1145/3274651.
- 16 Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. *SIAM Journal on Computing*, 48(4):1436–1480, 2019. doi:10.1137/18M1166791.

- 17 Michael Elkin and Ofer Neiman. Near-additive spanners and near-exact hopsets, A unified view. *Bull. EATCS*, 130, 2020. URL: <http://bulletin.eatcs.org/index.php/beatcs/article/view/608/624>.
- 18 Michael Elkin and David Peleg.  $(1 + \epsilon, \beta)$ -spanner constructions for general graphs. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 173–182, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/380752.380797.
- 19 Michael Elkin and Shay Solomon. Fast constructions of light-weight spanners for general graphs. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 513–525. SIAM, 2013. doi:10.1137/1.9781611973105.37.
- 20 Michael Elkin and Chhaya Trehan.  $\$(1+\epsilon)\$$ -approximate shortest paths in dynamic streams. *CoRR*, abs/2107.13309, 2021. arXiv:2107.13309.
- 21 Michael Elkin and Jian Zhang. Efficient algorithms for constructing  $(1 + \epsilon, \beta)$ -spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006. doi:10.1007/s00446-005-0147-2.
- 22 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming*, pages 531–543, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- 23 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2009. doi:10.1137/070683155.
- 24 Manuel Fernandez, David P. Woodruff, and Taisuke Yasuda. Graph spanners in the message-passing model. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPICs*, pages 77:1–77:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.77.
- 25 Arnold Filtser, Michael Kapralov, and Navid Nouri. *Graph Spanners by Sketching in Dynamic Streams and the Simultaneous Communication Model*, pages 1894–1913. SIAM, 2021. doi:10.1137/1.9781611976465.113.
- 26 S. Ganguly. Counting distinct items over update streams. *Theor. Comput. Sci.*, 378:211–222, 2007.
- 27 David Gibb, Bruce M. Kapron, Valerie King, and Nolan Thorn. Dynamic graph connectivity with improved worst case update time and sublinear space. *CoRR*, abs/1509.06464, 2015. arXiv:1509.06464.
- 28 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming*, pages 725–736, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
- 29 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 489?498, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897638.
- 30 Piotr Indyk, Eric Price, and David P. Woodruff. On the power of adaptivity in sparse recovery. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 285–294. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.83.
- 31 Vojtěch Jarník. Über die gitterpunkte auf konvexen kurven. *Mathematische Zeitschrift*, 24:500–518, 1926.
- 32 Michael Kapralov and David Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, pages 272–281, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2611462.2611497.

- 33 Valerie King, Shay Kutten, and Mikkel Thorup. Construction and impromptu repair of an MST in a distributed network with  $o(m)$  communication. *CoRR*, abs/1502.03320, 2015. [arXiv:1502.03320](#).
- 34 Philip N Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997. doi:10.1006/jagm.1997.0888.
- 35 Felix Lazebnik and Vasily A. Ustimenko. Some algebraic constructions of dense graphs of large girth and of large size. In Joel Friedman, editor, *Expanding Graphs, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, May 11-14, 1992*, volume 10 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 75–93. DIMACS/AMS, 1992. doi:10.1090/dimacs/010/07.
- 36 Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014. doi:10.1145/2627692.2627694.
- 37 Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error  $\ell_p$ -sampling with applications. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1143–1160. SIAM, 2010. doi:10.1137/1.9781611973075.92.
- 38 David Peleg and Alejandro A. Schaffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

## A Hash Functions

A hash function  $h$  maps elements from a given input domain to an output domain of bounded size. Ideally, we would like to draw our hash function randomly from the space of all possible functions on the given input/output domain. However, since we are concerned about the space used by our algorithm, we will rely on hash functions with limited independence. A family of functions  $H = \{h : \mathcal{U} \rightarrow [m]\}$ , from a universe  $\mathcal{U}$  to  $[m]$ , for some positive integers  $m$  and  $k$ , is said to be *k-wise independent*, if it holds that, when  $h$  is chosen uniformly at random from  $H$  then for any  $k$  distinct elements  $x_1, x_2, \dots, x_k \in \mathcal{U}$ , and any  $k$  elements  $z_1, z_2, \dots, z_k \in [m]$ ,  $x_1, x_2, \dots, x_k$  and mapped by  $h$  to  $z_1, z_2, \dots, z_k$  with probability  $1/m^k$ , i.e., as if they were perfectly random. Such functions can be described more compactly, but are sufficiently random to allow formal guarantees to be proven. The following lemma summarizes the space requirement of limited independence hash functions:

► **Lemma 11** ([8]). *A function drawn from a family of  $k$ -wise independent hash functions can be encoded in  $O(k \log n)$  bits.*

Specifically, we will be using *pairwise independent* hash functions.

The following lemma, a variant of which has also been proved in [27, 33] in a different context, is proved in the extended version.

► **Lemma 12.** *Let  $h : \mathcal{U} \rightarrow [2^\lambda]$  be a hash function sampled uniformly at random from a family of pairwise independent hash functions  $\mathcal{H}$ . If we use  $h$  to hash elements of a given set  $\mathcal{S} \subseteq \mathcal{U}$  such that  $|\mathcal{S}| = s$ , then a specific element  $d \in \mathcal{S}$  hashes to the set  $[2^t]$ ,  $t = \lambda - \lceil \log s \rceil - 1$  and no other element of  $\mathcal{S}$  does so with probability at least  $\frac{1}{8s}$ .*

Lemma 12 implies the following corollary:

► **Corollary 13.** *Let  $h : \mathcal{U} \rightarrow [2^\lambda]$  be a hash function sampled uniformly at random from a family of pairwise independent hash functions  $\mathcal{H}$ . If we use  $h$  to hash elements of a given set  $\mathcal{S} \subseteq \mathcal{U}$  with  $|\mathcal{S}| = s$ , then exactly one element in  $\mathcal{S}$  hashes to the set  $[2^t]$ ,  $t = \lambda - \lceil \log s \rceil - 1$ , with probability at least  $\frac{1}{8}$ .*

## B

 Approximate Bellman-Ford Explorations

In this section, we describe an algorithm for performing a given number of iterations of an approximate B-F (Bellman-Ford) exploration from a given subset  $S \subseteq V$  of *source* vertices in a weighted undirected graph  $G(V, E, \omega)$  with aspect ratio  $\Lambda$ . For a given vertex  $v \in V$  and a set  $S \subseteq V$ , the  $t$ -bounded distance between  $v$  and  $S$  in  $G$ , denoted  $d_G^{(t)}(v, S)$ , is the length of a shortest  $t$ -bounded path between  $v$  and some  $s \in S$  (See Definition 3) such that  $d_G^{(t)}(v, s) = \min\{d_G^{(t)}(s', v) \mid s' \in S\}$ .

Given an  $n$ -vertex weighted graph  $G(V, E, \omega)$ , a set  $S \subseteq V$  of vertices, an integer parameter  $\eta > 0$  and an error parameter  $\zeta \geq 0$ , an  $(\eta, \zeta)$ -B-F exploration of  $G$  rooted at  $S$  outputs for every vertex  $v \in V$ , a  $(1 + \zeta)$ -approximation of its  $\eta$ -bounded distance to the set  $S$ .

Throughout the execution of our algorithm, we maintain two variables for each vertex  $v \in V$ . One of them is a current estimate of  $v$ 's  $\eta$ -bounded distance to set  $S$ , denoted  $\hat{d}(v)$ , and the other is the ID of  $v$ 's neighbour through which it gets its current estimate, denoted  $\hat{p}(v)$ , and called the *parent* of  $v$ . We start by initializing  $\hat{d}(s) = 0$ ,  $\hat{p}(s) = \perp$ , for each  $s \in S$  and  $\hat{d}(v) = \infty$ ,  $\hat{d}(v) = \perp$  for each  $v \in V \setminus S$ . As the algorithm proceeds,  $\hat{d}(v)$  and  $\hat{p}(v)$  values of every vertex  $v \in V \setminus S$  are updated to reflect the current best estimate of  $d_G^{(\eta)}(v, S)$ . The final value of  $\hat{d}(v)$  for each  $v \in V$  is such that  $d_G^{(\eta)}(v, S) \leq \hat{d}(v) \leq (1 + \zeta) \cdot d_G^{(\eta)}(v, S)$ , and the final value of  $\hat{p}(v)$  for each  $v \in V$  contains the ID of  $v$ 's parent on the forest spanned by  $(\eta, \zeta)$ -B-F exploration of  $G$  rooted at the set  $S$ . The algorithm proceeds in phases, indexed by  $p$ ,  $1 \leq p \leq \eta$ . We make one pass through the stream in each phase.

**Phase  $p$ .** In every phase, we search for every vertex  $v \in V \setminus S$ , a *better* (smaller than the current value of  $\hat{d}(v)$ ) estimate (if exists) of its  $\eta$ -bounded distance to the set  $S$ , by keeping track of updates to edges  $e = (v, u)$  incident to  $v$ . Specifically, we divide the search space of potential better estimates,  $[1, 2 \cdot \Lambda]$ , into sub-ranges  $I_j = ((1 + \zeta')^j, (1 + \zeta')^{j+1}]$ , for  $j \in \{0, 1, \dots, \gamma\}$ , where  $\gamma = \lceil \log_{1+\zeta'} 2 \cdot \Lambda \rceil - 1$  and  $\zeta'$  is set to  $\zeta/2\eta$  for technical reasons. For  $j = 0$ , we make the sub-range  $I_0 = [(1 + \zeta')^0, (1 + \zeta')^1]$  closed to include the value 1. Recall that we are doing a  $(1 + \zeta)$ -approximate B-F exploration (and not an exact one). Due to this, some of the better estimates we get in a given phase may be between  $\Lambda$  and  $(1 + \zeta) \cdot \Lambda \leq 2 \cdot \Lambda$ , where  $\Lambda$  is the aspect ratio of the input graph. We therefore keep our search space from 1 to  $2\Lambda$  instead of  $\Lambda$ . In more detail, we make for each  $v \in V \setminus S$ ,  $\gamma$  *guesses*, one for each sub-range. In a specific guess for a vertex  $v$  corresponding to sub-range  $((1 + \zeta')^j, (1 + \zeta')^{j+1}]$  for some  $j$ , we make multiple simultaneous calls to a randomized procedure called *GuessDistance* which samples an edge (if exists) between  $v$  and some vertex  $u$  such that

$$\hat{d}(u) + \omega(v, u) \in I_j.$$

The exact number of calls we make to procedure *GuessDistance* in each guess will be specified later in the sequel. The smallest index  $j \in [0, \gamma]$ , for which the corresponding guess denoted  $\text{Guess}_v^{(j)}$  successfully samples an edge which gives a distance estimate better than the current estimate of  $v$ , is chosen to update  $\hat{d}(v)$ .

The pseudocode for procedure *GuessDistance* is given in Algorithm 2.

The procedure *GuessDistance* takes as input the ID of a vertex, a hash function  $h$  chosen at random from a family of pairwise independent hash functions and an input range  $I = (\text{low}, \text{high}]$ . The input range may be closed as well.

A successful invocation of procedure *GuessDistance* for an input vertex  $x$  and input range  $I$ , returns a tuple  $(\text{dist}, \text{parent})$ , (if there is at least one edge  $(x, y)$  in  $G$  such that  $\hat{d}(y) + \omega(x, y) \in I$ , and  $(\infty, \infty)$  otherwise), where  $\text{dist}$  is an estimate of  $x$ 's  $\eta$ -bounded

■ **Algorithm 2** Pseudocode for Procedure *GuessDistance*.

---

```

1: Procedure GuessDistance(x, h, I) ▷ Initialization
2: $slots \leftarrow \emptyset$ ▷ An array with λ elements indexed from 1 to λ , where $\lambda = \lceil \log n \rceil$.
 ▷ Each element of slots is a tuple $(xCount, xDist, xName)$. For a given index $1 \leq \tau \leq \lambda$, fields $xCount$, $xDist$ and $xName$ of $slots[\tau]$ can be accessed as $slots[\tau].xCount$, $slots[\tau].xDist$ and $slots[\tau].xName$, respectively.
 ▷ $slots[\tau].xCount$ is the number of sampled edges (x, y) with $h(y) \in [2^\tau]$. Initially, it is set to 0.
 ▷ $slots[\tau].xDist$ is the distance estimate for x provided by an edge (x, y) with $h(y) \in [2^\tau]$. Initially, it is set to 0.
 ▷ $slots[\tau].xName$ is encoding of the names of the endpoints y of sampled edges (x, y) with $h(y) \in [2^\tau]$. Initially, it is set to ϕ .
3: while (there is some update $(e_t, eSign_t, eWeight_t)$ in the stream) do ▷ Update Stage
4: if (e_t is incident on x and some y such that $\hat{d}(y) + eWeight_t \in I$) then
5: $\tau \leftarrow \lceil \log h(y) \rceil$
6: repeat ▷ Update $slots[\tau]$ for all $\lceil \log h(y) \rceil \leq \tau \leq \lambda$.
7: $slots[\tau].xCount \leftarrow slots[\tau].xCount + eSign_t$
8: $slots[\tau].xDist \leftarrow slots[\tau].xDist + (\hat{d}(y) + eWeight_t) \cdot eSign_t$
9: $slots[\tau].xName \leftarrow slots[\tau].xName \oplus name(y)$ ▷ \oplus stands for bitwise XOR.
10: $\tau = \tau + 1$
11: until $\tau > \lambda$ ▷ Recovery Stage
12: if ($slots$ array is empty) then
13: return (∞, ∞)
14: else if (\exists index τ | $slots[\tau].xCount = 1$) then
15: return $(slots[\tau].xDist, slots[\tau].xName)$
16: else
17: return (\perp, \perp)

```

---

distance to the set  $S$  in the range  $I$ , and *parent* is the *parent* of  $x$  in the forest spanned by  $(\eta, \zeta)$ -B-F exploration of  $G$  rooted at the set  $S$ . The procedure *GuessDistance* may fail to return (with a constant probability) a distance estimate in the desired range, even when such an estimate exists. It returns an error, denoted by  $(\perp, \perp)$ , in that case.

Before we start making calls to procedure *GuessDistance*, we sample uniformly at random a set of functions  $H_p$  of size  $c_1 \log_{8/7} n$  from a family of pairwise independent hash functions  $h : \{1, \dots, maxVID\} \rightarrow \{1, \dots, 2^\lambda\}$ , where  $\lambda = \lceil \log n \rceil$  and  $c_1$  is an appropriate constant. For every guess for a given vertex  $x \in V \setminus S$  and a given subrange  $I_j$ , we make  $|H_p|$  parallel calls to procedure *GuessDistance*, one for each  $h \in H_p$ , to get an estimate of  $d_G^{(\eta)}(x, S)$  in the given subrange. The multiple parallel calls are required since a single call to procedure *GuessDistance* succeeds only with a constant probability, while we need to succeed with high probability.

Additionally, before we start the phase  $p$ , we create for each  $v \in V \setminus S$ , a copy  $\hat{d}'(v)$  of its current distance estimate  $\hat{d}(v)$ . Any update to the distance estimate of a vertex  $v$  during phase  $p$  is made to its *shadow* distance estimate  $\hat{d}'(v)$ . On the other hand, the variable  $\hat{d}(v)$  for vertex  $v \in V \setminus S$  remains unchanged during the execution of phase  $p$ . At the end of phase  $p$ , we update  $\hat{d}(v)$  as  $\hat{d}(v) = \hat{d}'(v)$ . The purpose of using the shadow variable is to avoid any issues arising due to simultaneous reading from and writing to the distance estimate variable of a vertex by multiple parallel calls to procedure *GuessDistance*.



## B.1 Procedure GuessDistance

For a given vertex  $x$ , and a given distance range  $I$ , let  $y \in \Gamma_G(x)$  be such that

$$\hat{d}(y) + \omega(x, y) \in I.$$

In what follows, we will refer to such a vertex  $y \in \Gamma_G(x)$  as a *candidate neighbour* and the corresponding edge  $(x, y)$  as a *candidate edge* in the range  $I$ . For a given vertex  $x$ , let  $c_x^{(p,j)}$  be the number of candidate neighbours of  $x$  in the sub-range  $I_j$ . A call to procedure *GuessDistance* for vertex  $x$  with input range  $I = I_j$  works by sampling a *candidate* neighbour with probability  $\frac{1}{c_x^{(p,j)}}$ . We use the input hash function  $h$  to assign hash values to the candidate edges in the range  $\{1, \dots, 2^\lambda\}$ , where  $\lambda = \lceil \log n \rceil$ . We only know an upper bound of  $n$  and not the exact value of  $c_x^{(p,j)}$ . Therefore, we try to guess  $c_x^{(p,j)}$  on a geometric scale of values  $2^{\lambda-\tau}$ ,  $\tau = 1, 2, \dots, \lambda$ , and sample every candidate neighbour on a range of probabilities corresponding to our guesses of  $c_x^{(p,j)}$ .

To implement sampling on a range of probabilities, we use an array *slots* of  $\lambda$  elements indexed by *slot-levels* from 1 to  $\lambda$ . Every new candidate neighbour  $y$  witnessed by  $x$  is assigned a hash value  $h(y)$  by  $h$ . In every element of *slots*, we maintain a tuple  $(xCount, xDist, xName)$ , and  $xCount$ ,  $xDist$  and  $xName$  of  $slots[\tau]$  can be accessed as  $slots[\tau].xCount$ ,  $slots[\tau].xDist$  and  $slots[\tau].xName$ , respectively. The variable  $xCount \in \mathbb{Z}$  at slot-level  $\tau$  maintains the number of candidate neighbours with hash values in  $[2^\tau]$ . It is initialized to 0 at the beginning of the stream. Every time an update to a candidate edge  $e_t = (x, y)$  with  $h(y) \in [2^\tau]$  appears on the stream,  $slots[\tau].xCount$  is updated by adding the  $eSign_t$  value of  $e_t$  to its current value. The variable  $xDist$  at slot-level  $\tau$  is an estimate of  $\eta$ -bounded distance of  $x$  limited to the input distance range  $I$  provided by edge  $(x, y)$  with  $h(y) \in [2^\tau]$ . Initially, it is set to 0. Every time an update to a candidate edge  $e_t = (x, y)$  with  $h(y) \in [2^\tau]$  appears on the stream,  $slots[\tau].xDist$  is updated by adding the value of the expression  $(\hat{d}(y) + eWeight_t) \cdot eSign_t$  to its current value. (Recall that it is initialized as 0.) The variable  $xName$  is encoding of the names of endpoints  $y$  of the sampled edges  $(x, y)$  with  $h(y) \in [2^\tau]$ . It is set to  $\phi$  initially. Every time an update to a candidate edge  $e_t = (x, y)$  with  $h(y) \in [2^\tau]$  appears on the stream,  $slots[\tau].xName$  is updated by performing a bitwise XOR of its current value with  $name(y)$ .

At the end of the stream, if the *slots* array is empty, then there are no candidate neighbours in  $\Gamma_G(x)$  and the procedure *GuessDistance* returns  $(\infty, \infty)$ . If there is a slot-level  $\tau$  such that  $slots[\tau].xCount = 1$ , then only one candidate neighbour is mapped to slot-level  $\tau$ . In this case,  $slots[\tau].xDist$  gives us an estimate of  $x$ 's  $\eta$ -bounded distance to the set  $S$  in the input distance range  $I$ , and  $slots[\tau].xName$  gives us the name of  $x$ 's parent on the forest spanned by the  $(\eta, \zeta)$ -B-F exploration of  $G$  rooted at set  $S$ . Indeed, if no smaller scale estimate will be discovered, the vertex recorded in  $slots[\tau].xName$  will become the parent of  $x$  in the forest. The procedure *GuessDistance* returns  $(slots[\tau].xDist, slots[\tau].xName)$ . If the *slots* vector is not empty but there is no slot level with  $xCount = 1$ , then the procedure *GuessDistance* has failed to find a distance estimate in the input range  $I$  for  $x$ , and thus it returns an error  $(\perp, \perp)$ .

If the input vertex  $x$  has some candidate neighbours in the input distance range, we need to make sure that for some  $1 \leq \tau \leq \lambda$ , only one candidate neighbour will get mapped to  $slots[\tau]$ . By Corollary 13, only one of the  $c_x^{(p,j)}$  candidate neighbours gets mapped to the set  $[2^\tau]$ , for  $\tau = \lambda - \lceil \log c_x^{(p,j)} \rceil - 1$ , with at least a constant probability. Therefore, a single invocation of procedure *GuessDistance* for a given vertex  $x$  and a given distance range succeeds with at least a constant probability. Since we are running  $|H_p|$  parallel invocations of procedure *GuessDistance* for a given input vertex  $x$  and a given distance range  $I$ , we pick the output of a successful invocation of procedure *GuessDistance* as an estimate for  $x$  in the



input range. In the case that all the invocations of *GuessDistance* in a guess return an error, the algorithm terminates with an error. In the extended version, we show that when the set  $H_p$  is appropriately sized, the event of all the invocations of procedure *GuessDistance* in a given guess failing has a very low probability.

Once all the  $\gamma = O(\frac{\log \Lambda}{\zeta'})$  guesses for a given vertex  $x$  have completed their execution without failure, we pick the smallest index  $j$  for which the corresponding guess  $\text{guess}_x^{(j)}$  has returned a finite (non-failure) value, and compare this value with  $\hat{d}(x)$ . If this value gives a better estimate than the current value of  $\hat{d}(x)$ , we update the corresponding shadow variable  $\hat{d}'(x)$ , and the parent variable  $\hat{p}(x)$ . At the end of phase  $p$ , if the algorithm has not terminated with an error, for every vertex  $x \in V \setminus S$ , we update its current distance estimate variable with the value in the corresponding shadow variable as  $\hat{d}(x) = \hat{d}'(x)$ .

A detailed analysis of the algorithm is available in the extended version. The following theorem summarizes the results.

► **Theorem 14.** *For a sufficiently large positive constant  $c$ , given an integer parameter  $\eta$ , an error parameter  $\zeta$ , an input graph  $G(V, E, \omega)$ , and a subset  $S \subseteq V$ , our distance exploration algorithm performs, with probability at least  $1 - \frac{1}{n^c}$ , a  $(1 + \zeta)$ -approximate Bellman-Ford exploration of  $G$  rooted at the set  $S$  to depth  $\eta$ , and outputs for every  $v \in V$ , an estimate  $\hat{d}(v)$  of its distance to set  $S$  and  $v$ 's parent  $\hat{p}(v)$  on the forest spanned by this exploration such that  $d_G^{(\eta)}(v, S) \leq \hat{d}(v) \leq (1 + \zeta) \cdot d_G^{(\eta)}(v, S)$  in  $\eta$  passes through the dynamic stream using*

$$O_c(\eta/\zeta \cdot n \cdot \log^2 n \cdot \log \Lambda(\log n + \log \Lambda)) \text{ space in every pass.}$$

Note also that the space used by the algorithm on different passes can be reused, i.e., the total space used by the algorithm is  $O_c(\eta/\zeta \cdot n \cdot \log^2 n \cdot \log \Lambda(\log n + \log \Lambda))$ .



# Caching with Reserves

Sharat Ibrahimpur ✉

University of Waterloo, Canada

Manish Purohit ✉

Google Research, Mountain View, CA, USA

Zoya Svitkina ✉

Google Research, Mountain View, CA, USA

Erik Vee ✉

Google Research, Mountain View, CA, USA

Joshua R. Wang ✉

Google Research, Mountain View, CA, USA

---

## Abstract

Caching is among the most well-studied topics in algorithm design, in part because it is such a fundamental component of many computer systems. Much of traditional caching research studies cache management for a single-user or single-processor environment. In this paper, we propose two related generalizations of the classical caching problem that capture issues that arise in a multi-user or multi-processor environment. In the *caching with reserves* problem, a caching algorithm is required to maintain at least  $k_i$  pages belonging to user  $i$  in the cache at any time, for some given reserve capacities  $k_i$ . In the *public-private caching* problem, the cache of total size  $k$  is partitioned into subcaches, a private cache of size  $k_i$  for each user  $i$  and a shared public cache usable by any user. In both of these models, as in the classical caching framework, the objective of the algorithm is to dynamically maintain the cache so as to minimize the total number of cache misses.

We show that *caching with reserves* and *public-private caching* models are equivalent up to constant factors, and thus focus on the former. Unlike classical caching, both of these models turn out to be NP-hard even in the offline setting, where the page sequence is known in advance. For the offline setting, we design a 2-approximation algorithm, whose analysis carefully keeps track of a potential function to bound the cost. In the online setting, we first design an  $O(\ln k)$ -competitive fractional algorithm using the primal-dual framework, and then show how to convert it online to a randomized integral algorithm with the same guarantee.

**2012 ACM Subject Classification** Theory of computation → Caching and paging algorithms

**Keywords and phrases** Approximation Algorithms, Online Algorithms, Caching

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.52

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2207.05975> [11]

**Acknowledgements** We would like to thank Sungjin Im for suggesting the caching with reserves problem and for useful discussions.

## 1 Introduction

Caching is one of the most well-studied problems in online computation and also one of the most crucial components of many computer systems. In the classical caching (also referred to as paging) problem, page requests arrive online and an algorithm must maintain a small set of pages to hold in a cache so as to minimize the number of requests that are not served from the cache. Caching algorithms have been widely studied through the lens of competitive analysis and tight results are known [1, 10, 14]. Tight algorithms are also known for many



© Sharat Ibrahimpur, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang;  
licensed under Creative Commons License CC-BY 4.0  
Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 52; pp. 52:1–52:16



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

generalizations such as weighted paging [3, 4], generalized caching [2, 5] and paging with rejection penalties [9]. Due to its practical importance, a large number of heuristic algorithms have been proposed such as Least Recently Used (LRU), Least Frequently Used (LFU), CAR [7], ARC [15], and many others. Although they do not provide the best worst-case performance, they attempt to maximize the hit rate of the cache on practical instances. However, such traditional caching policies (both theoretical and practical) attempt to optimize the global efficiency of the system and are not necessarily suitable for cache management in a multi-user or multi-processor environment. In many of today’s cloud computing services, caches are shared among all the users utilizing the service and optimizing only for global efficiency can lead to highly undesirable allocation for some users. For example, a user who only accesses pages at long intervals may reap no benefit from the cache at all. In this paper, we propose two generalizations of the classical caching problem that are suited for caching in a shared multi-processor environment.

In a multi-user setting, a naive way to guarantee that all users benefit from the cache is to partition the cache among them and effectively maintain separate caches for each user. However, such a system can be extremely inefficient and lead to low overall throughput as the cache can remain underutilized. Instead, a number of recent systems [12, 13, 16, 17] aim to maximize the global efficiency of the cache while attempting to provide (approximate) *isolation guarantees* to each user, i.e., the cache hit rate for each user is at least as much as what it would be if the user was allocated its own isolated cache (of proportionally smaller size). We model the multi-user scenario as the *caching with reserves* problem wherein a caching algorithm is required to maintain at least  $k_i$  pages belonging to user  $i$  in the cache at any time for some input reserve capacities  $k_i$ . As in the classical caching framework, the objective of the algorithm is to dynamically maintain the cache so as to minimize the total number of cache misses. The reserve capacities for users provide an implicit isolation guarantee since  $k_i$  cache slots are reserved for pages of user  $i$ . We remark that when the reserve capacities are all zero, then the problem reduces to classical unweighted caching.

A similar issue arises in the multi-processor setting where we have different “levels” of caches. Lower-level caches tend to be smaller and dedicated to a particular processor, while higher-level caches can be used by multiple processors and are larger in size. Consider a system with  $m$  separate processors, each of which has its own independent cache. In addition, there is a separate public cache shared by all the processors. We model such a setting as the *public-private caching* problem where a cache of total size  $k$  is partitioned into  $(m + 1)$  subcaches, one private cache for each user and a shared public cache. In contrast with classical caching, in this case cache slots themselves have identities and a page requested by user  $i$  cannot be placed in a cache slot that belongs to the private cache of some other user  $j$ .

## 1.1 Our Contributions

We propose and study the *caching with reserves* and *public-private caching* problems. We show that the two problems are equivalent up to constant factors (Section 3).

► **Proposition 1.** *If  $\mathcal{A}$  is a  $c$ -competitive online algorithm for caching with reserves, then there exists an online algorithm  $\mathcal{A}'$  that is  $2c$ -competitive for public-private caching. Similarly, if  $\mathcal{B}$  is a  $c$ -competitive online algorithm for public-private caching, then there exists an online algorithm  $\mathcal{B}'$  that is  $2c$ -competitive for caching with reserves.*

Our next set of results considers the *offline* scenario where the entire request sequence is known in advance. Recall that in the classical setting, there is a simple exact solution (Belady’s algorithm [8], which evicts the page that is requested farthest in the future). In our more complex setting, we show both variations are NP-hard.

► **Theorem 2.** *Both the offline caching with reserves problem and the offline public-private caching problem are strongly NP-hard.*

We defer the full proof of the NP-hardness to the full version of the paper [11] and note here the key difficulty in our reduction from 3-SAT. A naive strategy to reduce 3-SAT to our problem is to try to transform boolean variable assignments (e.g.  $x_1 = T, x_2 = F$ ) into the contents of cache at a particular point in time (e.g., agent 1 has its “true” page in cache and agent 2 has its “false” page in cache). This runs into a stumbling block: to check that a clause is satisfied, one needs to request the relevant pages. Since we only expect one of them to actually be in cache, this provides the opportunity for a cheating solution to swap the contents of cache. Our construction sidesteps this issue by embracing page swapping and instead demanding that a variable assignment be encoded as a particular sequence of page swaps.

Despite this hardness result, we still provide constant-approximation algorithms in the offline setting. Due to the equivalence of the two models, we focus on *caching with reserves* problem for the rest of the paper. We give a 2-approximation algorithm in Section 4 for the offline setting. It is a non-trivial adaptation of Belady’s algorithm to the multi-agent setting. The analysis utilizes a potential function that was recently proposed to give an alternative proof of optimality for Belady’s algorithm [6]. It tracks how far in the future the cached pages are for the algorithm vs. the optimum.

► **Theorem 3.** *There is a 2-approximation algorithm for offline caching with reserves.*

In the *online* scenario, where the algorithm knows nothing about page requests until they occur, we give a fractional algorithm (which may keep pages fractionally in cache) using the primal-dual framework (Section 5).

► **Theorem 4.** *There is a  $2\ln(k+1)$ -competitive fractional algorithm for online caching with reserves.*

We also show that the fractional solution can be rounded online in a way that preserves the competitive ratio up to a constant, obtaining an online randomized (integral) algorithm (Section 6).

► **Theorem 5.** *There is an  $O(\ln k)$ -competitive integral algorithm for online caching with reserves.*

## 2 Preliminaries and Notation

In the classical caching problem, we are given  $\mathcal{U}$ , a universe of  $n$  pages, together with a cache of size  $k$ . At each time step, at most  $k$  pages are in cache. We are presented a sequence of page requests  $\sigma = \langle p_1, p_2, \dots \rangle$ , where each  $p_t \in \mathcal{U}$ . At time  $t$ , page  $p_t$  arrives. If  $p_t$  is not in cache, then a *cache miss* occurs and the algorithm incurs unit cost. It must then fetch page  $p_t$  into the cache, possibly by evicting some other page from the cache. That is, if there would be  $k+1$  pages in cache, the algorithm must remove some page other than  $p_t$  from cache. An *online* algorithm makes the eviction choice without knowing the future request sequence, whereas an *offline* algorithm is assumed to know the entire request sequence in advance.

Motivated by applications in multi-processor caching and shared cache systems, we define two new related problems. Let  $\mathcal{I} = \{1, \dots, m\}$  be a set of  $m$  agents and suppose that the universe  $\mathcal{U}$  is a disjoint union of pages belonging to each agent, i.e.,  $\mathcal{U} = \sqcup_{i \in \mathcal{I}} \mathcal{U}(i)$ . In the

*public-private caching* model, the cache of total size  $k$  is subdivided as follows: each agent  $i \in \mathcal{I}$  is allocated  $k_i$  cache slots and the remaining  $k_0 \triangleq k - \sum_{i \in \mathcal{I}} k_i$  slots are *public*.\* In this model, only pages belonging to agent  $i$  can be placed in any of the  $k_i$  cache slots allocated to agent  $i$ , while any page can be held in the public slots. As in the traditional caching problem, the goal of the algorithm is to minimize the total number of evictions. In the *caching with reserves* model, the cache is not divided, but instead for each agent  $i \in \mathcal{I}$ , the algorithm is required to maintain at least  $k_i$  pages from  $\mathcal{U}(i)$  in the cache at any time. To avoid any complications, we assume that we already have pages in cache that meet this constraint at the start of the algorithm. (These may be dummy pages that are never requested during the actual sequence.) Throughout, we let  $n_i = |\mathcal{U}(i)|$  denote the number of distinct pages owned by agent  $i$ , and for any page  $p \in \mathcal{U}(i)$ , we let  $ag(p) = i$  be the agent that owns page  $p$ .

We analyze the online algorithm in terms of its *competitive ratio*. This is the maximum ratio, over all possible problem instances, of the cost incurred by the algorithm over the cost of the optimal offline solution of this instance.

### 3 Equivalence of Public-Private Caching and Caching with Reserves

We now prove Proposition 1 (restated below for convenience), showing that the two models defined in the introduction are equivalent up to constant factors.

► **Proposition 1.** *If  $\mathcal{A}$  is a  $c$ -competitive online algorithm for caching with reserves, then there exists an online algorithm  $\mathcal{A}'$  that is  $2c$ -competitive for public-private caching. Similarly, if  $\mathcal{B}$  is a  $c$ -competitive online algorithm for public-private caching, then there exists an online algorithm  $\mathcal{B}'$  that is  $2c$ -competitive for caching with reserves.*

**Proof.** We first explain how to convert back-and-forth between caching strategies for the two problems. Note that both of the following conversions can be done online and we will maintain that the cache states in the two problems are identical after every page request.

The easy direction is turning a public-private caching strategy into a caching with reserves strategy. Suppose a page request  $p$  comes in. If  $p$  is in cache, then we do not evict any page in either strategy. If it is not, then the public-private caching strategy evicts some page  $q$  to make room for it. Our caching with reserves strategy makes precisely the same eviction, which we show maintains the reserve constraints:

- If evicted page  $q$  was in a private cache, then  $p$  is placed in that same private cache. Hence,  $p$  and  $q$  have the same agent  $i$ . And agent  $i$  has the same number of pages before and after the arrival of  $p$ , maintaining our reserve constraint.
- If  $q$  was in a public cache, then let  $i$  be the agent that owns  $q$ . Agent  $i$  must have at least  $k_i$  pages in cache other than  $q$ , namely the  $k_i$  pages in its private cache. So evicting  $q$  does not put agent  $i$  below its reserve for our caching with reserves algorithm.

We now turn to the harder case of turning a caching with reserves strategy into a public-private caching strategy. To keep the analysis clean, we permit the public-private caching strategy to perform extra evictions at any step (but it is still charged for each one). Suppose a page request  $p$  comes in. If  $p$  is in cache, then we do not evict in either strategy. If it is not, then the caching with reserves strategy evicts some page  $q$  to make room for it, which belongs to some agent  $i$ . We can handle this with at most two evictions, as the following shows:

---

\* We assume throughout the paper that  $\sum_{i \in \mathcal{I}} k_i < k$ . If  $\sum_{i \in \mathcal{I}} k_i = k$ , the problem can be solved as  $m$  separate instances of classical caching.

- If  $q$  is currently in the public cache, then we evict it and replace it with  $p$ , making the two caches match again.
- If  $q$  is currently in a private cache and the agent of  $p$  is also  $i$ , then we again can evict it and replace it with  $q$ , making the two caches match again.
- If  $q$  is currently in a private cache and the agent of  $p$  is not  $i$ , then agent  $i$  must have at least  $k_i + 1$  pages in cache at the start of this step since we're about to evict  $q$ . Hence, there must be some page  $q'$  in public cache. "Move"  $q'$  into private cache by evicting both  $q$  and  $q'$ , then placing  $q'$  into the slot previously occupied by  $q$ . We can then place  $p$  into the slot previously occupied by  $q'$  in public cache.

We now have conversions between the two problems that approximately preserve the number of evictions, and are ready to prove the main claim. We will use  $\tau_e$  to denote the first transformation, from public-private caching strategies into caching with reserves strategies. We will use  $\tau_h$  to denote the second transformation, from caching with reserves strategies to public-private caching strategies.

Suppose we have some algorithm  $\mathcal{A}$  for caching with reserves, and let  $\mathcal{A}' \triangleq \tau_h(\mathcal{A})$ . Furthermore, let the optimal solutions to caching with reserves and public-private caching be  $\mathcal{O}_{cr}$  and  $\mathcal{O}_{ppc}$ , respectively.

$$\begin{aligned}
 \text{evictions}(\mathcal{A}') &\leq 2 \cdot \text{evictions}(\mathcal{A}) && \text{Transformation Guarantee} \\
 &\leq 2c \cdot \text{evictions}(\mathcal{O}_{cr}) && \mathcal{A} \text{ is a } c\text{-approximation} \\
 &\leq 2c \cdot \text{evictions}(\tau_e(\mathcal{O}_{ppc})) && \mathcal{O}_{cr} \text{ Optimality} \\
 &\leq 2c \cdot \text{evictions}(\mathcal{O}_{ppc}) && \text{Transformation Guarantee}
 \end{aligned}$$

Similarly, suppose we have some algorithm  $\mathcal{B}$  for public-private caching and let  $\mathcal{B}' \triangleq \tau_e(\mathcal{B})$ . Again, let the optimal solutions to caching with reserves and public-private caching be  $\mathcal{O}_{cr}$  and  $\mathcal{O}_{ppc}$ , respectively.

$$\begin{aligned}
 \text{evictions}(\mathcal{B}') &\leq \text{evictions}(\mathcal{B}) && \text{Transformation Guarantee} \\
 &\leq c \cdot \text{evictions}(\mathcal{O}_{ppc}) && \mathcal{B} \text{ is a } c\text{-approximation} \\
 &\leq c \cdot \text{evictions}(\tau_h(\mathcal{O}_{cr})) && \mathcal{O}_{ppc} \text{ Optimality} \\
 &\leq 2c \cdot \text{evictions}(\mathcal{O}_{cr}) && \text{Transformation Guarantee}
 \end{aligned}$$

This completes the proof. ◀

## 4 Offline Caching with Reserves

In this section, we present a 2-approximation algorithm for the offline caching with reserves problem. The algorithm itself can be thought of as a generalization to Belady's classic Farthest-in-Future algorithm [8]. Indeed, the algorithm we present reduces to it in the trivial case that  $k_i = 0$  for all  $i$ . However, in general, in our setting, there are cases where the farthest-in-future page cannot be evicted due to the reserve constraints.

Our algorithm maintains a partition of the pages in cache into sets  $N_i$ . For  $i > 0$ , the set  $N_i$  consists only of pages for agent  $i$ ; further, we maintain  $|N_i| = k_i$  at the beginning of each time step. The set  $N_0$  contains the remaining cached pages. When a page  $p$  associated with agent  $i$  arrives and is not already in cache, we insert it into  $N_i$ . This causes  $|N_i| = k_i + 1$ , so we move the farthest-in-future page from  $N_i$  to  $N_0$ . This, in turn, causes  $N_0$  to be too large. So we evict the farthest-in-future page from  $N_0$ . Notice that we are always allowed to evict such a page, since we maintain  $k_i$  pages of agent  $i$  in each  $N_i$ . In the case that  $p$  arrives but is already in  $N_0$ , we first move it to  $N_i$ , then proceed similarly. In this way, an arriving page always "passes through"  $N_i$ . The full details are in Algorithm 1.



---

**Algorithm 1** Offline algorithm for caching with reserves.

---

```

Let $N \leftarrow$ set of pages in the cache initially
Partition $N = \sqcup_{i=0}^m N_i$ where each N_i (for $i \neq 0$) contains some arbitrary k_i pages belonging
to agent i and N_0 contains all the remaining pages
Set $\text{rank}(q)$, for each page q , to the time of q 's first request
for each requested page p do
 Let $i = \text{ag}(p)$
 if $p \in N_i$ then /* Cache hit in a set reserved for i . */
 Serve page p from cache
 else if $p \in N_0$ then /* Cache hit in a set not reserved for i . */
 Serve page p from cache
 /* Move p from N_0 to N_i . */
 $N_i \leftarrow N_i \cup \{p\}$ and $N_0 \leftarrow N_0 \setminus \{p\}$
 /* Move highest-ranked page from N_i to N_0 . */
 Let $q_i \in N_i$ be the page in N_i with maximum rank (if $k_i = 0$, this will be p)
 $N_i \leftarrow N_i \setminus \{q_i\}$ and $N_0 \leftarrow N_0 \cup \{q_i\}$
 else /* Cache miss. */
 /* Add p to N_i , then move highest-ranked page from N_i to N_0 . */
 $N_i \leftarrow N_i \cup \{p\}$
 Let $q_i \in N_i$ be the page in N_i with maximum rank (if $k_i = 0$, this will be p)
 $N_i \leftarrow N_i \setminus \{q_i\}$ and $N_0 \leftarrow N_0 \cup \{q_i\}$
 /* Evict highest-ranked page from N_0 . */
 Let q be the page in N_0 with maximum rank ($q \neq p$ even if $q_i = p$)
 $N_0 \leftarrow N_0 \setminus \{q\}$
 Evict page q , fetch page p into cache and serve it
 Set $\text{rank}(p)$ to the time of p 's next request (if none, set it later than the last request)

```

---

Our analysis proving the 2-approximation generalizes a potential argument for Belady's algorithm (proposed recently [6]), but is technically more complicated due to the multi-tiered approach we take. The proof compares our sets  $N_i$  with sets  $N_i^*$  for the optimal algorithm. (To be more precise, the optimal algorithm maintains a certain set of pages in cache at each time step. We define a partition of these pages into the  $N_i^*$  such that each  $N_i^*$  consists only of pages from agent  $i$ , and  $|N_i^*| = k_i$  at the beginning of each time step.) We call any page's next request time its *rank*. We define, for any rank  $s$ , the value  $n_i(s)$  to be the number of pages in the set  $N_i$  with rank at least  $s$  at a given time. Similarly,  $n_i^*(s)$  is the number of pages in the set  $N_i^*$  with rank at least  $s$ .<sup>†</sup>

We define our potential function as

$$\Phi = \sum_{i=0}^m \phi_i, \quad \text{where } \phi_i = \max_s [n_i(s) - n_i^*(s)].$$

Notice that  $\phi_i \geq 0$  for every  $i$ , because when  $s$  is larger than the rank of any page in cache, we have  $n_i(s) = n_i^*(s) = 0$ . Hence  $\Phi \geq 0$ .

We show that Algorithm 1 satisfies the requirements of Theorem 3 (restated below).

► **Theorem 3.** *There is a 2-approximation algorithm for offline caching with reserves.*

---

<sup>†</sup> The sets  $N_i$  and  $N_i^*$  and the quantities  $n_i(s)$  and  $n_i^*(s)$  vary over time, but we suppress the dependence on  $t$  in the notation for brevity.

The proof requires repeated reasoning about how the potential  $\Phi$  changes with each step. For example, adding a page to  $N_i$  will increase  $\phi_i$  by at most 1 (and possibly leave it unchanged). However, adding a page  $p$  to  $N_i$  whose rank is higher than anything in  $N_i^*$  guarantees that  $\phi_i$  will increase by exactly 1 (since  $n_i(s)$  increases by 1 for every  $s \leq \text{rank}(p)$ ).

Initially let  $N_i^* = N_i$  for all  $i$  from 0 to  $m$  (the sets  $N_i$  are initialized by Algorithm 1). Let  $ALG$  be the cost incurred by Algorithm 1 and  $OPT$  be the cost incurred by an optimal algorithm. Let  $\Delta(ALG)$ ,  $\Delta(\Phi)$ ,  $\Delta(OPT)$  be incremental changes in  $ALG$ ,  $\Phi$ ,  $OPT$ , respectively, with older value subtracted from the newer value.

► **Lemma 6.** *The runs of Algorithm 1 and of the optimal algorithm on a given sequence of page requests can be partitioned into steps such that for each step,  $\Delta(ALG) + \Delta(\Phi) \leq 2 \cdot \Delta(OPT)$ .*

Knowing Lemma 6, the approximation factor of 2 now follows from summing over all the incremental steps indexed by  $t$ , where  $\cdot(t)$  is the value of each function after step  $t$ . We have  $ALG(0) = \Phi(0) = OPT(0) = 0$  initially. By Lemma 6, for each  $t$ ,

$$ALG(t) - ALG(t-1) + \Phi(t) - \Phi(t-1) \leq 2 \cdot (OPT(t) - OPT(t-1)).$$

Summing over all  $t$  (up to the last step  $T$ ) and telescoping,

$$\begin{aligned} ALG(T) - ALG(0) + \Phi(T) - \Phi(0) &\leq 2 \cdot (OPT(T) - OPT(0)) \\ ALG(T) &\leq 2 \cdot OPT(T), \end{aligned}$$

where the last inequality uses  $\Phi(T) \geq 0$ .

**Proof of Lemma 6.** To prove Lemma 6, we break the runs of Algorithm 1 and the optimal algorithm (together with updates to sets  $N_i^*$ ) into steps, and for each step show that  $\Delta(ALG) + \Delta(\Phi) \leq 2 \cdot \Delta(OPT)$ . All the steps below constitute the processing of one request for a page  $p$  belonging to agent  $i$ . Let  $\delta_i(s) = n_i(s) - n_i^*(s)$ , so that  $\phi_i = \max_s \delta_i(s)$ .

### Step 1 (Add $p$ to both $N_i$ and $N_i^*$ )

Update  $N_i \leftarrow N_i \cup \{p\}$  and  $N_i^* \leftarrow N_i^* \cup \{p\}$ .

Neither  $ALG$  nor  $OPT$  changes in this step, since we don't evict anything. In addition, the potential  $\Phi$  doesn't increase. To see this, we'll use the fact that the rank of  $p$  is the smallest among any page in cache (for our algorithm as well as for the optimal algorithm), since it is the page that has just arrived. We consider four cases based on whether  $N_i$  and  $N_i^*$  contained  $p$  before this step.

- If both  $N_i$  and  $N_i^*$  contained  $p$  already, then nothing changes.
- If neither contained it, then both  $n_i(s)$  and  $n_i^*(s)$  increase by 1 for all  $s \leq \text{rank}(p)$ , so their difference is unchanged.
- If  $p$  was newly added only to  $N_i^*$ , then  $\Phi$  can only decrease.
- The remaining case is that  $p$  was newly added only to  $N_i$ . Note that since  $p$  is the page that was just requested (and its rank hasn't been updated to the next occurrence yet), it has the minimum rank of all pages. We prove that  $\Phi$  doesn't increase by showing that before this step,  $\phi_i \geq 1$ , and after this step, any  $\delta_i(\cdot)$  that might have changed are at most 1. Specifically, before this step,  $|N_i| = |N_i^*| = k_i$ . Since  $N_i$  did not contain  $p$ , and all other pages have higher rank, before this step we had  $n_i(\text{rank}(p) + 1) = k_i$ . Since  $N_i^*$  contained  $p$ , we had  $n_i^*(\text{rank}(p) + 1) = k_i - 1$ . Thus, before this step,  $\phi_i \geq \delta_i(\text{rank}(p) + 1) = 1$ . After this step,  $n_i(s) = k_i + 1$ ,  $n_i^*(s) = k_i$ , and  $\delta_i(s) = 1$  for  $s \leq \text{rank}(p)$  (and  $\delta_i(s)$  is unchanged for  $s > \text{rank}(p)$ ). Thus,  $\Phi$  doesn't increase.

**Step 2 (Remove  $p$  from both  $N_0$  and  $N_0^*$ )**

Update  $N_0 \leftarrow N_0 \setminus \{p\}$  and  $N_0^* \leftarrow N_0^* \setminus \{p\}$ .

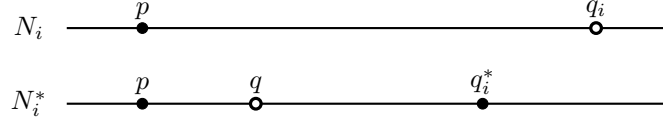
Again,  $ALG$  and  $OPT$  don't change since we make no evictions. Further, removing  $p$  – the lowest-ranked page in cache for both our algorithm and the optimal algorithm – does not increase  $\Phi$ ; the reasoning is similar to above.

- If neither  $N_0$  nor  $N_0^*$  changes, then  $\Phi$  remains the same.
- If  $p$  is newly removed from both, then  $n_0(s)$  and  $n_0^*(s)$  decrease by 1 for all  $s \leq \text{rank}(p)$ , and  $\delta_0(s)$  for all  $s$  are unchanged.
- If  $p$  is newly removed only from  $N_0$ ,  $\Phi$  can only decrease.
- The remaining case is that  $p$  was newly removed only from  $N_0^*$ . Before this step,  $|N_0| = |N_0^*| = k_0$ . Since  $p$  is the page with minimum rank, before the step,  $n_0(s) = n_0^*(s) = k_0$  for  $s \leq \text{rank}(p)$ . Also, since before the step  $p \notin N_0$  and  $p \in N_0^*$ , we had  $n_0(\text{rank}(p) + 1) = k_0$  and  $n_0^*(\text{rank}(p) + 1) = k_0 - 1$ , implying  $\Phi \geq \delta_0(\text{rank}(p) + 1) = 1$ . After the removal of  $p$ ,  $n_0(s) = k_0$ ,  $n_0^*(s) = k_0 - 1$  and  $\delta_0(s) = 1$  for  $s \leq \text{rank}(p)$ . Thus,  $\Phi$  doesn't increase.

**Step 3 (Ensure  $|N_i| = |N_i^*| = k_i$ )**

In Step 1, we added  $p$  to  $N_i$  (resp.,  $N_i^*$ ). If it wasn't already there, we increased the size by 1. If that happened, then in this step, we move a page from  $N_i$  to  $N_0$  to ensure  $|N_i| = k_i$  (resp., move from  $N_i^*$  to  $N_0^*$  to ensure  $|N_i^*| = k_i$ ). Let  $q_i$  be the page in  $N_i$  with maximum rank. If  $|N_i| = k_i + 1$ , then  $q_i$  is moved to  $N_0$ , consistent with Algorithm 1. We choose which page to move from  $N_i^*$  to  $N_0^*$  based on the cases below. It could be the page  $p$  itself if it is the only one available, the page  $q \in N_i^*$  with minimum rank other than  $p$  (so it actually has the second-minimum rank in  $N_i^*$ ), or the page  $q_i^* \in N_i^*$  with maximum rank.  $ALG$  and  $OPT$  don't change in this step, and in each case we show that  $\Phi$  doesn't increase.

- If  $k_i = 0$ , then  $N_i = N_i^* = \{p\}$ . Move  $p$  from  $N_i$  to  $N_0$  and from  $N_i^*$  to  $N_0^*$ .  
 $\Phi$  is unaffected in this case because for any  $s$ ,  $n_i(s)$  changes by the same amount as  $n_i^*(s)$ , and  $n_0(s)$  changes by the same amount as  $n_0^*(s)$ .  
 All the cases below assume that  $k_i > 0$ .
- If  $|N_i| = k_i + 1$  but  $|N_i^*| = k_i$ , move  $q_i$  from  $N_i$  to  $N_0$ .  
 We show that when  $q_i$  is removed from  $N_i$ ,  $\phi_i$  decreases by 1. Since  $N_i$  had more pages than  $N_i^*$ , before this step  $\phi_i \geq 1$ . Also before this step,  $\delta_i(s) \leq 0$  for  $s > \text{rank}(q_i)$  (since  $n_i(s) = 0$  for those  $s$ ), so the maximum was not achieved for those values of  $s$ . And for  $s \leq \text{rank}(q_i)$ ,  $\delta_i(s)$  decreases by 1 after this step, leading to the decrease of  $\phi_i$ . Now, when  $q_i$  is added to  $N_0$ ,  $\phi_0$  increases by at most 1. But this is compensated by the decrease in  $\phi_i$ , showing that overall  $\Phi$  doesn't increase.
- If  $|N_i| = k_i$  but  $|N_i^*| = k_i + 1$ , move the second-lowest-ranked page  $q \in N_i^*$  to  $N_0^*$ . Note that by our assumption that  $k_i > 0$ ,  $N_i^*$  has at least two pages.  
 Adding a page to  $N_0^*$  can only decrease the potential. Now we consider the effect on  $\phi_i$  of removing  $q$  from  $N_i^*$ . We show that for any  $s$  for which  $\delta_i(s)$  could have changed, it was negative before this step. For any  $s > \text{rank}(q)$ ,  $\delta_i(s)$  doesn't change. Note that page  $p$  has minimum rank in both  $N_i$  and  $N_i^*$ . So, before this step, for  $s \leq \text{rank}(p)$ ,  $n_i^*(s) = |N_i^*| = k_i + 1$  and  $n_i(s) = |N_i| = k_i$ , so  $\delta_i(s) < 0$ . For  $s \in (\text{rank}(p), \text{rank}(q)]$ ,  $n_i^*(s) = k_i$  and  $n_i(s) \leq k_i - 1$ , so again  $\delta_i(s) < 0$ . Thus when  $\delta_i(s)$  for  $s \leq \text{rank}(q)$  increases by 1, it remains at most 0, and does not increase  $\Phi$  (which is always at least 0).
- Recall that  $q_i \in N_i$  and  $q_i^* \in N_i^*$  are the pages with maximum ranks in the respective sets. If  $|N_i| = |N_i^*| = k_i + 1$  and  $\text{rank}(q_i) \leq \text{rank}(q_i^*)$ , move  $q_i$  from  $N_i$  to  $N_0$  and  $q_i^*$  from  $N_i^*$  to  $N_0^*$ .



■ **Figure 1** Illustration for the last case of Step 3 in the proof of Lemma 6.

We first consider the removal of  $q_i$  from  $N_i$  and of  $q_i^*$  from  $N_i^*$ . For  $s \leq \text{rank}(q_i)$ , both  $n_i(s)$  and  $n_i^*(s)$  decrease by 1, so  $\delta_i(s)$  doesn't change. For  $s > \text{rank}(q_i^*)$ ,  $n_i(s)$ ,  $n_i^*(s)$ , and  $\delta_i(s)$  are unchanged. For  $s \in (\text{rank}(q_i), \text{rank}(q_i^*)]$ , before this step we had  $n_i(s) = 0$  and  $n_i^*(s) \geq 1$ , with  $\delta_i(s) \leq -1$ . So increasing  $\delta_i(s)$  by 1 for these  $s$  does not change  $\Phi$ . Now we consider the addition of  $q_i$  to  $N_0$  and of  $q_i^*$  to  $N_0^*$ . For any  $s$ ,  $n_0^*(s)$  increases at least as much as  $n_0(s)$  does, so  $\Phi$  does not increase.

- If  $|N_i| = |N_i^*| = k_i + 1$  and  $\text{rank}(q_i^*) < \text{rank}(q_i)$ , move  $q_i$  to  $N_0$  and the second-lowest-ranked page in  $N_i^*$  (call it  $q$ ) to  $N_0^*$ . Note again that  $N_i^*$  has at least two pages.

In this case  $\phi_0$  may increase by 1, but we show that this is offset by a decrease in  $\phi_i$ . We analyze what happens for values of  $s$  in the intervals separated by three values:  $\text{rank}(p) < \text{rank}(q) < \text{rank}(q_i)$  (see Figure 1). Before this step,  $\delta_i(\text{rank}(q_i)) = n_i(\text{rank}(q_i)) - n_i^*(\text{rank}(q_i)) = 1 - 0 = 1$ , so  $\phi_i \geq 1$ . Page  $p$  is the page with minimum rank in both  $N_i$  and  $N_i^*$ . For  $s \leq \text{rank}(p)$ , before the step  $\delta_i(s) = 0$ , and it stays 0 after the step. For  $s \in (\text{rank}(p), \text{rank}(q)]$ , before the step  $n_i^*(s) = |N_i^*| - 1 = k_i$  and  $n_i(s) \leq |N_i| - 1 = k_i$ , so  $\delta_i(s) \leq 0$ , and it stays that way. For  $s > \text{rank}(q_i)$ , also  $\delta_i(s) = 0$  and stays 0. Thus, the maximum  $\delta_i(s)$  was achieved for some  $s \in (\text{rank}(q), \text{rank}(q_i))$ . But in this interval,  $n_i(s)$  decreases by 1, while  $n_i^*(s)$  stays the same. Thus, the maximum  $\delta_i(s)$  decreases by 1, causing  $\phi_i$  to also decrease.

#### Step 4 (OPT moves)

If  $p$  was in cache, then the optimal algorithm doesn't do anything. Note that in this case, based on previous rearrangements,  $|N_0^*| = k_0$ . Neither  $\text{OPT}$  nor  $\Phi$  changes. If  $p$  was not in cache, the optimal algorithm fetches  $p$  and evicts some page, say  $q \in N_j^*$ . Then  $\Delta(\text{OPT}) = 1$ . Also note that in this case the previous steps added  $p$  to  $\bigcup_{\ell} N_{\ell}^*$ , resulting in  $|N_0^*| = k_0 + 1$ . If  $j = 0$ , delete  $q$  from  $N_0^*$ . This restores  $|N_0^*| = k_0$  and increases  $\Phi$  by at most 1. If  $j \neq 0$ , then there must be some  $q' \in N_0^*$  belonging to agent  $j$  (otherwise it would mean that agent  $j$  had only  $k_j$  pages in cache, and the optimal algorithm violated reserve sizes by evicting agent  $j$ 's page). Move  $q'$  from  $N_0^*$  to  $N_j^*$  and delete  $q$  from  $N_j^*$ . This increases  $\Phi$  by at most 2, satisfying the desired inequality.

#### Step 5 (ALG moves)

If  $p$  was in cache, then do nothing. Otherwise, fetch  $p$  and evict the page  $q$  with maximum rank in  $N_0$ , also deleting it from  $N_0$ . In this case,  $\Delta(\text{ALG}) = 1$ . We show that this is compensated by  $\Delta(\Phi) = -1$ . Before this step, we had  $|N_0| = k_0 + 1$  but  $|N_0^*| = k_0$ , so  $\phi_0 \geq 1$ . For  $s > \text{rank}(q)$ , we had  $\delta_0(s) \leq 0$ , and this doesn't change. So the maximum must have been achieved for  $s \leq \text{rank}(q)$ , and  $\delta_0(s)$  for those  $s$  decreases by 1.

#### Step 6 (Update the rank of $p$ )

At this point, if  $k_i = 0$ , then  $p \in N_0 \cap N_0^*$ ; otherwise,  $p \in N_i \cap N_i^*$ . In either case, changing  $\text{rank}(p)$  preserves  $\delta_0(s)$  and  $\delta_i(s)$  for all  $s$ , so  $\Phi$  is unchanged. ◀

This completes the proof of Lemma 6 and the proof of Theorem 3.

## 5 Online Caching with Reserves

In this section, we design an  $O(\log k)$ -competitive fractional online algorithm for caching with reserves. In particular, we prove Theorem 4, which is restated here for convenience. In Section 6, we show that any fractional algorithm for online caching with reserves can be *rounded* to obtain a randomized integral algorithm by losing only a constant factor in the competitive ratio. We remark that our rounding algorithm does not necessarily run in polynomial time.

► **Theorem 4.** *There is a  $2 \ln(k+1)$ -competitive fractional algorithm for online caching with reserves.*

The fractional algorithm is based on the primal-dual framework and closely follows the analysis of [4]. As page requests arrive, the algorithm maintains a feasible solution to the primal LP, which corresponds to its eviction decisions, and an approximately feasible solution to the dual LP. The costs of these two solutions are within a factor 2 of each other. Using LP duality, this results in a bound on the cost of the algorithm compared to the optimum.

### 5.1 Notation

Consider some fixed page  $p \in \mathcal{U}$ , and let  $t_{p,1} < t_{p,2} < \dots$  be the time steps when page  $p$  is requested in the online sequence. For any  $a \geq 0$ , define  $I(p, a) = \{t_{p,a} + 1, \dots, t_{p,a+1} - 1\}$  to be the time interval between the  $a$ th and  $(a+1)$ st requests for page  $p$  (assume that  $t_{p,0} = 0$  for all pages). Let  $a(p, t)$  be the number of requests to page  $p$  that have been seen until time  $t$  (inclusive). Hence, by definition, for any time  $t$ , and any page  $p \in \mathcal{U} \setminus \{p_t\}$ , we have  $t \in I(p, a(p, t))$ . At any time  $t$ , an agent  $i \in \mathcal{I}$  is said to be *tight* if exactly  $k_i$  pages of agent  $i$  are held in cache. Let  $\mathcal{T}$  denote the set of tight agents.<sup>‡</sup>

### 5.2 Formulation

We use the variable  $x(p, a) \in \{0, 1\}$  to denote whether page  $p$  is evicted between its  $a$ th and  $(a+1)$ th request, i.e., in the interval  $I(p, a)$  (where 1 denotes an eviction). We have the following linear programming relaxation and its dual formulation.<sup>§</sup>

| Primal LP                                                                                                                                                                                                                                                                                                                                                   | Dual LP                                                                                                                                                                                                                                                                      |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\min \sum_{p \in \mathcal{U}} \sum_{a \geq 1} x(p, a)$ <p>subject to:</p> $\sum_{p \in \mathcal{U}, p \neq p_t} x(p, a(p, t)) \geq n - k \quad \forall t \quad (1)$ $\sum_{p \in \mathcal{U}(i), p \neq p_t} x(p, a(p, t)) \leq n_i - k_i \quad \forall t, \forall i \quad (2)$ $x(p, a) \leq 1 \quad \forall p, \forall a \quad (3)$ $x \geq 0 \quad (4)$ | $\max \sum_t (n - k) \alpha(t) - \sum_{t, i} (n_i - k_i) \beta(t, i) - \sum_{p, a} \gamma(p, a)$ <p>subject to:</p> $\sum_{t \in I(p, a)} (\alpha(t) - \beta(t, ag(p))) - \gamma(p, a) \leq 1 \quad \forall p, \forall a \quad (5)$ $\alpha, \beta, \gamma \geq 0 \quad (6)$ |

<sup>‡</sup> The set of tight agents varies with the time  $t$ , but we suppress the dependence on  $t$  for convenience.

<sup>§</sup> We assume without loss of generality that the algorithm is aware of the total number of pages belonging to each agent,  $n_i = |\mathcal{U}(i)|$ .

The primal objective simply measures the total number of evictions. The first constraint enforces that at any time  $t$  at least  $n - k$  pages apart from  $p_t$  are outside the cache, which implies that at most  $k$  pages (including  $p_t$ ) are inside the cache. The second constraint enforces that at any time, at most  $(n_i - k_i)$  pages of agent  $i$  are outside cache (which implies that at least  $k_i$  pages are inside the cache). Note that this is true even if  $p_t \in \mathcal{U}(i)$ , since then we know that  $p_t$  must be in cache, so of the remaining  $n_i - 1$  pages, at least  $k_i - 1$  must be in the cache, so the total amount outside cache must be at most  $(n_i - 1) - (k_i - 1) = n_i - k_i$ .

### 5.3 Algorithm

For convenience, we assume without loss of generality that the cache is initialized to an arbitrary feasible configuration, i.e., each agent  $i$  has some arbitrary  $k_i$  pages in the cache, and the rest of the cache has  $k_0$  other arbitrary pages. The LP variables are also initialized to reflect this initial configuration. At each time step, as a new page request arrives online, a new set of constraints for the primal LP are revealed, along with the corresponding new variables in the dual. All newly introduced variables are initialized to zero. Note that after the arrival of a new page request at time  $t$ , only the primal constraint (1) may now be unsatisfied; however, (2) and (3) remain feasible. So to maintain a feasible primal solution, we modify the primal (and dual) variables until Constraint (1) is satisfied. The online algorithm is required to maintain that all the primal variables  $x(p, a)$  only monotonically increase over time. We remark that the dual solution that we maintain will always be approximately feasible. The violation in (5) is at most  $O(\log k)$  at all times (Claim 8).

■ **Algorithm 2** Fractional Online Algorithm for Caching with Reserves.

---

```

Let $\eta \leftarrow \frac{1}{k}$
foreach request for page p at time t do
 Initialize $x(p, a(p, t)) \leftarrow 0, \alpha(t) \leftarrow 0, \gamma(p, a(p, t)) \leftarrow 0$ and $\forall i \in \mathcal{I}, \beta(t, i) \leftarrow 0$
 while primal constraint (1) is unsatisfied do
 Increase dual variable $\alpha(t)$ by $d\alpha$
 foreach tight agent $i \in \mathcal{T}$ do
 Increase dual variable $\beta(t, i)$ by $d\alpha$
 foreach page $q \in \mathcal{U}$ do
 if $ag(q) \in \mathcal{T}$ then
 Do nothing
 else if $x(q, r(q, t)) = 1$ then
 Increase $\gamma(q, r(q, t))$ by $d\alpha$
 else
 Increase $x(q, r(q, t))$ by $dx = (x(q, r(q, t)) + \eta)d\alpha$

```

---

### 5.4 Analysis

First, we note that the primal solution that we construct is feasible by design.

▷ **Claim 7.** At all times  $t$ , we maintain the inequality: Primal Objective  $\leq 2 \cdot$  Dual Objective.

**Proof.** At time  $t = 0$ , both the primal and dual solutions are initialized to have an objective of zero. Since the algorithm increases the primal and dual variables in a continuous fashion, consider any infinitesimal time step and let  $\Delta P$  and  $\Delta D$  denote the change in the primal and dual objectives in this step respectively. It suffices to show that  $\Delta P \leq 2 \cdot \Delta D$  holds at all times.

Let  $\mathcal{T}$  denote the set of agents that are tight during this step. Also partition the set  $\mathcal{U} \setminus \{p\}$  into three parts:  $T$  is the set of pages belonging to tight agents,  $E = \{q \in \mathcal{U} \setminus T \mid x(q, r(q, t)) = 1\}$  is the set of pages of non-tight agents that have been fully evicted, and  $S$  is the remaining set of pages. So we have  $|T| + |S| + |E| = n - 1$ , and  $|T| = \sum_{i \in \mathcal{T}} n_i$ . We also define  $k' := k - \sum_{i \in \mathcal{T}} k_i$ .

The change in the dual objective is given by:

$$\begin{aligned} \Delta D &= (n - k)d\alpha - \sum_{i \in \mathcal{T}} (n_i - k_i)d\alpha - |E|d\alpha = \left( n - k - |T| + \sum_{i \in \mathcal{T}} k_i - |E| \right) d\alpha \\ &= \left( |S| - \left( k - \sum_{i \in \mathcal{T}} k_i \right) + 1 \right) d\alpha = (|S| - k' + 1)d\alpha \end{aligned}$$

On the other hand, the change in primal objective is given by:

$$\begin{aligned} \Delta P &= \sum_{q \in S} (x(q, r(q, t)) + \eta) d\alpha \\ &= \left( \sum_{q \in \mathcal{U} \setminus \{p\}} x(q, r(q, t)) - \sum_{q \in T} x(q, r(q, t)) - \sum_{q \in E} x(q, r(q, t)) + |S|\eta \right) d\alpha \end{aligned}$$

Since the variables are updated only as long as constraint (1) is not satisfied, we can bound the first term in the above expression by  $n - k$ . All pages in  $T$  belong to tight agents, so we have  $\sum_{q \in T} x(q, r(q, t)) = \sum_{i \in \mathcal{T}} (n_i - k_i)$ . Lastly, all pages in  $E$  have  $x(q, r(q, t)) = 1$ . So

$$\begin{aligned} \Delta P &\leq \left( n - k - \sum_{i \in \mathcal{T}} (n_i - k_i) - |E| + |S|\eta \right) d\alpha = \left( |S| - \left( k - \sum_{i \in \mathcal{T}} k_i \right) + 1 + |S|\eta \right) d\alpha \\ &\leq (|S| - k' + 1 + |S|/k') d\alpha \quad (\text{since } \eta = 1/k \leq 1/k') \\ &\leq 2(|S| - k' + 1)d\alpha = 2 \cdot \Delta D \end{aligned}$$

It remains to justify the final inequality, which is equivalent to showing that  $|S| \geq k'$ . By definition, we have  $|S| = n - 1 - |E| - |T|$ . Since (1) is violated and (2) is tight for  $i \in \mathcal{T}$ , the following strict inequality holds:

$$\sum_{q \in S} x(q, r(q, t)) + |E| + \sum_{i \in \mathcal{T}} (n_i - k_i) = \sum_{q \in S \cup T \cup E} x(q, r(q, t)) < n - k.$$

Combining the above, we get  $|S| > k' - 1$ , which implies that  $|S| \geq k'$ .  $\triangleleft$

$\triangleright$  **Claim 8.** Dual solution maintained by the algorithm is  $\ln(k + 1)$ -approximately feasible.

**Proof.** Consider any page  $p$  and interval  $I(p, a) = \{t_{p,a} + 1, \dots, t_{p,a+1} - 1\}$ . We show that the following inequality holds at all times:

$$\sum_{t \in I(p,a)} (\alpha(t) - \beta(t, ag(p))) - \gamma(p, a) \leq \ln(k + 1),$$

which implies dual feasibility of the solution  $(\alpha, \beta, \gamma)$  scaled down by a factor  $\ln(k + 1)$ .

We analyze the changes that occur in the LHS of the above inequality. We interpret the set  $I(p, a)$  in an online fashion: time  $t \in \{t_{p,a} + 1, \dots, t_{p,a+1} - 1\}$  is included in  $I(p, a)$  at the start of time step  $t$ . Note that  $x(p, a) = 0$  and the LHS is 0 at the start of time  $t_{p,a} + 1$ . Over time, as page requests  $p_t (\neq p)$  arrive during times  $t \in \{t_{p,a} + 1, \dots, t_{p,a+1} - 1\}$ , the LHS increases whenever the  $\alpha(t)$  variable increases, but there is no corresponding increase



in the  $\beta(t, ag(p))$  or  $\gamma(p, a)$  variables. We couple such increases to increases in the primal variable  $x(p, a)$ . Note that  $x(p, a)$  gets capped at 1, and after that  $\gamma(p, a)$  is coupled with  $\alpha(t)$ .

At any infinitesimal step, if some  $\alpha(t)$  increases by  $d\alpha$ , then we have one of three cases. *Case 1:* Agent  $ag(p)$  is tight and  $\beta(t, ag(p))$  increases by  $d\alpha$ ; *Case 2:*  $x(p, a) = 1$  and  $\gamma(p, a)$  increases by  $d\alpha$ ; *Case 3:*  $x(p, a)$  increases by  $dx = (x(p, a) + \eta)d\alpha$ . In the first two cases, the LHS does not change at all, while in the second case, the LHS changes by  $d\alpha$ . So overall

$$d(\text{LHS}) = \left( \frac{1}{x(p, a) + \eta} \right) dx(p, a)$$

A straightforward integration gives:

$$\begin{aligned} \text{LHS} &= \int_0^X \left( \frac{1}{x(p, a) + \eta} \right) dx(p, a) && (\text{where } X \text{ is the final value of } x(p, a)) \\ &\leq \int_0^1 \left( \frac{1}{x(p, a) + \eta} \right) dx(p, a) \\ &= [\ln(x(p, a) + \eta)]_0^1 = \ln\left(\frac{1 + \eta}{\eta}\right) = \ln(k + 1) \end{aligned} \quad \triangleleft$$

**Proof of Theorem 4.** The proof follows directly from the two claims above. Let  $(x, \alpha, \beta, \gamma)$  denote the primal and dual variables constructed by Algorithm 2, and  $(x^*, \alpha^*, \beta^*, \gamma^*)$  be the corresponding variables in the optimal solutions. Using LP duality for the last step, we have:

$$\begin{aligned} \sum_{p \in \mathcal{U}} \sum_{\substack{a \geq 1: \\ t_{p,a} \leq T}} x(p, a) &\leq 2 \left( \sum_t (n - k) \alpha(t) - \sum_{t,i} (n_i - k_i) \beta(t, i) - \sum_{p,a} \gamma(p, a) \right) && (\text{by Claim 7}) \\ &\leq 2 \ln(k + 1) \left( \sum_t (n - k) \alpha^*(t) - \sum_{t,i} (n_i - k_i) \beta^*(t, i) - \sum_{p,a} \gamma^*(p, a) \right) \\ &&& (\text{by Claim 8}) \\ &\leq 2 \ln(k + 1) \left( \sum_{p \in \mathcal{U}} \sum_{\substack{a \geq 1: \\ t_{p,a} \leq T}} x^*(p, a) \right) \end{aligned} \quad \blacktriangleleft$$

## 6 Rounding

We now describe an  $O(1)$ -approximate rounding scheme for the fractional algorithm of Section 5, thus proving Theorem 5.

► **Theorem 5.** *There is an  $O(\ln k)$ -competitive integral algorithm for online caching with reserves.*

**Proof.** For any time  $t = 1, 2, \dots$ , the randomized integral algorithm will maintain a distribution  $\mu^t$  of cache states such that for any page  $p$ , the probability that  $p$  is not in the cache (of the randomized algorithm) at time  $t$  is exactly  $x^t(p, r(p, t))$ , where  $x^t$  denotes the value of LP variable  $x$  at time  $t$ . By the design of our primal-dual algorithm, the  $x$ -variables never decrease, so the cost incurred by the fractional algorithm to serve page  $p_t$  is given by:

$$\text{cost}(t) := \sum_{p \in \mathcal{U}, p \neq p_t} \left( x^{t+1}(p, r(p, t)) - x^t(p, r(p, t)) \right).$$

We will shortly describe how the integral algorithm moves from the distribution  $\mu^t$  to  $\mu^{t+1}$  while ensuring that the expected number of fetches and evictions is at most  $O(\text{cost}(t))$ . We remark that our rounding algorithm does not necessarily run in polynomial time. This is because the support size of  $\mu^t$  can be super-polynomial in  $|\mathcal{U}|$  and  $k$ . This is not an issue for competitive analysis of online algorithms, so we simply assume that we are maintaining a probability distribution over  $\binom{|\mathcal{U}|}{k}$  cache states.

Fix some time  $t$ . For each page  $p \in \mathcal{U} \setminus \{p_t\}$ , define  $y(p) := 1 - x^t(p, r(p, t))$  and  $y'(p) := 1 - x^{t+1}(p, r(p, t))$  to be the portion of page  $p$  that is in the cache at the start of times  $t$  and  $t + 1$ , respectively. Also define  $y(p_t) = 1 - x^t(p_t, r(p_t, t))$  and  $y'(p_t) := 1$ ; note that the fractional algorithm pays cost  $1 - y(p_t)$  to fully fetch  $p_t$  into the cache by the end of time step  $t$ . With the above notation, for any page  $p \in \mathcal{U}$ , we have  $\Pr_{C \sim \mu^t}[p \in C] = y(p)$  and  $\Pr_{C \sim \mu^{t+1}}[p \in C] = y'(p)$ .

To simplify the description of our rounding scheme, we further assume that the changes that occur in the primal solution between states  $x^t$  and  $x^{t+1}$  do so through a sequence of smaller changes where the  $x$ -value changes for exactly two pages (and hence the  $y$ -value also changes for exactly two pages). Let  $p, q \in \mathcal{U}$  and  $\epsilon \in [0, 1]$  be such that  $y'(p) = y(p) + \epsilon$ ,  $y'(q) = y(q) - \epsilon$ , and  $y'(p') = y(p')$  for all  $p' \in \mathcal{U} \setminus \{p, q\}$ .<sup>¶</sup> Let  $\mu, \mu'$  denote distributions over integral cache states that agree with  $y$  and  $y'$ , respectively. The cost incurred by the fractional algorithm to move from  $y$  to  $y'$  is exactly  $\epsilon$  (because it only pays for evictions). We now describe how the integral algorithm moves from  $\mu$  to  $\mu'$  by incurring a cost of at most  $6\epsilon$ . To modify a  $\delta$  probability measure of the cache-state from  $C$  to  $C'$ , the integral algorithm pays a cost of  $\delta \cdot |C \setminus C'|$ . We divide the modification steps into three phases:

1. **Fixing the marginals:** In this phase, we modify the distribution  $\mu$  so that for any page  $p' \in \mathcal{U}$ ,  $\Pr_{C \sim \mu}[p' \in C]$  changes from  $y(p')$  to  $y'(p')$ . We accomplish this by: (i) adding  $p$  to an  $\epsilon$  probability measure of cache states from  $\mu$  that do not contain  $p$ ; and (ii) removing  $q$  from an  $\epsilon$  measure of cache states from  $\mu$  that contain  $q$ . The cost incurred in this step is exactly  $\epsilon$ .

By the end of this phase, for any (possibly infeasible) cache state  $C$  in  $\mu$ , we have  $|C| \in \{k - 1, k, k + 1\}$ . Let  $0 \leq \epsilon_1 \leq \epsilon$  denote the probability measure of cache states with exactly  $k - 1$  pages. By the description of the modification step, it is clear that *exactly*  $\epsilon_1$  measure of cache states have cardinality  $k + 1$ . Further, let  $0 \leq \epsilon_2 \leq \epsilon$  denote the measure of cache states that violate the reserve constraint for some agent. Since only removing the page  $q$  could lead to a constraint violation, we must have  $|C| \in \{k - 1, k\}$  for any such violating cache state.

2. **Fixing the size:** In this phase, we match an  $\epsilon_1$  measure of cache-states of size  $k - 1$  with an  $\epsilon_1$  measure of cache-states of size  $k + 1$ . Let  $C$  and  $C'$  denote page-sets of size  $k - 1$  and  $k + 1$ , respectively, that are matched with some positive measure  $\alpha$ . Pick an arbitrary page  $p' \in C' \setminus C$ . We remove  $p'$  from an  $\alpha$  measure of state  $C'$ , and add it to an  $\alpha$  measure of state  $C$ . The total cost incurred in this phase is exactly  $\epsilon_1 \leq \epsilon$ .

By the end of this phase, all cache-states have cardinality exactly  $k$ . However, the removal of the page  $p'$  above may cause violations of the reserve constraint. Let  $\epsilon_3 \in [0, 1]$  denote the measure of cache states that satisfied all reserve constraints at the end of the first phase, but now violate some reserve constraint. By the above discussion, such cache states arise from the removal of page  $p' \in C' \setminus C$  from  $C'$  (that had size  $k + 1$ ), so  $\epsilon_3 \leq \epsilon_1$ . Overall, exactly  $\epsilon_2 + \epsilon_3$  measure of cache states violate some reserve constraint. In fact, every violated cache state violates a single reserve constraint.

<sup>¶</sup> Here,  $p$  plays the role of page  $p_t$  that is fetched into the cache, and  $q$  plays the role of pages in  $\mathcal{U} \setminus \{p_t\}$  that are evicted to make space for  $p_t$ .

- 3. Fixing the violated reserve constraint:** We now fix all violated reserve constraints by matching an  $\epsilon_2 + \epsilon_3$  measure of cache states with exactly an  $\epsilon_2 + \epsilon_3$  measure of cache states that have an excess in that reserve constraint. More precisely, if  $C$  is a cache state that violates the reserve constraint for agent  $i \in \mathcal{I}$ , then we match an  $\alpha > 0$  measure of  $C$  with another cache state  $C'$  that satisfies  $|C' \cap \mathcal{U}(i)| \geq k_i + 1$ . Such a matching exists because the fractional solution  $y'$  satisfies all reserve constraints and (by the end of the first phase we ensured that) the distribution  $\mu$  satisfies the reserve constraint in expectation: for every cache state  $C$  with  $|C \cap \mathcal{U}(i)| < k_i$ , there must exist another cache state  $C'$  with  $|C' \cap \mathcal{U}(i)| > k_i$ . We move an arbitrary page  $p' \in \mathcal{U}(i) \cap (C' \setminus C)$  from  $C'$  to  $C$ . In exchange for  $p'$ , we move an arbitrary page  $q' \in (\mathcal{U} \setminus \mathcal{U}(i)) \cap (C \setminus C')$  from  $C$  to  $C'$  that does not violate any reserve constraints for the state  $C$ . The choice of  $q'$  is well-defined because the size of  $C$  is  $k + 1$  right after  $p'$  is moved from  $C'$  to  $C$ , and we also have  $|C \cap \mathcal{U}(i)| = k_i$ , so there must exist some other agent  $j \neq i$  satisfying  $|C \cap \mathcal{U}(j)| > k_j$ . The cost incurred in this phase is at most  $2(\epsilon_2 + \epsilon_3) \leq 4\epsilon$ . At the end of this step, all cache states have size exactly  $k$  and satisfy all reserve constraints. The marginal probabilities in the resulting distribution  $\mu'$  matches  $y'$ .



This completes the description of our rounding scheme. In the first step, we incurred a total cost of exactly  $\epsilon$  while in the second and third steps, we incurred a total cost of at most  $\epsilon_1 \leq \epsilon$  and  $2(\epsilon_2 + \epsilon_3) \leq 4\epsilon$ . Since the fractional algorithm incurs a cost of  $\epsilon$ , the theorem follows.  $\blacktriangleleft$

## References

- 1 Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *TCS*, 234(1-2):203–218, 2000.
- 2 Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An  $O(\log k)$ -competitive algorithm for generalized caching. In *SODA*, pages 1681–1689, 2012.
- 3 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Towards the randomized  $k$ -server conjecture: A primal-dual approach. In *SODA*, pages 40–55, 2010.
- 4 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):1–24, 2012.
- 5 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SICOMP*, 41(2):391–414, 2012.
- 6 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-augmented weighted paging. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–89. SIAM, 2022.
- 7 Sorav Bansal and Dharmendra S. Modha. CAR: Clock with adaptive replacement. In *3rd USENIX Conference on File and Storage Technologies (FAST 04)*, San Francisco, CA, March 2004. USENIX Association. URL: <https://www.usenix.org/conference/fast-04/car-clock-adaptive-replacement>.
- 8 L. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.
- 9 Leah Epstein, Csanád Imreh, Asaf Levin, and Judit Nagy-György. Online file caching with rejection penalties. *Algorithmica*, 71(2):279–306, 2015.
- 10 Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.
- 11 Sharat Ibrahimpur, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua Wang. Caching with reserves, 2022. [arXiv:2207.05975](https://arxiv.org/abs/2207.05975).

- 12   Wu Kan, Tu Kaiwei, Patel Yuvraj, Sen Rathijit, Park Kwanghyun, Arpaci-Dusseau Andrea, and Remzi Arpaci-Dusseau. NyxCache: Flexible and efficient multi-tenant persistent memory caching. In *20th USENIX Conference on File and Storage Technologies (FAST 22)*, pages 1–16, Santa Clara, CA, February 2022. USENIX Association. URL: <https://www.usenix.org/conference/fast22/presentation/wu>.
- 13   Mayuresh Kunjir, Brandon Fain, Kamesh Munagala, and Shivnath Babu. Robus: fair cache allocation for data-parallel workloads. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 219–234, 2017.
- 14   Lyle A McGeoch and Daniel D Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(1-6):816–825, 1991.
- 15   Nimrod Megiddo and Dharmendra S Modha. {ARC}: A {Self-Tuning}, low overhead replacement cache. In *2nd USENIX Conference on File and Storage Technologies (FAST 03)*, 2003.
- 16   Qifan Pu, Haoyuan Li, Matei Zaharia, Ali Ghodsi, and Ion Stoica. {FairRide}:{Near-Optimal}, fair cache sharing. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 393–406, 2016.
- 17   Yinghao Yu, Wei Wang, Jun Zhang, and Khaled Ben Letaief. Lacs: Load-aware cache sharing with isolation guarantee. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 207–217. IEEE, 2019.

# Space Optimal Vertex Cover in Dynamic Streams

Kheeran K. Naidu  

Department of Computer Science, University of Bristol, UK

Vihan Shah  

Department of Computer Science, Rutgers University, Piscataway, NJ, USA

---

## Abstract

We optimally resolve the space complexity for the problem of finding an  $\alpha$ -approximate minimum vertex cover ( $\alpha$ MVC) in dynamic graph streams. We give a randomised algorithm for  $\alpha$ MVC which uses  $O(n^2/\alpha^2)$  bits of space matching Dark and Konrad’s lower bound [CCC 2020] up to constant factors. By computing a random greedy matching, we identify “easy” instances of the problem which can trivially be solved by returning the entire vertex set. The remaining “hard” instances, then have sparse induced subgraphs which we exploit to get our space savings and solve  $\alpha$ MVC.

Achieving this type of optimality result is crucial for providing a complete understanding of a problem, and it has been gaining interest within the dynamic graph streaming community. For connectivity, Nelson and Yu [SODA 2019] improved the lower bound showing that  $\Omega(n \log^3 n)$  bits of space is necessary while Ahn, Guha, and McGregor [SODA 2012] have shown that  $O(n \log^3 n)$  bits is sufficient. For finding an  $\alpha$ -approximate maximum matching, the upper bound was improved by Assadi and Shah [ITCS 2022] showing that  $O(n^2/\alpha^3)$  bits is sufficient while Dark and Konrad [CCC 2020] have shown that  $\Omega(n^2/\alpha^3)$  bits is necessary. The space complexity, however, remains unresolved for many other dynamic graph streaming problems where further improvements can still be made.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Graph Streaming Algorithms, Vertex Cover, Dynamic Streams, Approximation Algorithm

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.53

**Category** APPROX

**Funding** Kheeran K. Naidu: EPSRC Doctoral Training Studentship EP/T517872/1.

Vihan Shah: Research supported in part by a NSF CAREER Grant CCF-2047061.

**Acknowledgements** We are grateful to Sepehr Assadi and Christian Konrad for many helpful discussions. We also appreciate the valuable comments from our APPROX 2022 reviewers.

## 1 Introduction

*Graph streaming* is a setting in which a graph is specified by a sequence of edges, typically in arbitrary order. It is particularly useful for processing massive graphs where having random access to the edges of the graph is either impossible or computationally infeasible.

Research in this area began with *insertion-only streams*, where the stream is made up of a sequence of edge insertions only. In their seminal work, Feigenbaum, Kannan, McGregor, Suri, and Zhang [19] showed that for many problems including minimum spanning tree, connectivity, and bipartiteness,  $\Omega(n)$  bits of space is necessary and  $O(n \log n)$  bits is sufficient for any  $n$ -vertex graph. This logarithmic gap was often overlooked and deemed not important

---

A full version of the paper with the same title appears on arXiv.



© Kheeran K. Naidu and Vihan Shah;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 53; pp. 53:1–53:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

when proving optimality for graph problems, but it left unresolved the question of whether the logarithmic factor was required for simply storing edges or if other techniques could remove it. About a decade later, Sun and Woodruff [39] showed that the logarithmic factor was indeed necessary by improving the lower bounds to  $\Omega(n \log n)$  bits, asymptotically matching the upper bounds up to constant factors.

*Dynamic graph streams*, which allow for sequences of both edge insertions and deletions, prove to be more difficult. Edges that arrive in the stream are not necessarily in the final graph as they may later be deleted. In fact, it is well-known in the community that it is impossible to deterministically return a single edge of a dense graph without storing all of its edges. As a result, almost all dynamic graph streaming algorithms rely on counters which use  $O(\log n)$  bits of space or they rely on  $L_0$ -sampling which optimally uses  $\Theta(\log^3 n)$  bits of space<sup>1</sup> [22, 25]. In essence, counters are used to solve the problem of determining whether an edge is present in an edge induced subgraph [18] (see also [15]), whereas  $L_0$ -sampling also returns the identity of a uniform random edge if one is present [1, 2, 27, 15, 9, 7, 30, 26, 11]. A notable exception includes spectral sparsification [23, 24] which relies on  $L_2$ -heavy-hitters (non-uniform sampling).

Resolving the space complexity up to constant factors for dynamic graph streaming problems has continued to be an elusive task. Ahn, Guha, and McGregor [1] gave an algorithm for connectivity using  $O(n \log^3 n)$  bits of space, and for several years, the best known lower bound was the insertion-only bound of  $\Omega(n \log n)$  bits [39]. However, in 2019, Nelson and Yu [36] improved the lower bound to  $\Omega(n \log^3 n)$  bits in the dynamic graph streaming setting. To the best of our knowledge, this is the only problem in this setting which has space bounds that prove the necessity of the  $\Theta(\log^3 n)$  overhead of randomly sampling an edge (using  $L_0$ -sampling). The approximate minimum cut problem which has a  $\Omega(n \log^3 n)$  bit lower bound [36] (and a  $O(n \log^4 n)$  bit upper bound [1]) similarly shows that logarithmic factors are necessary. A perhaps more surprising result was the recent progress on  $\alpha$ -approximate maximum matching ( $\alpha$ MM). The lower bound of  $\Omega(n^2/\alpha^3)$  bits [18] (see also [9]) and the previous upper bound of  $O(n^2/\alpha^3 \cdot \log^4 n)$  bits [9, 15] seem to indicate that the logarithmic overhead of sampling an edge is required. However, Assadi and Shah [11] improved the upper bound to  $O(n^2/\alpha^3)$  bits showing that this is not the case. On the other hand, for problems such as vertex cover [18], dominating set [26], and spectral sparsification [24], their space bounds have a gap of logarithmic factors, and therefore further improvements can still be made.

**Our Results.** In this work, we optimally resolve the space complexity up to constant factors for the problem of finding an  $\alpha$ -approximate minimum vertex cover ( $\alpha$ MVC) in a dynamic graph stream. In particular, we improve the upper bound to  $O(n^2/\alpha^2)$  bits, matching the  $\Omega(n^2/\alpha^2)$  bits lower bound [18] and showing that the logarithmic overhead is not required. Our main result is the following:

► **Theorem 1.** *There exists a randomised dynamic graph streaming algorithm for  $\alpha$ MVC that succeeds with high probability and uses  $O(n^2/\alpha^2)$  bits of space for any  $\alpha \leq n^{1-\delta}$  where  $\delta > 0$ .*

**Previous Work.** It has been shown by Dark and Konrad [18] that  $\Omega(n^2/\alpha^2)$  bits is necessary for  $\alpha$ MVC. They also gave a simple deterministic algorithm which uses  $O(n^2/\alpha^2 \cdot \log \alpha)$  bits of space, matching the lower bound up to logarithmic factors. Their algorithm arbitrarily

---

<sup>1</sup> This optimal space bound applies when the probability of success is at least  $1 - \frac{1}{\text{poly}(n)}$ .

partitions the vertex set into  $n/\alpha$  groups of size  $\alpha$  and uses counters, which introduce the logarithmic overhead, to maintain the number of edges between each of the  $\Theta(n^2/\alpha^2)$  pairs of vertex groups. The solution follows by computing a group-level minimum vertex cover, and then returning the vertices of the covering groups.

**Main Techniques.** We improve the approach of Dark and Konrad [18] by additionally computing a supporting random GREEDY matching and randomly partitioning the vertex set into  $n/\alpha$  groups, effectively using randomisation to reduce the space required. The random GREEDY matching returned is either large enough to imply a trivial solution for  $\alpha$ MVC (“easy” case) or implies sparseness properties of the residual subgraph induced by the unmatched vertices (“hard” case). To solve the “hard” cases, we use the sparseness properties and the random partitioning to argue that there are only  $O(1)$  many edges between each pair of vertex groups in the residual subgraph. Therefore, storing edge counters for each of the  $\Theta(n^2/\alpha^2)$  many pairs, as done by Dark and Konrad [18], now requires only  $O(n^2/\alpha^2)$  bits of space in total.

**Sampling Strategies.** The sparseness properties (of the residual subgraph) implied are reliant on the method of randomly sampling edges from the graph. Uniformly sampling from the edge set only implies sparseness properties sufficient for a small range of  $\alpha$  since it is skewed to sampling high degree vertices. On the other hand, non-uniform sampling – sampling from the neighbourhood of a random set of vertices, coined *neighbourhood edge sampling* by Assadi and Shah [11] – is less biased towards high degree vertices and implies the necessary sparseness properties for the full range of  $\alpha$ . Indeed, Assadi and Shah [11] also use the approach of computing a GREEDY matching on non-uniformly sampled edges to identify the “easy” and “hard” instances of  $\alpha$ MM. However, for  $\alpha$ MVC, our “easy” and “hard” instances differ from those of  $\alpha$ MM, so we require different guarantees. Furthermore, we use different techniques for solving the “hard” instances.

**Further Related Work.** Resolving the space complexity up to constant factors has also been achieved for non-graph problems in the general data streaming setting. For instance, Braverman, Katzman, Seidell, and Vorsanger [14] gave an upper bound for finding a constant factor approximation to the  $k$ -th frequency moment in constantly many passes that matches the lower bound of Woodruff and Zhang [40]. Price and Woodruff [37] showed a lower bound for any adaptive sparse recovery scheme that matches the upper bound of Indyk, Price, and Woodruff [21]. Graph problems in other streaming settings have also been studied. For example, the settings which allow multiple passes over the stream [31, 28, 6, 10, 32, 4, 8], have a random arrival order [31, 5, 12], or have highly structured deletions via a sliding window [16, 17, 13] have been considered. See the work by McGregor [33] for an excellent survey on graph streaming algorithms.

**Outline.** We begin in Section 2 with some important notation and tools which we will later use. In Section 3, we discuss the guarantees required from a random GREEDY matching for  $\alpha$ MVC. In Section 4, we present and analyse our algorithm that proves Theorem 1. Then, we conclude in Section 5.



## 2 Preliminaries

For any  $n$ -vertex graph  $G = (V, E)$ , let  $\mu(G)$  be the size of the maximum matching of the graph, let  $V^*(G)$  be a minimum vertex cover, and let  $\text{opt}(G)$  be its size. We will simply use  $\mu$ ,  $V^*$  or  $\text{opt}$  if the graph is clear from context. For any subset of edges  $F \subseteq E$ , we denote the set of their endpoints by  $V(F)$ . For any subgraph  $H$  of  $G$  and vertex  $v \in V$ , we use  $N_H(v)$  to denote the neighbourhood of  $v$  in  $H$ .

The graph  $G$  may be specified as a dynamic graph stream<sup>2</sup>  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_N)$  such that  $\sigma_j = (i_j, \Delta_j)$  where  $i_j \in [m]$  for  $m = \binom{n}{2}$  and  $\Delta_j \in \{1, -1\}$  (insertions or deletions). Note that edges may only be deleted if they have previously been inserted. Additionally, the stream must produce a vector  $\text{vec}(E) \in \{0, 1\}^m$  that defines the edge set  $E$ , i.e., the  $i^{\text{th}}$  entry of the vector indicates the presence of the edge indexed by  $i \in [m]$ .

In our work, we will rely on limited independence hash functions to reduce the space complexity of our algorithm. Roughly speaking, a hash function sampled from a family of  $k$ -wise independent hash functions behaves like a totally random function when considering at most  $k$  elements. For simplicity, when we mention a  $k$ -wise independent hash function, we will mean a hash function sampled from a family of  $k$ -wise independent hash functions. We use the following standard result for  $k$ -wise independent hash functions.

► **Proposition 2** ([34]). *For all integers  $n, m, k \geq 2$ , there is a family of  $k$ -wise independent hash functions  $\mathcal{H} = \{h : [n] \rightarrow [m]\}$  such that sampling and storing a function  $h \in \mathcal{H}$  takes  $O(k \cdot (\log n + \log m))$  bits of space.*

We shall also use the following concentration result on an extension of Chernoff-Hoeffding bounds for  $k$ -wise independent hash functions.

► **Proposition 3** ([38]). *Suppose  $h$  is a  $k$ -wise independent hash function and  $X_1, \dots, X_m$  are  $m$  random variables in  $\{0, 1\}$  where  $X_i = 1$  iff  $h(i) = 1$ . Let  $X := \sum_{i=1}^m X_i$ . Then, for any  $\varepsilon > 0$ ,*

$$\Pr(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]) \leq \exp\left(-\min\left\{\frac{k}{2}, \frac{\varepsilon^2}{4 + 2\varepsilon} \cdot \mathbb{E}[X]\right\}\right).$$

Finally, we will use the following sketching tool for dynamic graph streams to test the size of the neighbourhood of a subset of vertices.

► **Proposition 4** ([11]). *Let  $a \geq b \geq 2$  be known integers. Consider a  $n$ -vertex graph  $G = (V, E)$  specified in a dynamic stream and let  $S \subseteq V$  be a known set. Then, given a set  $T \subseteq V$  of size at most  $a$  at the end of the stream, there exists a randomised algorithm that returns “Yes” if  $|N_G(S) \setminus T| \geq b$  or “No” if  $|N_G(S) \setminus T| \leq \frac{1}{2} \cdot b$ , uses  $O(\frac{a}{b} \cdot \log^3 n)$  bits of space, and succeeds with probability at least  $1 - n^{-3}$ . We denote one such algorithm as  $\mathcal{ALG}_{NT}(S; a, b)$ .*

## 3 Sampling Strategies for Random Greedy Matchings

In this section, we discuss and present the tool that we use to either find a large matching or show that the residual subgraph induced by the unmatched vertices is sparse.

<sup>2</sup> A dynamic graph stream is a special case of the strict turnstile data streaming model [35] where we consider only bit-vectors which represent the edges of a graph.

This approach was also used in Assadi and Shah’s recent work for  $\alpha$ MM [11] to identify “easy” and “hard” instances of the problem. For  $\alpha$ MVC, the “easy” case is finding a large enough matching to imply that we can trivially return the entire vertex set to solve the problem. The “hard” case is when we get a sparse residual subgraph, which is where our main savings in space come from. Identifying these cases can be accomplished by computing a GREEDY matching on randomly sampled edges of the graph.

Uniformly sampling as many edges as possible from a  $n$ -vertex graph (using  $L_0$ -sampling) without exceeding  $O(n^2/\alpha^2)$  bits of space followed by computing a GREEDY matching implies sparseness properties based on an already known maximum degree bound of the residual subgraph induced by the unmatched vertices [3, 29, 20]. Intuitively, uniform sampling is skewed towards sampling edges incident to high degree vertices. Hence, a GREEDY matching either matches these high degree vertices or matches many of its neighbours (decreasing their residual degree), and regardless of the size of the matching found, this gives a  $\text{poly}(\alpha)$  max degree bound in the residual graph. Furthermore, we can show that this also bounds the average degree (even when a small matching is found) since a worst-case instance<sup>3</sup> practically has all vertices in the residual subgraph with max degree. This degree bound, however, is only sufficient for solving  $\alpha$ MVC for any  $\alpha \ll n^{\frac{1}{3.5}}$ .

Non-uniformly sampling the edges using neighbourhood edge sampling followed by GREEDY, as done by Assadi and Shah [11], proves to give better sparseness properties, and thus a better average degree bound<sup>4</sup>. The benefit of neighbourhood edge sampling is that it biases away from sampling high degree vertices. Furthermore, when a small GREEDY matching is found, the implication is that the residual subgraph is sparse. Therefore, the average degree bound is sufficient for solving the “hard” case of  $\alpha$ MVC for the full range of  $\alpha$ .

As previously mentioned, Assadi and Shah’s algorithm called Match-or-Sparsify [11], does exactly this, although its guarantees are not sufficient for our purposes. Hence, we first discuss their algorithm, and then explain the alterations we make.

**Match-or-Sparsify.** For some parameter  $\beta \leq n$ , Assadi and Shah’s Match-or-Sparsify $_{\beta}$  algorithm non-uniformly samples edges using space  $O(\beta^2/\alpha^3)$  bits, and then computes a GREEDY matching from them. They give an intricate analysis to show that their algorithm either finds a large matching of size at least  $\beta/8\alpha$  or implies that the residual subgraph has at most  $20 \cdot \beta \cdot \log^4 n$  edges [11, Lemma 16]. Unlike uniform sampling, the residual properties (sufficiently) only hold when the matching is small – a key property exploited in their analysis. Additionally, in order for the guarantees to hold, they rely on the assumption that  $\beta \geq \alpha^2 \cdot n^{\delta}$ . Informally, when  $\beta$  is set as the size of the maximum matching  $\mu$ , Match-or-Sparsify $_{\mu}$  finds a large matching in “easy” graph cases and a sparse residual subgraph in the “hard” graph cases. However,  $\mu$  is not known, so they find a setting of  $\beta$  close to  $\mu$  by running Match-or-Sparsify $_{\beta}$  in parallel with  $\beta$  as all powers of 2 between 1 and  $n$ .

**Our Alterations.** The first thing to note is that the “easy” and “hard” instances for  $\alpha$ MM and  $\alpha$ MVC are not the same. Consider Match-or-Sparsify $_{\beta}$  when a large GREEDY matching is found. Since at least one endpoint of each matching edge must be in a vertex cover, it implies that  $\text{opt} \geq \frac{\beta}{8\alpha}$ . However, returning a solution to  $\alpha$ MVC at this stage can only be

<sup>3</sup> Consider a graph with a large clique on  $\Theta(n/\alpha)$  vertices where most the edges are sampled from, and many smaller cliques which assert the guaranteed max degree bounds.

<sup>4</sup> Having an average degree bound is more difficult to work with, but in this case, the bound on the average degree is much smaller than the bound on the max degree in the uniform case.

of size at most  $\Theta(\beta)$ , which would not be a trivial solution (the entire vertex set) with  $\beta \ll n$ . Furthermore, we have no guaranteed sparseness properties since the matching found is large. Hence, instead of needing  $\beta \approx \mu$ , which requires  $\log n$  many runs to find, we only need a single run of  $\text{Match-or-Sparsify}_n$  (with the parameter  $\beta$  fixed as  $n$ ). Secondly, their assumption that  $\beta \geq \alpha^2 \cdot n^\delta$  implies that  $\alpha \leq n^{\frac{1-\delta}{2}}$ , but we require it to hold for any  $\alpha \leq n^{1-\delta}$ . Since we have an additional  $\alpha$  factor of space (see [18]), we can increase the number of non-uniformly sampled edges to use  $O(n^2/\alpha^2)$  bits instead, which allows us to remove the assumption. Finally, the increase in the number of samples also allows us to increase the sparseness guarantees of the residual subgraph by an  $\alpha$  factor. Therefore, this altered  $\text{Match-or-Sparsify}_n$  algorithm, denoted by  $\mathcal{ALG}_{MS}$ , gives us the following lemma (the full proof is given in the arXiv version for completeness).

► **Lemma 5.** *There is a linear sketch for dynamic graph streams that, given any graph  $G = (V, E)$  specified via  $\text{vec}(E)$ , uses  $O(n^2/\alpha^2)$  bits of space and with high probability outputs a matching  $M_{\text{easy}}$  that satisfies at least one of the following conditions for any  $\alpha \leq n^{1-\delta}$  and  $\delta > 0$ :*

- **Match-case:** *The matching  $M_{\text{easy}}$  has at least  $\frac{n}{8\alpha}$  edges;*
- **Sparsify-case:** *The induced subgraph of  $G$  on vertices not matched by  $M_{\text{easy}}$ , denoted by  $G_R$ , has at most  $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$  edges.*

## 4 Main Result

In this section, we give a dynamic graph streaming algorithm for  $\alpha\text{MVC}$  for any  $n$ -vertex graph which implies our main result:

► **Theorem 1.** *There exists a randomised dynamic graph streaming algorithm for  $\alpha\text{MVC}$  that succeeds with high probability and uses  $O(n^2/\alpha^2)$  bits of space for any  $\alpha \leq n^{1-\delta}$  where  $\delta > 0$ .*

Before proceeding, we give the following standard assumption (with reason) which simplifies what we need to prove.

► **Assumption 6.** *A randomised dynamic graph streaming  $\Theta(\alpha)$ -approximation algorithm that uses  $O(n^2/\alpha^2)$  bits of space and succeeds on graphs where  $\text{opt} \geq \frac{n}{\alpha \cdot \log^2 n}$  is sufficient to prove Theorem 1.*

**Reason.** Let  $\mathcal{A}$  be an algorithm that returns a  $(c \cdot \alpha)$ -approximation using  $O(n^2/\alpha^2)$  bits of space. Run  $\mathcal{A}$  with parameter  $\alpha/c$  to get an  $\alpha$ -approximation which similarly uses  $O(n^2/\alpha^2)$  bits.

Then, since we can run  $\Theta(1)$  many algorithms which use  $O(n^2/\alpha^2)$  bits of space in parallel without asymptotically increasing the space, we run an additional algorithm which detects and outputs a solution for graphs with small  $\text{opt}$ .

**Algorithm for small  $\text{opt}$ .** We use the well-known algorithm for finding an exact minimum vertex cover in dynamic graph streams given the promise that  $\text{opt} \leq k$  with  $k = \frac{n}{\alpha \cdot \log^2 n}$  [15]. If  $\text{opt} < k$ , then we get an optimal solution; otherwise, we get a set of vertices of size  $k$  which are not necessarily a solution. Thus, we can detect this case by the size of the returned vertex cover being smaller than  $k$ . The space taken by the algorithm is  $O(k^2 \cdot \log^4 n) = O(n^2/\alpha^2)$  bits and it works for all  $\alpha = \omega(1)$  (for  $\alpha = \Theta(1)$  we can store the entire graph). ◀

■ **Algorithm 1** Optimal Dynamic Vertex Cover.

**Input:** A dynamic graph stream  $\sigma$  for a  $n$ -vertex graph  $G = (V, E)$ , a small constant  $\delta > 0$ , and a positive integer  $\alpha \leq n^{1-\delta}$

**Output:** A vertex cover  $V_C$  of  $G$

**Pre-processing:**

- 1: Initialise  $\mathbf{M}$  to be an instance of  $\mathcal{ALG}_{MS}$  (Lemma 5)
- 2: Randomly partition  $V$  into groups  $V_1, V_2, \dots, V_{\frac{n}{\alpha}}$  having size in  $[\alpha/2, 2\alpha]$
- 3: For each group  $V_i$ , initialise  $\mathbf{N}_i$  to be an instance of  $\mathcal{ALG}_{NT}(V_i; a, b)$  (Proposition 4) with  $a = n/\alpha$  and  $b = n^{\delta/2}$
- 4: Set  $c = 15/\delta$

**Processing the stream:**

- 5: Update  $\mathbf{M}$  and each  $\mathbf{N}_i$  using  $\sigma$
- 6: For every pair of groups  $V_i$  and  $V_j$ , store a counter  $C_{i,j}$  for the number of edges between them modulo  $c$
- 7: For every group  $V_i$ , store a counter  $C_i$  for the number of internal edges

**Post-processing:**

- 8: Let  $M_{\text{easy}}$  be the matching returned by  $\mathbf{M}$
- 9: **if**  $M_{\text{easy}}$  has at least  $\frac{n}{8 \cdot \alpha}$  edges **then return**  $V$
- 10: Let  $V_C$  be the union of all groups  $V_i$  containing a vertex of  $M_{\text{easy}}$  or with  $C_i > 0$
- 11: Add to  $V_C$  all remaining vertex groups  $V_i$  where  $\mathbf{N}_i$  returns “Yes” when  $T = V(M_{\text{easy}})$
- 12: Consider the multi-graph  $G'$  obtained by contracting the vertices of each remaining vertex group  $V_i$  into a single vertex  $v_i$  where  $C_{i,j}$  represents the multiplicity modulo  $c$  of each edge  $(v_i, v_j)$  in  $G'$
- 13: Greedily compute a vertex cover  $V'_C$  of  $G'$
- 14: For all  $v_i \in V'_C$ , add vertex group  $V_i$  to  $V_C$
- 15: **return**  $V_C$

**Algorithm Description.** Let  $G = (V, E)$  be specified by a dynamic graph stream,  $\delta > 0$ , and  $\alpha \leq n^{1-\delta}$  be the inputs to Algorithm 1. The algorithm, in its pre-processing step, partitions  $V$  into  $n/\alpha$  groups using a  $(10 \cdot \log n)$ -wise independent hash function (when the space allows, i.e., for small  $\alpha$ , we do this using a uniform random permutation instead), and we later show that all their sizes lie between  $\alpha/2$  and  $2\alpha$  with high probability. During the stream, it maintains counters modulo some constant for the number of edges between each pair of groups and (standard) counters for the number of internal edges of each group. In parallel, it computes a random matching  $M_{\text{easy}}$  using an instance of  $\mathcal{ALG}_{MS}$  (Lemma 5) and maintains residual neighbourhood size testers for each vertex group using instances of  $\mathcal{ALG}_{NT}$  (Proposition 4). In the post-processing step, if the matching is of size at least  $\frac{n}{8 \cdot \alpha}$ , then the entire vertex set is returned. Otherwise, the vertex groups containing any vertex of the matching or any internal edges are entirely picked in the solution – we call these *simple vertex groups*. Next, the remaining vertex groups  $V_i$  whose residual neighbourhood is large,  $|N_{G_R}(V_i)| = |N_G(V_i) \setminus V(M_{\text{easy}})| \geq n^{\delta/2}$  where  $G_R = G[V \setminus V(M_{\text{easy}})]$ , are added to the solution – we call these *residual vertex groups*. Finally, among the leftover *clean vertex groups*, the algorithm uses the counters modulo some constant to perform a group-level vertex cover, and then further adds the covering groups to the solution before returning it.

► **Definition 7** (Simple Vertex Groups). *We say that a vertex group  $V_i$  is simple if any of its vertices are matched by  $M_{\text{easy}}$  or it has at least one internal edge, i.e.,  $|V_i \cap V(M_{\text{easy}})| > 0$  or  $C_i > 0$ .*

► **Definition 8** (Residual Vertex Groups). *We say that a vertex group  $V_i$  is residual if it is not simple and has a large residual neighbourhood, i.e.,  $|N_{G_R}(V_i)| \geq n^{\delta/2}$ .*

► **Definition 9** (Clean Vertex Groups). *We say that a vertex group  $V_i$  is clean if it is not simple or residual, i.e.,  $|V_i \cap V(M_{\text{easy}})| = 0$ ,  $C_i = 0$  and  $|N_{G_R}(V_i)| < n^{\delta/2}$ .*

Note that throughout the subsequent analysis of Algorithm 1, all results succeed with high probability. Hence, at any point, we can do a simple union bound to show that they all hold with high probability. As such, we condition on this event here to avoid explicitly doing so during the analysis.

Let  $G$  be the input graph of the algorithm. We begin the analysis with the following observation: If  $G$  contains a matching of size at least  $\frac{n}{8\alpha}$ , then  $V$  is a valid  $(8\alpha)$ -approximation of a minimum vertex cover  $V^*$  since at least one endpoint of a matching edge must be in a valid vertex cover. Therefore, if the condition of Algorithm 1 is satisfied, the algorithm terminates and the solution is a valid  $\Theta(\alpha)$ -approximation (“easy” graph instances). Otherwise, the algorithm progresses with  $|M_{\text{easy}}| < \frac{n}{8\alpha}$ , i.e., the sparsify-case of Lemma 5 (“hard” graph instances). This implies that the residual subgraph  $G_R$  is sparse with at most  $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$  many edges. As such, we need to prove that we also get a  $\Theta(\alpha)$ -approximation in the sparsify-case.

We highlight here that the algorithm adds vertex groups to the solution for various reasons, which are determined by whether it is a simple, residual, or clean vertex group (see Definitions 7–9). Hence, we proceed with the analysis of the sparsify-case by considering these different types of vertex groups separately.

**Simple Vertex Groups.** Let  $\mathcal{I}_s$  be the index set of the simple vertex groups. We argue that there are not too many of these, so we can add all of them to the solution.

► **Claim 10.** The number of simple vertex groups  $|\mathcal{I}_s|$  is at most  $2 \cdot \text{opt}(G)$ .

*Proof.* Each edge of the matching  $M_{\text{easy}}$  can cause up to two vertex groups to be classified as simple; however, they must have at least one vertex of  $V^*$  since at least one endpoint of every matching edge must be in  $V^*$ . Therefore, for every two groups classified as simple in this way, there is at least one vertex of  $V^*$  in their union. On the other hand, a group could also be classified as simple if it contains an internal edge, where one of its endpoints must be in  $V^*$ . Hence, for each group classified as simple in this way, there is at least one vertex of  $V^*$  in it. Then, it follows that the number of simple vertex groups must be at most  $2 \cdot |V^*| = 2 \cdot \text{opt}$ .  $\triangleleft$

**Residual Vertex Groups.** Let  $\mathcal{I}_r$  be the index set of the residual vertex groups. Recall that any residual vertex group must have at least  $n^{\delta/2}$  many residual neighbours. We note, however, that due to the guarantees of the neighbourhood size tester algorithm  $\mathcal{ALG}_{NT}$  (see Proposition 4), there are some misclassifications, so some residual vertex groups are also of size between  $\frac{1}{2} \cdot n^{\delta/2}$  and  $n^{\delta/2}$ . This will not be an issue, and moving forward, when we mention residual vertex groups, we assume that this includes the misclassifications. Now, we argue that there are not too many residual vertex groups, so we can add them all to the solution.

► **Claim 11.** The number of residual vertex groups  $|\mathcal{I}_r|$  is at most  $\text{opt}(G)$  with high probability.

Proof. We have that  $|V(M_{\text{easy}})|$  is at most  $\frac{n}{4\alpha}$  and  $G_R$  has at most  $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$  many edges. As such,  $G_R$  has  $n - |V(M_{\text{easy}})| \geq \frac{n}{2}$  vertices, and the average degree of a vertex in  $G_R$  is at most  $20 \cdot \frac{n}{\alpha} \cdot \log^4 n \cdot \frac{2}{n} = \frac{40 \log^4 n}{\alpha}$ . Since each non-simple vertex group  $V_i$  is fully contained in  $G_R$  and has at most  $2\alpha$  vertices, we have that  $\mathbb{E}[|N_{G_R}(V_i)|] \leq \frac{40 \log^4 n}{\alpha} \cdot 2\alpha = 80 \log^4 n$ . Then, it follows by Markov's inequality that

$$\begin{aligned} \Pr(V_i \text{ is residual} \mid V_i \text{ is non-simple}) &\leq \Pr\left(|N_{G_R}(V_i)| \geq \frac{1}{2} \cdot n^{\delta/2}\right) \\ &\leq \frac{2 \cdot 80 \log^4 n}{n^{\delta/2}} \leq \frac{\log^5 n}{n^{\delta/2}}. \end{aligned} \quad (1)$$

Let  $X_i$  be the indicator random variable that a non-simple vertex group  $V_i$  is a residual vertex group, then  $R = \sum_{i \in [\frac{n}{\alpha}] \setminus \mathcal{I}_s} X_i$  is the number of residual vertex groups. By Equation (1), we have the following:

$$\mathbb{E}[R] = \sum_{i \in [\frac{n}{\alpha}] \setminus \mathcal{I}_s} \Pr(X_i) \leq \sum_{i \in [\frac{n}{\alpha}]} \frac{\log^5 n}{n^{\delta/2}} = \frac{n \cdot \log^5 n}{\alpha \cdot n^{\delta/2}}.$$

Finally, since  $\text{opt}(G) \geq \frac{n}{\alpha \cdot \log^2 n}$  (Assumption 6), a further application of Markov's inequality implies the result:

$$\Pr(|\mathcal{I}_r| > \text{opt}) \leq \Pr\left(R > \frac{n}{\alpha \cdot \log^2 n}\right) \leq \frac{n \cdot \log^5 n}{\alpha \cdot n^{\delta/2}} \cdot \frac{\alpha \log^2 n}{n} \leq n^{-\delta/4}.$$

Note that we can easily increase the success probability by running the algorithm in parallel  $40/\delta$  times and detecting failures when the number of residual groups is more than  $n/\alpha \cdot \log^2 n$ . Then, with probability at least  $1 - n^{-10}$ , one of the runs will succeed. This only increases the space of the algorithm by a constant factor since  $40/\delta = \Theta(1)$ .  $\triangleleft$

**Clean Vertex Groups.** Let  $\mathcal{I}_c$  be the index set of the clean vertex groups and let  $\mathcal{I}_c^+$  be the ones added to the solution, which also corresponds to the group-level vertex cover  $V'_C$  in Algorithm 1.

Before analysing the group-level vertex cover, we note that the relevant counters are stored modulo  $c$ . This means that if the number of edges between clean vertex groups is some multiple of  $c$ , the corresponding counter would be 0 and the group-level vertex cover would be incorrect. Hence, we want the number of edges between clean vertex groups to be less than  $c$  with high probability.

$\triangleright$  **Claim 12.** For all pairs of clean vertex groups  $V_i$  and  $V_j$ , with high probability,

$$|N_G(V_i) \cap V_j| < c.$$

Proof. We prove a slightly generalised statement which implies what we need. We show that there are less than  $c$  edges of  $G_R$  between any clean vertex group  $V_i$  and any other vertex group  $V_j$ . This implies what we need since, by definition, all edges between clean vertex groups are in  $G_R$ .

Consider the random partitioning of  $V$  using an at least  $(3 \cdot c)$ -wise independent hash function (the algorithm uses  $(10 \cdot \log n)$ -wise independence). A residual neighbour of the clean vertex group  $v \in N_{G_R}(V_i)$  uniformly belongs to any of the other vertex groups. Since there are  $\frac{n}{\alpha} - 1$  of these (including  $V_j$ , but not including  $V_i$ ), the probability that  $v \in V_j$  is at most  $\frac{2\alpha}{n}$ .

### 53:10 Space Optimal Vertex Cover in Dynamic Streams

Now, since clean vertex groups are non-residual,  $|N_{G_R}(V_i)| \leq n^{\delta/2}$ , and for a fixed  $V_i$  and  $V_j$ , we have that

$$\Pr(|N_{G_R}(V_i) \cap V_j| \geq c) \leq \binom{n^{\delta/2}}{c} \cdot \left(\frac{2\alpha}{n}\right)^c \leq \left(\frac{2\alpha}{n^{1-\delta/2}}\right)^{15/\delta} \leq n^{-7}$$

where we have used  $\alpha \leq n^{1-\delta}$  and  $c = 15/\delta$  in the final inequalities. Then, the result holds with probability at least  $1 - n^{-5}$  by a union bound over all pairs of vertex groups.

Note that for small  $\alpha$  we will partition  $V$  into groups of size exactly  $\alpha$  with a uniform random permutation due to concentration and space reasons (see Claim 15), but the above arguments also hold in this case.  $\triangleleft$

With Claim 12, we can assume that all the counters between clean vertex groups count exactly the number of edges with high probability, that is, the modulo has no effect on the correctness of the algorithm. Thus, the setting is now identical to that of Dark and Konrad's algorithm [18], and we follow a similar argument as they did to analyse the group-level vertex cover and the corresponding subset of clean vertex groups added.

$\triangleright$  **Claim 13.** The number of clean vertex groups added  $|\mathcal{I}_c^+|$  is at most  $2 \cdot \text{opt}(G)$ .

*Proof.* Consider the subgraph  $H = G[\cup_{i \in \mathcal{I}_c} V_i]$  induced by the clean vertex groups. Observe that since  $H$  is an induced subgraph of  $G$ ,  $\text{opt}(H) \leq \text{opt}(G)$ . Then, since the vertex contractions to obtain the multi-graph  $G'$  from  $H$  cannot increase the size of its minimum vertex cover, we have that  $\text{opt}(G') \leq \text{opt}(H)$ . Finally, since we greedily compute the group-level vertex cover  $V'_C$ , it is a 2-approximation and we have that  $|\mathcal{I}_c^+| = |V'_C| \leq 2 \cdot \text{opt}(G') \leq 2 \cdot \text{opt}(G)$ .  $\triangleleft$

By combining the analysis of the simple, residual, and clean vertex groups, we prove the approximation factor of the algorithm.

$\blacktriangleright$  **Lemma 14.** *Algorithm 1 returns a valid  $\Theta(\alpha)$ -approximation of a minimum vertex cover for any input graph  $G$  with  $\text{opt} \geq \frac{n}{\alpha \cdot \log^2 n}$ .*

**Proof.** We first show that the solution  $V_C$  is indeed a valid vertex cover, then we prove that it is a  $\Theta(\alpha)$ -approximation.

**Validity.** For the sake of finding a contradiction, let  $e \in E$  be an edge which is not covered by  $V_C$ . Observe that any non-clean vertex group  $V_i$  is added to  $V_C$ ; thus, all edges with at least one endpoint in any of these vertex groups are covered. So, we have that  $e$  must be in  $G[\cup_{i \in \mathcal{I}_c} V_i]$ , the subgraph induced by the clean vertex groups.

Let  $i, j \in \mathcal{I}_c$  be such that  $e$  has endpoints in the clean vertex groups  $V_i$  and  $V_j$ , implying that there is an edge between their corresponding contracted vertices  $v_i$  and  $v_j$  in the multi-graph  $G'$ . It follows that one of  $v_i$  or  $v_j$  must be in the computed group-level vertex cover  $V'_C$ , so all vertices of either  $V_i$  or  $V_j$ , including at least one endpoint of  $e$ , are added to  $V_C$ . However, this means that  $e$  is covered by  $V_C$ , a contradiction.

**Approximation.** Observe that the solution  $V_C$  is comprised of a (disjoint) union of all simple vertex groups, all residual vertex groups, and a subset of clean vertex groups. Recall that  $\mathcal{I}_s$ ,  $\mathcal{I}_r$  and  $\mathcal{I}_c^+$  are the corresponding index sets of these groups.

By Claims 10, 11, and 13, we have that  $|\mathcal{I}_s| + |\mathcal{I}_r| + |\mathcal{I}_c^+| \leq 5 \cdot \text{opt}$ . Finally, since the size of each vertex group is at most  $2\alpha$ , we can bound the size of the solution as follows:

$$|V_C| = \sum_{i \in \mathcal{I}_s \cup \mathcal{I}_r \cup \mathcal{I}_c^+} |V_i| \leq 2\alpha \cdot (|\mathcal{I}_s| + |\mathcal{I}_r| + |\mathcal{I}_c^+|) \leq 10\alpha \cdot \text{opt}. \quad \blacktriangleleft$$



It remains to show that the algorithm can be implemented using  $O(n^2/\alpha^2)$  bits of space. Algorithm 1 randomly partitions  $V$ , maintains several instances of  $\mathcal{ALG}_{MS}$  (Lemma 5) and  $\mathcal{ALG}_{NT}$  (Proposition 4), and stores various counters. To show the space usage of the algorithm, we first consider each of these components separately.

▷ **Claim 15.** The partitioning of  $V$  into  $\frac{n}{\alpha}$  vertex groups of size in the range  $[\alpha/2, 2\alpha]$  uses  $O(n^2/\alpha^2)$  bits of space and succeeds with high probability.

*Proof.* We show that for small  $\alpha < \log^2 n$ , i.e., when we have sufficient space, we can achieve this with a uniform random permutation, and for large  $\alpha \geq \log^2 n$ , we use a  $(10 \cdot \log n)$ -wise independent hash function.

**Small  $\alpha$ .** For any  $\alpha < \log^2 n$ , we can randomly permute the vertices using  $O(n \log n) = O(n^2/\alpha^2)$  random bits to create a uniform random partitioning of  $V$  into  $\frac{n}{\alpha}$  groups of size  $\alpha$ .

**Large  $\alpha$ .** For any  $\alpha \geq \log^2 n$ , we can partition  $V$  using a  $(10 \cdot \log n)$ -wise independent hash function  $h : [n] \rightarrow [\frac{n}{\alpha}]$  which uses  $O(\log^2 n) = O(n^2/\alpha^2)$  bits by Proposition 2. We bound the size of the groups as follows: Consider any group  $V_j$  ( $j \in [\frac{n}{\alpha}]$ ) and let  $X_i$  be the random variable that is 1 if vertex  $i$  is hashed to  $V_j$ , i.e.,  $h(i) = j$ . Let  $X = \sum_i X_i$  represent the number of vertices in group  $V_j$ . We have  $\mathbb{E}[X] = n \cdot (\alpha/n) = \alpha$ . Using Proposition 3 with  $\varepsilon = 0.1$ ,

$$\Pr(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]) \leq \exp(-5 \log n) \leq n^{-5}.$$

A union bound over all groups implies that with probability at least  $1 - n^{-4}$ , all groups have size between  $0.9\alpha$  and  $1.1\alpha$ . ◁

▷ **Claim 16.** The instances of  $\mathcal{ALG}_{MS}$  and  $\mathcal{ALG}_{NT}$ , and the counters use  $O(n^2/\alpha^2)$  bits of space.

*Proof.* We use one instance of  $\mathcal{ALG}_{MS}$  (Lemma 5) which takes space  $O(n^2/\alpha^2)$  bits. We use  $n/\alpha$  instances of  $\mathcal{ALG}_{NT}$  (Proposition 4) with parameters  $a = n/\alpha$  and  $b = n^{\delta/2}$  each of which take space  $O(\frac{a}{b} \log^3 n) = O((n/\alpha) \cdot (\log^3 n / n^{\delta/2})) = o(n/\alpha)$  bits. This implies that the total space used by  $n/\alpha$  instances is  $O(n^2/\alpha^2)$  bits. We maintain counters modulo a constant  $c = 15/\delta$  for the number of edges between every pair of vertex groups. Each takes  $O(1)$  bits of space, and since there are  $O(n^2/\alpha^2)$  many of these counters, this totals  $O(n^2/\alpha^2)$  bits of space. We also maintain counters for the number of internal edges for each group which requires  $O(\log n) = o(n/\alpha)$  bits of space each. Since there are  $\frac{n}{\alpha}$  many groups, this totals  $O(n^2/\alpha^2)$  bits of space. ◁

Hence, by Claims 15 and 16, we have shown that the components of Algorithm 1 use  $O(n^2/\alpha^2)$  bits in total. We still, however, need to consider the format of the output. When  $\alpha$  gets large enough, the space is only  $o(n)$ , whereas simply storing the output – the vertices of a solution – could require  $\Theta(n)$  bits of space. We solve this by showing that we can implicitly store the solution when there is limited space.

▷ **Claim 17.** The output of Algorithm 1 can be maintained using  $O(n^2/\alpha^2)$  bits of space.

*Proof.* For  $\alpha < \log^2 n$ , we can maintain the vertices of the solution explicitly. For  $\alpha \geq \log^2 n$ , we rely on the hash function  $h$  used to partition  $V$  (see Claim 15). Recall that vertices are added to the solution at a group level, so we can simply maintain a bit vector of length  $\frac{n}{\alpha}$  representing the groups added to the solution. Then, the output consists of  $h$  and the bit vector which is sufficient for checking if a vertex belongs to the solution and uses  $O(\log^2 n + \frac{n}{\alpha}) = O(n^2/\alpha^2)$  bits of space. ◁

We have now shown that Algorithm 1 can be implemented using  $O(n^2/\alpha^2)$  bits of space. Therefore, combined with Lemma 14 and Assumption 6, we have proven our main result, Theorem 1.

## 5 Conclusion

In this paper, we have resolved the space complexity of  $\alpha$ MVC for the full range of  $\alpha$ . We have provided a randomised algorithm which asymptotically matches the lower bound [18] up to constant factors, showing that  $\Theta(n^2/\alpha^2)$  is necessary and sufficient for this problem.

The previous best algorithm for  $\alpha$ MVC was a deterministic one using  $O(n^2/\alpha^2 \cdot \log \alpha)$  bits of space [18]. We have shown that we can remove the logarithmic overhead using randomness. Can we, however, remove this logarithmic factor using deterministic techniques or otherwise prove a deterministic lower bound which shows that it is necessary?

Our work continues the direction set by the results on connectivity [1, 36] and matchings [18, 11]; we resolve the space complexity (up to constant factors) of another problem in the dynamic graph streaming setting. However, other problems still remain open. Hence, can we achieve this for other dynamic graph streaming problems such as dominating set [26] and spectral sparsification [24]?

---

## References

- 1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. doi:10.1137/1.9781611973099.40.
- 2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. doi:10.1145/2213556.2213560.
- 3 KookJin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *International Conference on Machine Learning*, pages 2237–2246. PMLR, 2015.
- 4 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.
- 5 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635. SIAM, 2019. doi:10.1137/1.9781611975482.98.
- 6 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 265–276. ACM, 2019. doi:10.1145/3313276.3316361.
- 7 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for  $(\delta+1)$  vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–786. SIAM, 2019.

- 8 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 627–669. SIAM, 2022. doi:10.1137/1.9781611977073.29.
- 9 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavl'tsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10–12, 2016*, pages 1345–1364. SIAM, 2016. doi:10.1137/1.9781611974331.ch93.
- 10 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pages 354–364. IEEE, 2020. doi:10.1109/FOCS46700.2020.00041.
- 11 Sepehr Assadi and Vihan Shah. An asymptotically optimal algorithm for maximum matching in dynamic streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.9.
- 12 Aaron Bernstein. Improved bounds for matching in random-order streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8–11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 12:1–12:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.12.
- 13 Leyla Biabani, Mark de Berg, and Morteza Monemizadeh. Maximum-weight matching in sliding windows and beyond. In Hee-Kap Ahn and Kunihiro Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6–8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 73:1–73:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.73.
- 14 Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. An optimal algorithm for large frequency moments using  $\tilde{O}(n^{1-2/k})$  bits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
- 15 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10–12, 2016*, pages 1326–1344. SIAM, 2016. doi:10.1137/1.9781611974331.ch92.
- 16 Michael S. Crouch, Andrew McGregor, and Daniel M. Stubbs. Dynamic graphs in the sliding-window model. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2–4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013. doi:10.1007/978-3-642-40450-4\_29.
- 17 Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4–6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 96–104. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.

- 18 Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.30.
- 19 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In Josep Diaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 2004. doi:10.1007/978-3-540-27836-8\_46.
- 20 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 – August 2, 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.
- 21 Piotr Indyk, Eric Price, and David P Woodruff. On the power of adaptivity in sparse recovery. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 285–294. IEEE, 2011.
- 22 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011. doi:10.1145/1989284.1989289.
- 23 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.66.
- 24 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1814–1833. SIAM, 2020. doi:10.1137/1.9781611975994.111.
- 25 Michael Kapralov, Jelani Nelson, Jakub Pachocki, Zhengyu Wang, David P. Woodruff, and Mobin Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 475–486. IEEE Computer Society, 2017.
- 26 Sanjeev Khanna and Christian Konrad. Optimal bounds for dominating set in graph streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 93:1–93:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.93.
- 27 Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 840–852. Springer, 2015. doi:10.1007/978-3-662-48350-3\_70.
- 28 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.74.
- 29 Christian Konrad. Mis in the congested clique model in  $o(\log \log \delta)$  rounds. *arXiv preprint*, 2018. arXiv:1802.07647.

- 30 Christian Konrad. Frequent elements with witnesses in data streams. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 83–95, 2021.
- 31 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. doi:10.1007/978-3-642-32512-0\_20.
- 32 Christian Konrad and Kheeran K. Naidu. On two-pass streaming algorithms for maximum bipartite matching. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.19.
- 33 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.
- 34 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- 35 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2), 2005. doi:10.1561/04000000002.
- 36 Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1844–1860. SIAM, 2019.
- 37 Eric Price and David P Woodruff. Lower bounds for adaptive sparse recovery. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 652–663. SIAM, 2013.
- 38 Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discret. Math.*, 8(2):223–250, 1995.
- 39 Xiaoming Sun and David P Woodruff. Tight bounds for graph problems in insertion streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.
- 40 David P Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 941–960, 2012.



# Approximating LCS and Alignment Distance over Multiple Sequences

Debarati Das ✉ 🏠

Pennsylvania state University, University Park, PA, USA

Barna Saha ✉ 🏠

University of California, San Diego, CA, USA

---

## Abstract

We study the problem of aligning multiple sequences with the goal of finding an alignment that either maximizes the number of aligned symbols (the longest common subsequence (LCS) problem), or minimizes the number of unaligned symbols (the alignment distance aka the complement of LCS). Multiple sequence alignment is a well-studied problem in bioinformatics and is used routinely to identify regions of similarity among DNA, RNA, or protein sequences to detect functional, structural, or evolutionary relationships among them. It is known that exact computation of LCS or alignment distance of  $m$  sequences each of length  $n$  requires  $\Theta(n^m)$  time unless the Strong Exponential Time Hypothesis is false. However, unlike the case of two strings, fast algorithms to approximate LCS and alignment distance of multiple sequences are lacking in the literature. A major challenge in this area is to break the triangle inequality. Specifically, by splitting  $m$  sequences into two (roughly) equal sized groups, then computing the alignment distance in each group and finally combining them by using triangle inequality, it is possible to achieve a 2-approximation in  $\tilde{O}_m(n^{\lceil \frac{m}{2} \rceil})$  time. But, an approximation factor below 2 which would need breaking the triangle inequality barrier is not known in  $O(n^{\alpha m})$  time for any  $\alpha < 1$ . We make significant progress in this direction.

First, we consider a semi-random model where, we show if **just one** out of  $m$  sequences is  $(p, B)$ -pseudorandom then, we can get a below-two approximation in  $\tilde{O}_m(nB^{m-1} + n^{\lceil \frac{m}{2} \rceil + 3})$  time. Such semi-random models are very well-studied for two strings scenario, however directly extending those works require one but all sequences to be pseudorandom, and would only give an  $O(\frac{1}{p})$  approximation. We overcome these with significant new ideas. Specifically an ingredient to this proof is a new algorithm that achieves below 2 approximations when alignment distance is large in  $\tilde{O}_m(n^{\lceil \frac{m}{2} \rceil + 2})$  time. This could be of independent interest.

Next, for LCS of  $m$  sequences each of length  $n$ , we show if the optimum LCS is  $\lambda n$  for some  $\lambda \in [0, 1]$ , then in  $\tilde{O}_m(n^{\lceil \frac{m}{2} \rceil + 1})$  time, we can return a common subsequence of length at least  $\frac{\lambda^2 n}{2 + \epsilon}$  for any arbitrary constant  $\epsilon > 0$ . In contrast, for two strings, the best known subquadratic algorithm may return a common subsequence of length  $\Theta(\lambda^4 n)$ .

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis

**Keywords and phrases** String Algorithms, Approximation Algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.54

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2110.12402>

**Funding** *Barna Saha*: Partly supported by NSF 1652303, 1909046, and HDR TRIPODS 1934846 grants, and an Alfred P. Sloan Fellowship.

---

<sup>1</sup> In the context of multiple sequence alignment, we use  $\tilde{O}_m$  to hide factors like  $c^m \log^m n$  where  $c$  is a constant.



© Debarati Das and Barna Saha;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 54; pp. 54:1–54:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

Given  $m$  sequences each of length  $n$ , we are interested to find an alignment that either maximizes the number of aligned characters (the longest common subsequence problem (LCS)), or minimizes the number of unaligned characters (the minimum alignment distance problem, aka the complement of LCS)<sup>2</sup>. Both these problems are extremely well-studied, are known to be notoriously hard, and form the cornerstone of *multiple sequence alignment* [29, 26, 16], which according to the survey in Nature is one of the most widely used modeling methods in biology [31]. Long back in 1978, the multi-sequence LCS problem (and therefore, the minimum alignment distance problem) was shown to be NP Hard [23]. Moreover, for any constant  $\delta > 0$ , the multi-sequence LCS and alignment distance cannot be approximated within  $n^{1-\delta}$  unless  $P = NP$  [18]. These hardness results hold even under restricted conditions such as for sequences over relatively small alphabet [9], or with certain structural properties [10]. Various other multi-sequence based problems such as finding the median or center string are shown “hard” by reduction from the minimum alignment distance problem [25]. Interested readers may refer to the chapter entitled “Multi String Comparison-the Holy Grail” of the book [16] for a comprehensive study on this topic.

From a fine-grained complexity viewpoint, an  $O(n^{m-\epsilon})$  algorithm to compute alignment distance of  $m$  sequences for any constant  $\epsilon > 0$  will refute the Strong Exponential Time Hypothesis (SETH) [1]. On the other hand, a basic dynamic programming solves these problems in time  $O(mn^m)$ . This raises the question whether we can solve these problems faster in  $O(n^{\alpha m})$  time for  $\alpha < 1$  by allowing approximation. The approximation vs running time trade-off for  $m = 2$  (edit distance problem) has received extensive attention over the last two decades with many recent breakthroughs [21, 7, 6, 8, 3, 5, 11, 14, 27, 13, 19, 4]. To bypass the worst-case hardness in the two-strings setting, multiple prior works have studied semi-random models for sequence comparisons [2, 20, 12]. Semi-random models may capture real-life scenarios better where adversarial examples are rare. In addition, it may also carry several inherent difficulties of the worst case model. Thus studying semi-random models can be a stepping stone towards attacking the worst-case model.

More than a decade back, such a study was initiated by Andoni and Krauthgamer [2], where the authors studied smoothed complexity of sequence alignment. They proposed a semi-random model as follows: first, an adversary chooses two binary strings of length  $n$  and a longest common subsequence  $A$  of them. Then, every character is perturbed independently with probability  $p$ , except that  $A$  is perturbed in exactly the same way inside the two strings. Kuszmaul further generalized this model and considered one input string to be pseudorandom (any pair of disjoint substrings are at large edit distance) whereas the other input string can be adversarial [20]. Both of these works [2, 20] provide  $O(1)$  approximation of the edit distance in almost linear time, and face the triangle inequality barrier. Recently Boroujeni, Seddighin, and Seddighin [12] improved the approximation guarantee to  $(1 + \epsilon)$  thus bypassing the triangle inequality hardness while increasing the running time from near-linear to subquadratic. Therefore, if we aim to generalize the pseudorandom model for multiple strings, it is not obvious how to achieve the best of the above two results simultaneously: (i) a running time of  $O(n^{m/2})$ , and (ii) a below 2 approximation. Another major issue is that any direct generalization of [2, 20, 12] for multiple strings require all but one string to be pseudorandom.

---

<sup>2</sup> While one may define alignment distance in many different ways among multiple sequences, taking the complement of LCS is possibly the cleanest way of defining such a distance both because indel distance between two strings naturally generalizes to it, and due to its close connection to LCS.

In this work we consider a much stronger model where only one input string is pseudorandom and the rest  $m - 1$  input strings can be adversarial. We show how it is possible to accomplish the best of both the running time and the approximation guarantee by giving an  $\tilde{O}(n^{m/2+3})$  time algorithm that breaks the triangle inequality and computes truly below 2 approximation of alignment distance. Towards this we first design an algorithm that takes as input  $m$  adversarially chosen strings and provides a below 2 approximation of their alignment distance in time  $\tilde{O}(n^{m/2+2})$  provided the distance is large. Note this model considers all strings to be adversarial and thus can be of independent interest. Moreover we show that these techniques can be extended to design an algorithm computing constant approximation of the LCS of  $m$  strings in time  $\tilde{O}(n^{m/2+2})$  provided the length of the LCS is large.

It is interesting to note that our results on LCS implies a constant approximation of LCS of three strings is possible in the large distance regime in quadratic time (a reduction from cubic to quadratic time complexity), whereas a worse constant approximation is currently known in the large distance regime for the LCS of two strings to go below the quadratic running time [28].

**Contributions.** We now describe our results in more details.

**Minimizing Alignment Distance of Multiple Sequences with One Pseudorandom String.**

Let  $\mathcal{L}(s_1, \dots, s_m)$  denote the length of LCS of  $m$  strings  $s_1, s_2, \dots, s_m$  (each of length  $n$ ) and  $\mathcal{A}(s_1, \dots, s_m) = n - \mathcal{L}(s_1, \dots, s_m)$  denote the optimal alignment distance of  $s_1, s_2, \dots, s_m$ . We consider the case where one input string is pseudorandom out of  $m$  strings, and the rest of  $m - 1$  strings are chosen adversarially. We provide an algorithm that breaks the triangle inequality barrier and provides  $(2 - \frac{3p}{512} + \epsilon)$  (for any arbitrary small constant  $\epsilon > 0$ ) approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(nB^{m-1} + n^{\lfloor m/2 \rfloor + 3})$ . Formally we show the following.

► **Definition 1** ( $(p, B)$ -pseudorandom). *Given a string  $s$  of length  $n$  and parameters  $p, B \geq 0$  where  $p$  is a constant, we call  $s$  a  $(p, B)$ -pseudorandom string if for any two disjoint  $B$  length substrings  $x, y$  of  $s$ ,  $\mathcal{A}(x, y) \geq pB$ .*

► **Theorem 2.** *Given a  $(p, B)$ -pseudorandom string  $s_1$ , and  $m - 1$  adversarial strings  $s_2, \dots, s_m$  of length  $n$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes  $(2 - \frac{3p}{512} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(nB^{m-1} + n^{\lfloor m/2 \rfloor + 3})$ .*

The theorem can be extended to get a  $c(1 - \frac{3p}{1024} + \epsilon)$  approximation in  $\tilde{O}_m(nB^{\lceil 2m/c \rceil - 1} + n^{\lceil m/c \rceil + 3})$  time. Assuming  $c$  to be even, we divide the input strings into  $\frac{c}{2}$  groups each containing at most  $\lceil \frac{2m}{c} \rceil$  strings. Then for each group we compute a below 2-approximation of the alignment distance in time  $\tilde{O}_m(nB^{\lceil 2m/c \rceil - 1} + n^{\lceil m/c \rceil + 3})$ . Finally, we apply triangle inequality  $\frac{c}{2}$  times to combine these groups to get a  $\frac{c}{2}(2 - \frac{3p}{512} + \epsilon) = c(1 - \frac{3p}{1024} + \frac{\epsilon}{2})$  approximation.

**What do we know in the two strings case?** Let us contrast this result to what is known for  $m = 2$  case [2, 20, 12]. When one of the two strings is  $(p, B)$ -pseudorandom, Kuszmaul gave an algorithm that runs in time  $\tilde{O}(nB)$  time but only computes an  $O(\frac{1}{p})$  (can be large constant) approximation to edit distance [20]. Boroujeni, Seddighin and Seddighin consider a different random model for string generation under which they give a  $(1 + \epsilon)$  approximation but in subquadratic time [12]. While their model captures the case when one string is generated uniformly at random, it does not extend to pseudorandom strings. In fact, there is no result in the two strings case that breaks the triangle inequality barrier and provides

below-2 approximation when one of the strings is pseudorandom. Moreover, in order to apply their technique to multi-string setting, *we would need all but one string to be generated according to their model*. Interestingly, our algorithm obtains all the desired results and provides below-2 approximation of alignment distance with just *one pseudorandom string*. We stress that this is one of the important contributions of our work and is technically involved.

**Key Tool: breaking triangle inequality for large alignment distance.** To construct the above mentioned algorithm, we first design an algorithm that takes as input  $m$  adversarially chosen strings and provides truly below 2 approximation of their alignment distance provided the distance is large. More generally we show if  $\mathcal{A}(s_1, \dots, s_m) = \theta n$  then for any arbitrary small constant  $\epsilon > 0$ , it is possible to obtain a  $c(1 - \frac{3\theta}{32} + \epsilon)$  approximation<sup>3</sup> in time  $\tilde{O}_m(n^{\lceil m/c \rceil + 2})$  time.

► **Theorem 3.** *Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$  such that  $\mathcal{A}(s_1, \dots, s_m) = \theta n$ , where  $\theta \in (0, 1)$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes a  $(2 - \frac{3\theta}{16} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 2})$ . Moreover, for any integer  $c > 0$ , there exists an algorithm that computes  $c(1 - \frac{3\theta}{32} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lceil m/c \rceil + 2})$ .*

For constant  $\theta$ , the above theorem asserts that there exists an algorithm that breaks the triangle inequality barrier and computes a truly below 2-approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 2})$ . Note here all the input strings are adversarially chosen and this result can be of independent interest. Moreover we show these techniques can be extended to compute LCS of multiple strings.

**LCS of Multiple Sequences.** We show if  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$  for some  $\lambda \in [0, 1]$ , then we can return a common subsequence of length  $\frac{\lambda^2 n}{2+\epsilon}$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 1})$ . To contrast, we can get a quadratic algorithm for  $m = 3$  with  $\frac{\lambda}{2+\epsilon}$  approximation (for any arbitrary small constant  $\epsilon > 0$ ), whereas the best known bound for  $m = 2$  case may return a subsequence of length  $\Theta(\lambda^4 n)$  in  $\tilde{O}(n^{1.95})$  time [28].

► **Theorem 4.** *Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$  such that  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$ , where  $\lambda \in [0, 1]$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes an  $\frac{\lambda}{2+\epsilon}$  approximation of  $\mathcal{L}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 1})$ .*

## 1.1 Technical Overview

### Notation

We use the following notations throughout the paper. Given  $m$  strings  $s_1, \dots, s_m$ , each of length  $n$  over some alphabet set  $\Sigma$ , the longest common subsequence (LCS) of  $s_1, \dots, s_m$ , denoted by  $LCS(s_1, \dots, s_m)$  is one of the longest sequences that is present in each  $s_i$ . Define  $\mathcal{L}(s_1, \dots, s_m) = |LCS(s_1, \dots, s_m)|$ . The optimal alignment distance (AD) of  $s_1, \dots, s_m$ , denoted by  $\mathcal{A}(s_1, \dots, s_m)$  is  $n - \mathcal{L}(s_1, \dots, s_m)$ .

For a given string  $s$ ,  $s[i]$  represents the  $i$ th character of  $s$  and  $s[i, j]$  represents the substring of  $s$  starting at index  $i$  and ending at index  $j$ . Given a LCS  $\sigma$  of  $s_1, \dots, s_m$  define  $\sigma(s_j) \subseteq [n]$  be the set of indices such that for each  $k \in \sigma(s_j)$ ,  $s_j[k]$  is aligned in  $\sigma$  and  $\bar{\sigma}(s_j) \subseteq [n]$  be the

<sup>3</sup> We will assume  $c$  is even for simplicity. But all the algorithms work equally well if  $c$  is odd.

set of indices of the characters in  $s_j$  that are not aligned in  $\sigma$ . Define the *alignment cost* of  $\sigma$  to be  $|\bar{\sigma}(s_1)|$  and the *cumulative alignment cost* of  $\sigma$  to be  $\sum_{j=1}^m |\bar{\sigma}(s_j)| = m|\bar{\sigma}(s_1)|$ . Given a set  $T \subseteq [n]$  and a string  $s$ , let  $s^T$  denote the subsequence of  $s$  containing characters with indices in  $T$ .

Given a string  $s$ , we define a window  $w$  of size  $d$  of  $s$  to be a substring of  $s$  having length  $d$ . Given  $m$  strings  $s_1, \dots, s_m$ , we define a  $m$ -window tuple to be a set of  $m$  windows denoted by  $(w_1, \dots, w_m)$ , where  $w_j$  is a window of string  $s_j$ .

Given two characters  $a, b \in \Sigma$ ,  $a \circ b$  represents the concatenation of  $b$  after  $a$ . Given two string  $x, y$ ,  $x \circ y$  represents the concatenation of string  $y$  after  $x$ . For notational simplicity we use  $\tilde{O}_m$  to hide factors like  $c^m \log^m n$ , where  $c$  is a constant. Moreover we use  $\tilde{O}$  to hide polylog factors.

### 1.1.1 Breaking the Triangle Inequality Barrier for Large Alignment Distance and Approximating LCS

We first give an overview of our algorithms leading to Theorem 5 (Section 2). Let us consider the problem of minimizing the alignment distance. Given  $m$  (say  $m$  is even) sequences  $s_1, s_2, \dots, s_m$  each of length  $n$ , partition them into two groups  $G_1 = \{s_1, s_2, \dots, s_{m/2}\}$  and  $G_2 = \{s_{m/2+1}, \dots, s_m\}$ . Suppose the optimum alignment distance of the  $m$  sequences is  $d = \theta n$ . With each alignment, we can associate a set of indices of  $s_1$  that are not aligned in that alignment. Let  $\sigma^*$  be an optimum alignment and  $\bar{\sigma}^*(s_1)$  be that set. We have  $|\bar{\sigma}^*(s_1)| = d$ . Let  $\mathcal{X}_1 = \{(\sigma_i, \bar{\sigma}_i(s_1))\}$  denote all possible alignments  $\sigma_i$  of  $G_1$  of cost at most  $d$ ,  $|\bar{\sigma}_i(s_1)| \leq d$ . Then  $(\sigma^*, \bar{\sigma}^*(s_1)) \in \mathcal{X}_1$ . Therefore, if we can (i) find all possible alignments  $\mathcal{X}_1$ , and (ii) for each  $(\sigma_i, \bar{\sigma}_i(s_1)) \in \mathcal{X}_1$  can verify if that is a valid alignment of  $G_2$ , we can find an optimal alignment.

Unfortunately, it is possible that  $|\mathcal{X}_1| = \sum_{l \leq d} \binom{n}{l}$  which is prohibitively large. Therefore, instead of trying to find all possible alignments, we try to find a *cover* for  $\mathcal{X}_1$  using a few alignments  $(\tau_j, \bar{\tau}_j(s_1))$ ,  $j = 1, 2, \dots, k$  such that for any  $(\sigma_i, \bar{\sigma}_i(s_1)) \in \mathcal{X}_1$ , there exists a  $(\tau_j, \bar{\tau}_j(s_1))$  with large  $|\bar{\sigma}_i(s_1) \cap \bar{\tau}_j(s_1)|$ . In fact, one of the key ingredients of our algorithm is to show such a covering exists and can be obtained in time (roughly)  $n^{|G_1|}$ . With just  $k = \frac{4}{\theta}$  alignments, we show it is possible to cover  $\mathcal{X}_1$  such that for any  $(\sigma_i, \bar{\sigma}_i(s_1)) \in \mathcal{X}_1$ , there exists a  $(\tau_j, \bar{\tau}_j(s_1))$  having  $|\bar{\sigma}_i(s_1) \cap \bar{\tau}_j(s_1)| \geq \frac{3\theta^2 n}{16}$ .

The algorithm to compute the covering starts by finding any optimal alignment  $(\sigma_1, \bar{\sigma}_1(s_1))$  of  $G_1$ . Next it finds another alignment  $(\sigma_2, \bar{\sigma}_2(s_1))$  of cost at most  $d$  which is *farthest* from  $(\sigma_1, \bar{\sigma}_1(s_1))$ , that is  $|\bar{\sigma}_1(s_1) \cap \bar{\sigma}_2(s_1)|$  is minimized. We find these alignments using dynamic programming. If  $|\bar{\sigma}_1(s_1) \cap \bar{\sigma}_2(s_1)| \sim |\bar{\sigma}_2(s_1)|$ , then it stops. Otherwise, it finds another alignment  $(\sigma_3, \bar{\sigma}_3(s_1))$  such that  $|\bar{\sigma}_1(s_1) \cup \bar{\sigma}_2(s_1) \cap \bar{\sigma}_3(s_1)|$  is minimized. We show the process terminates after at most  $\frac{4}{\theta}$  rounds.

Suppose without loss of generality,  $|\bar{\sigma}^*(s_1) \cap \bar{\tau}_1(s_1)| \geq \frac{3\theta^2 n}{16}$ . Given  $\bar{\tau}_1(s_1), \bar{\tau}_2(s_1), \dots, \bar{\tau}_k(s_1)$ , for each  $(\tau_i, \bar{\tau}_i(s_1))$ , we find an alignment  $(\rho_i, \bar{\rho}_i(s_1))$  of  $G_2 \cup s_1$  of cost at most  $d$  such that  $\bar{\rho}_i(s_1)$  is *nearest* to  $\bar{\tau}_i(s_1)$ , that is  $|\bar{\rho}_i(s_1) \cap \bar{\tau}_i(s_1)|$  is maximized. Then, we must have  $|\bar{\rho}_i(s_1) \cap \bar{\tau}_1(s_1)| \geq |\bar{\sigma}^*(s_1) \cap \bar{\tau}_1(s_1)| \geq \frac{3\theta^2 n}{16}$ . Our alignment cost is  $\min_j (|\bar{\tau}_j(s_1) \cup \bar{\rho}_j(s_1)|) \leq |\bar{\tau}_1(s_1) \cup \bar{\rho}_1(s_1)| \leq 2d - \frac{3\theta^2 n}{16} = d(2 - \frac{3\theta}{16})$  giving the desired below-2 approximation when  $\theta$  is a constant.

Of course, there are two main parts in this algorithm that we have not elaborated; given a set of indices  $T$  of  $s_1$ , and a group of strings  $G$ , we need to find an alignment of cost at most  $d$  of  $G \cup s_1$  that is farthest from (nearest to)  $T$ . In general, any application that needs to compute multiple diverse (or similar) alignments can be benefited by such subroutines. We

can use dynamic programming to solve these problems; however, it is important to keep in mind – an alignment that has minimum cost may not necessarily be the farthest (or nearest). Thus, we need to check all possible costs up to the threshold  $d$  to find such an alignment.

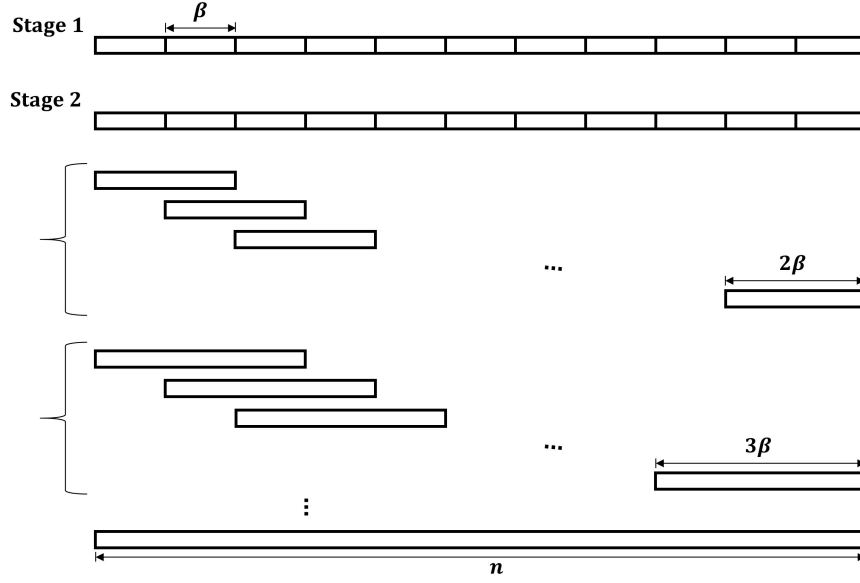
Our algorithm for obtaining a  $\frac{\lambda}{(2+\epsilon)}$  approximation for multi-sequence LCS is nearly identical to the above, and in fact simpler. This helps us to improve the running time slightly (contrast Theorem 4 with Theorem 5). Moreover, the result holds irrespective of the size of LCS. We provide the details in Section 4.

### 1.1.2 Approximating Alignment Distance with just One Pseudorandom String

Next we consider the case where the input consists of a single  $(p, B)$  pseudorandom string and  $m - 1$  adversarial strings each of length  $n$ . We give an overview of our algorithm that returns a below 2 approximation of the optimal alignment distance (even for small regime) proving Theorem 2. The details are provided in Section 3.

In most of the previous literature for computing edit distance of two strings, the widely used framework first partitions both the input strings into windows (substrings) and finds distance between all pairs of windows. Then using dynamic program all these subsolutions are combined to find the edit distance between the input strings. However instead of considering two arbitrary strings as input if one input is  $(p, B)$  pseudorandom, then as we know that any pair of disjoint windows from the pseudorandom string have large edit distance, if we consider a window from the adversarial string then by triangle inequality, there exists at most one window in the pseudorandom string with which it can have small edit distance ( $\leq \frac{pB}{4}$ ). We call this low cost match between an adversarial string window and a pseudorandom string window a *unique match*. Notice if we can identify one such unique match that is part of an optimal alignment, we can put restriction on the indices where the rest of the substrings can be matched. This observation still holds for multiple strings but only when we compare a pair of windows, one from the pseudorandom string and the other from an adversarial string. Thus it is not obvious how we can extend this restriction on pairwise matching to a matching of  $m$ -window tuples as  $(m - 1)$ -window tuples come from  $(m - 1)$  different adversarial strings and their *unique matches* with the pseudorandom string can be very different from each other. Another drawback of this approach is that, to optimize the running time, here the algorithm aims to identify only the matchings with low cost i.e.  $< \frac{pB}{4}$ . Hence the best approximation ratio we can hope for is  $O(1/p)$  which can be a large constant.

Therefore to shed the approximation factor below 2, we also need to find a good approximation of the cost of pair of windows having distance  $\geq \frac{pB}{4}$ . We call a matching with cost  $\geq \frac{pB}{4}$  a *large cost match*. However as the unique match property fails here, without having any prior knowledge about the optimal alignment we need to compute the cost for all pairs of windows having large cost. However doing it trivially can not provide us the desired running time. Fortunately, as  $p$  is a constant  $pB/4 = \Omega(B)$  and thus we can use our large alignment distance approximation algorithm to get a improved running time while ensuring below 2 approximation of the cost for these *large cost match* tuples. Though this simple idea seems promising, if we try to compute an approximation over all large distance  $m$ -window tuples the running time can become as large as  $\tilde{O}_m(n^{\frac{11m}{16}})$ . We show this with an example. Given  $m$  input strings, start by partitioning each string into windows of length  $\beta = n^{\frac{5}{8}}$  (for simplicity assume the windows are disjoint). Hence there are  $n^{\frac{3}{8}}$  windows in each string. Now there can be as many as  $n^{\frac{3m}{8}}$  many  $m$ -window tuples of large cost. If we evaluate each of their cost using our *large alignment distance* algorithm then time taken for each  $m$ -tuple is roughly  $\tilde{O}_m(n^{\frac{5m}{16}})$ . Hence total time required is  $\tilde{O}_m(n^{\frac{11m}{16}})$ . Note to improve this running



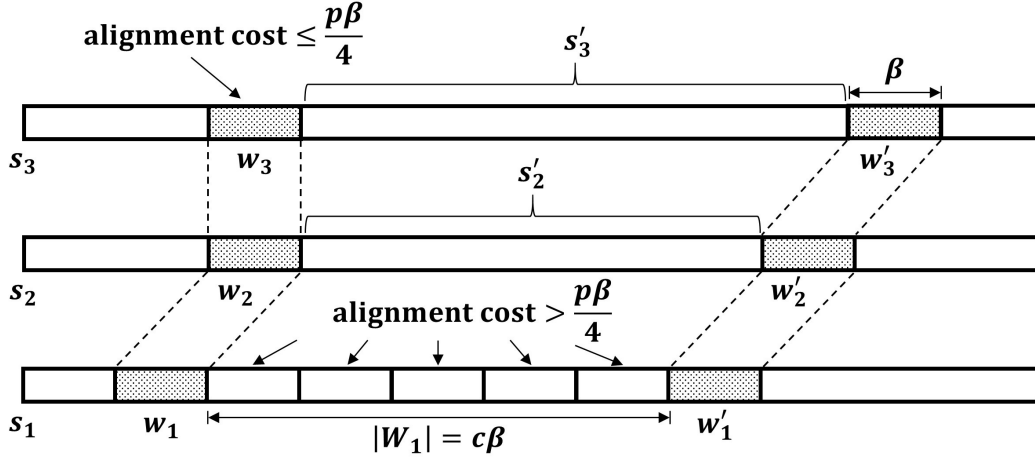
■ **Figure 1** Construction of windows for the  $(p, B)$ -pseudorandom string  $s_1$ .

time by reducing the number of windows, we can not grow the window size arbitrarily large as in that case identifying the unique (low cost) match  $m$ -window tuples will become more time consuming. Later we show for window size  $\beta$  this time can be  $\tilde{O}_m(\beta^m)$ .

Thus to further reduce the running time, instead of evaluating all window tuples having large cost match, we restrict the computation by estimating the cost of only those tuples that maybe necessary to compute an optimal alignment. This is challenging as we do not have any prior information of the optimal alignment. For this purpose we use an adaptive strategy where depending on the unique matches computed so far, we perform a restricted search to estimate the cost of window tuples having large alignment distance. Moreover the length of these windows are also decided adaptively. We remark that this adaptive strategy differs significantly from the previous windowing strategies where the window lengths are fixed to start with. Next we give a brief overview of the three main steps of our algorithm. In Step 1, we provide the construction of windows of the input strings that will be used as input to Step 2 and 3. In Step 2, we estimate the alignment cost of the  $m$ -window tuples such that the matching is unique i.e. the optimal alignment distance is at most  $pB/4$ . In step 3, we further find an approximation of the cost of  $m$ -window tuples that are relevant for an optimal alignment and have large cost i.e.  $\geq pB/4$ .

### 1.1.2.1 Step 1

The windows of the input strings are constructed in two stages. In stage one, we follow a rather straightforward strategy similar to the one used in [15] and partition the  $(p, B)$  pseudorandom string into  $\frac{n}{\beta}$  disjoint windows each of size  $\beta$  (except the right most one). Here  $\beta = \max(B, \sqrt{n})$ . For the rest of the strings we generate a set of overlapping variable sized windows. If the distance threshold parameter is  $\theta$  and the error tolerance parameter is  $\epsilon$ , then for each adversarial strings we generate windows of size  $\{(\beta - \theta\beta), (1 + \epsilon)(\beta - \theta\beta), (1 + \epsilon)^2(\beta - \theta\beta), \dots, (\beta + \theta\beta)\}$  and from starting indices in  $\{1, \epsilon\theta\beta + 1, 2\epsilon\theta\beta + 1, \dots\}$ . These windows are fed as an input to Step 2 of our algorithm.



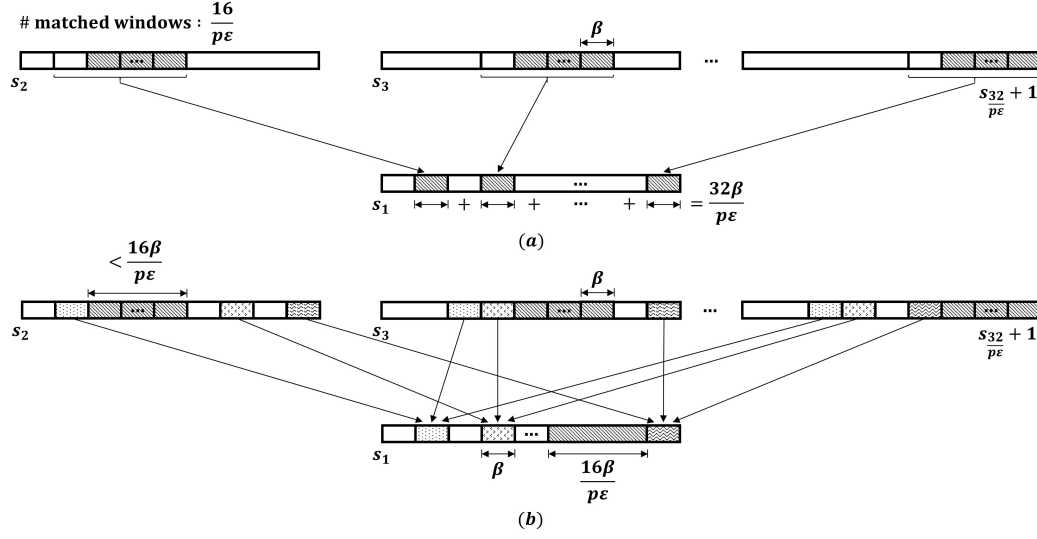
■ **Figure 2** An example of large cost match.

The second stage is much more involved where the strategy significantly differs from the previous literature. The primary difference here is that instead of just considering a fixed length partition of the pseudorandom string (e.g.  $\beta$  in stage one), we take variable sizes that are multiple of  $\beta$  i.e.  $\{\beta, 2\beta, \dots, n\}$  and the windows start from indices in  $\{1, \beta+1, 2\beta+1, \dots\}$ . Next for each adversarial string, we adaptively try to guess a set of useful substrings/windows that can be matched with the large cost windows of the pseudorandom string under the optimal alignment (we fix one for the analysis purpose). These guesses are guided by the unique low cost matches found in Step 2. Next for each such useful substring/window from the adversarial string and each length in  $\{\beta, 2\beta, \dots, n\}$ , we create a set of overlapping windows as described in stage one. Note the windows created in this stage are used as an input to Step 3 of the algorithm.

We explain the motivation behind the variable sized window partitioning for the pseudorandom string and the restricted window construction for the adversarial strings with an example(see Figure 2).

Let we are given three strings  $s_1, s_2, s_3$  each of length  $n$  as input where  $s_1$  is  $(p, B)$ -pseudorandom. We divide all three strings in windows (for simplicity assume these windows are disjoint) of size  $\beta$ . For the analysis purpose fix an optimal alignment where window  $w_1$  is aligned with window  $w_2, w_3$  and window  $w'_1$  is aligned with window  $w'_2, w'_3$ . Moreover their costs are  $\leq \frac{p\beta}{4}$ . Hence we get an estimation of the alignment cost of these window tuples at Step 2. Also assume that all the windows appearing between  $w_1$  and  $w'_1$  in  $s_1$  has alignment cost  $> \frac{p\beta}{4}$  but  $\leq \frac{\beta}{2}$ . Hence, for these windows we can not use a trivial maximum cost of  $\beta$  in order to get a below two approximation. Here we estimate the cost of these windows using Algorithm LargeAlign(). Now if we compute this estimation separately for each  $\beta$  size window between  $w_1$  and  $w'_1$  in  $s_1$ , as there can be as many as  $\sqrt{n}$  (assume  $\beta = \sqrt{n}$ ) such windows and each call to algorithm LargeAlign() takes time  $O(n^{m/4})$ , the total running time will be  $O(n^{3m/4})$ . Thus to achieve the promised running time, instead of considering all small  $\beta$  size windows separately we consider the whole substring between  $w_1$  and  $w'_1$  as one single window  $W_1$  and compute its cost estimation. Observe as we do not have the prior information about the optimal alignment (and therefore  $w_1$  and  $w'_1$ ) we try all possible lengths in  $\beta, 2\beta, \dots, n$ . The overall idea here is to use a window length, so that we can represent the whole substring with large optimal alignment distance lying between two windows having low cost unique match in the optimal alignment with a single window.





**Figure 3** (a) Each hatch window of  $s_1$  is matched with  $\frac{16}{p\epsilon}$  windows of one adversarial string. Deleting all these windows deletes  $\frac{32\beta}{p\epsilon}$  characters from each string. (b) The optimal alignment deletes  $< \frac{16\beta}{p\epsilon}$  characters from each string.

### 1.1.2.2 Step 2

In step 2, we start with a set of windows, each of size  $\beta$ , generated from strings  $s_1, \dots, s_m$  (assume  $s_1$  to be the  $(p, B)$ -pseudorandom string). Our objective is to identify all  $m$ -window tuples that have optimal alignment distance  $\leq \frac{p\beta}{4}$ . Here we use the fact that for every adversarial window there exists at most one window in the pseudorandom string at distance  $\leq \frac{p\beta}{4}$ . We start the algorithm by computing for each window  $w$  from the pseudorandom string, and for each adversarial string  $s_j$  the set  $S_j^w$  containing all windows from  $s_j$  that are at distance  $\leq \frac{p\beta}{4}$  from  $w$ . Notice for two string case, if for a pseudorandom window  $w$ ,  $|S_j^w| \geq \frac{16}{p\epsilon}$  (i.e. many windows from  $s_j$  are close to  $w$ ), then as in the optimal alignment at most one adversarial window from  $S_j^w$  can be matched with  $w$  with cost  $\leq \frac{p\beta}{4}$  and the rest of the windows from  $S_j^w$  have cost at least  $\frac{p\beta}{4}$ , the optimal cost for all the windows from  $S_j^w$  is  $\geq \frac{\beta}{\epsilon}$ . Thus even if we take a cost estimation  $\beta$  (maximum cost) for the pseudorandom window  $w$  this still gives a  $(1 + \epsilon)$  approximation of the alignment cost. For multiple strings we can not use a similar idea as for a pseudorandom window  $w$  we can not take a trivial cost estimation if for only one adversarial string, there are many windows which are close to  $w$ . Thus we need to device a new strategy.

We explain with an example. Consider  $k$  windows  $w_1, \dots, w_k$  (where  $k \geq \frac{32}{p\epsilon}$ ) from the pseudorandom string such that for each  $w_j$ , there are  $\frac{16}{p\epsilon}$  windows from the adversarial string  $s_{j+1}$  that are at distance  $\leq \frac{p\beta}{4}$  with  $w_j$  and for every other string there is exactly one window with distance  $\leq \frac{p\beta}{4}$ . Here following the above argument if for each of the  $k$  pseudorandom windows, we take a trivial cost estimation of  $\beta$  then the total cost will be  $k\beta \geq \frac{32\beta}{p\epsilon}$ . Whereas the optimal cost can be  $\frac{16\beta}{p\epsilon}$  (check Figure 3 (a)).

Therefore for every window  $w_i$  of the pseudorandom string, we count the total number of windows in all adversarial strings that are at distance  $\leq \frac{p\beta}{4}$  and if the count is at least  $\frac{16m}{p\epsilon}$ , then we take a trivial cost estimation of  $|w_i|$  for  $w_i$ . (Note here for two different windows from the pseudorandom string, the sets of close windows from the adversarial strings are disjoint.)

Otherwise if the count is small let  $\mathcal{W}_j$  be the set of windows from string  $s_j$  that are close to  $w_i$ . Then we can bound  $|\mathcal{W}_2| \times \dots \times |\mathcal{W}_m| \leq [(|\mathcal{W}_2| + \dots + |\mathcal{W}_m|)/(m-1)]^{m-1} \leq (\frac{32}{pe})^m = \tilde{O}_m(1)$  and for each choice of  $m$ -window tuples in  $w_i \times \mathcal{W}_2 \times \dots \times \mathcal{W}_m$ , we find the exact alignment cost in time  $\tilde{O}_m(\beta^m)$ . As there are  $n/\beta$  disjoint windows in the pseudorandom string we can bound the total running time by  $\tilde{O}_m(n\beta^{m-1})$ .

### 1.1.2.3 Step 3

In this step, our objective is to find a cost estimation for all the windows of the pseudorandom string that have large alignment cost in the optimal alignment. Consider Figure 2 (consider  $m = 3$ ), and let  $W_1$  be such a window from  $s_1$ . Also assume that it can not be extended to left or right, i.e. both  $w_1$  and  $w'_1$  have small cost match which we have already identified in Step 2. Let the length of  $W_1$  is  $c\beta$  where  $c \geq 1$  and assume no  $\beta$  length window of  $W_1$  has small cost match. Note, we have the assurance that window  $W_1$  is generated in Step 1. Next to find the match of  $W_1$ , instead of checking whole  $s_2$  and  $s_3$ , we consider the substring between  $w_2$  and  $w'_2$  in  $s_2$  and  $w_3$  and  $w'_3$  in  $s_3$ . Now if the sum of the length of all the substrings is large (i.e.  $|s'_2| + |s'_3| \geq 5m|W_1| = 15|W_1|$ ), we can claim that the cost of  $W_1$  in the optimal alignment is very large and we can take a trivial cost estimation of  $|W_1|$ . Otherwise, we can ensure that the total choices for  $m$ -window tuples that need to be evaluated for  $W_1$  is at most  $\tilde{O}_m(1)$  and for each of them, we calculate a below 2 approximation of the cost using Algorithm LargeAlign(). In the algorithm, as we don't know the optimal alignment, for every window of the pseudorandom string generated in Step 1 stage two, we assume it to be large cost and check whether the  $\beta$  length window appearing just before and after this window has a small cost unique match (notice this observation is very crucial to decide the maximal length of any large cost window). If not we discard it and consider a larger window. Otherwise we use LargeAlign() to find its cost estimation.

Overall in Step 2 and 3, we ensure that for every window of the pseudorandom string, our algorithm provides  $< 2$  cost approximation. Moreover if the window has small cost match then in Step 2 we evaluate the cost of at most  $\tilde{O}_m(1)$   $m$ -window tuples where each cost estimation takes time  $\tilde{O}_m(\beta^m)$  (as the window size is  $\beta$ ) and otherwise if the cost is large we use algorithm LargeAlign() that computes an approximation in time  $\tilde{O}_m(n^{m/2})$  (here the window length can be as large as  $n$ ). As the number of windows generated for string  $s_1$  is polynomial in  $n$ , taking  $\beta = \max(B, \sqrt{n})$  we get the required running time bound.

## Organization

In Section 2, we give the algorithm for minimizing alignment distance when the distance is large. Section 3 provides the details of the below-2 approximation algorithm for alignment distance when one string is  $(p, B)$  pseudorandom. In Section 4, we give our  $\frac{\lambda}{2+\epsilon}$  approximation algorithm for multi-sequence LCS.

## 2 Below-2 Approximation for Multi-sequence Alignment Distance

In this section, we provide an algorithm LargeAlign() that given  $m$  strings  $s_1, \dots, s_m$  each of length  $n$ , such that  $\mathcal{A}(s_1, \dots, s_m) = \theta n$ , where  $\theta \in (0, 1)$  computes a  $(2 - \frac{3\theta}{16} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 2})$ . Notice when  $\theta = \Omega(1)$ , this implies a below 2 approximation of  $\mathcal{A}(s_1, \dots, s_m)$ .

► **Theorem 5.** *Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$  such that  $\mathcal{A}(s_1, \dots, s_m) = \theta n$ , where  $\theta \in (0, 1)$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes a  $(2 - \frac{3\theta}{16} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 2})$ . Moreover, there exists an algorithm that computes a  $c(1 - \frac{3\theta}{32} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lceil m/c \rceil + 2})$ .*

As we do not have any prior knowledge of  $\theta$ , instead of proving the theorem directly, we solve the following gap version for a given fixed threshold  $\theta$ . We define the gap version as follows.

**GapMultiAlignDist( $s_1, \dots, s_m, \theta, c$ ).** Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$ ,  $\theta \in (0, 1)$  and a constant  $c > 1$ , decide whether  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$  or  $\mathcal{A}(s_1, \dots, s_m) > c\theta n$ . More specifically if  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$  we output 1, else if  $\mathcal{A}(s_1, \dots, s_m) > c\theta n$  we output 0, otherwise output any arbitrary answer.

► **Theorem 6.** *Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$  and a parameter  $\theta \in (0, 1)$ , there exists an algorithm that computes  $\text{GapMultiAlignDist}(s_1, \dots, s_m, \theta, (2 - \frac{3\theta}{16}))$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 2})$ .*

**Proof of Theorem 5 from Theorem 6.** Let us consider an arbitrary small constant  $\epsilon' > 0$ , and fix a sequence of parameters  $\theta_0, \theta_1, \dots$  as follows: for  $i = 0, 1, \dots, \log_{1+\epsilon'} n$ ,  $\theta_i = 1/(1 + \epsilon')^i$ . Find the largest  $i$  such that  $\text{GapMultiAlignDist}(s_1, \dots, s_m, \theta_i, (2 - \frac{3\theta_i}{16})) = 1$ . Let  $\mathcal{A}(s_1, \dots, s_m) = \theta n$ . Then there exists a  $\theta_i$ , such that  $\theta_i n \geq \theta n > \theta_{i+1} n$  as  $\theta_i = (1 + \epsilon')\theta_{i+1}$ . In this case the algorithm outputs a value at most  $(2 - \frac{3\theta_i}{16})\theta_i n \leq (2 - \frac{3\theta}{16} + \epsilon')(2 - \frac{3\theta}{16})\theta n$ . As  $\theta \geq 1/n$ , by appropriately scaling  $\epsilon'$ , we get the desired running time of Theorem 5. ◀

The rest of the section is dedicated towards proving Theorem 6. Before providing the the algorithm for computing  $\text{GapMultiAlignDist}(s_1, \dots, s_m, \theta_i, (2 - \frac{3\theta_i}{16}))$ , we first outline another two algorithms that will be used as subroutines in our main algorithm.

## 2.1 Finding Alignment with Maximum Deletion Similarity

Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$ , a set  $S \subseteq [n]$  and a parameter  $0 \leq d \leq n$ , our objective is to compute an alignment  $\sigma_n$  of  $s_1, \dots, s_m$  with alignment cost at most  $d$  such that  $|\bar{\sigma}_n(s_1) \cap S|$  is maximized.

► **Theorem 7.** *Given  $m$  strings  $s_1, \dots, s_m$ , each of length  $n$ , a set  $S \subseteq [n]$  and a parameter  $0 \leq d \leq n$ , there exists an algorithm that computes a common alignment  $\sigma_n$  such that  $\sum_{k \in [m]} |\bar{\sigma}_n(s_k)| \leq dm$  and  $|\bar{\sigma}_n(s_1) \cap S|$  is maximized in time  $\tilde{O}(2^m m n^{m+1})$ .*

We design an algorithm  $\text{MaxDelSimilarAlignment}()$  that uses dynamic programming to compute  $\sigma_n$ . We defer the details to the full version.

If  $d = \theta n$  where  $\theta \in (0, 1)$ , then as we know from every string no more than  $\theta n$  characters will be deleted, using [30] we can claim the following.

► **Corollary 8.** *Given  $m$  strings  $s_1, \dots, s_m$ , each of length  $n$ , a set  $S \subseteq [n]$  and a parameter  $d = \theta n$ , where  $\theta \in (0, 1)$  there exists an algorithm that computes a common alignment  $\sigma_n$  such that  $\sum_{k \in [m]} |\bar{\sigma}_n(s_k)| \leq dm$  and  $|\bar{\sigma}_n(s_1) \cap S|$  is maximized in time  $\tilde{O}(2^m \theta^m m n^{m+1})$ .*

## 2.2 Finding Alignment with Minimum Deletion Similarity

Given  $m$  strings  $s_1, \dots, s_m$  of length  $n$ ,  $q$  sets  $S_1, \dots, S_q \subseteq [n]$  and a parameter  $0 \leq d \leq n$ , our objective is to compute an alignment  $\sigma_n$  of  $s_1, \dots, s_m$  with alignment cost at most  $d$  such that  $|\cup_{j \in [q]} (\bar{\sigma}_n(s_1) \cap S_j)|$  is minimized.

► **Theorem 9.** *Given  $m$  strings  $s_1, \dots, s_m$ , each of length  $n$ ,  $q$  sets  $S_1, \dots, S_q \subseteq [n]$  and a parameter  $0 \leq d \leq n$ , there exists an algorithm that computes a common alignment  $\sigma_n$  such that  $\sum_{k \in [m]} |\bar{\sigma}_n(s_k)| \leq dm$  and  $|\cup_{j \in [q]} (\bar{\sigma}_n(s_1) \cap S_j)|$  is minimized in time  $\tilde{O}(2^m mn^{m+1})$ .*

We design an algorithm `MinDelSimilarAlignment()` that uses dynamic programming just like the one used in finding alignment with maximum deletion similarity to compute  $\sigma_n$ . We defer the details to the full version. If  $d = \theta n$  where  $\theta \in (0, 1)$ , again using [30] we can claim the following. Note in this case we only need to compute  $\theta^m n^m$  entries in the dynamic program table.

► **Corollary 10.** *Given  $m$  strings  $s_1, \dots, s_m$ , each of length  $n$ ,  $q$  sets  $S_1, \dots, S_q \subseteq [n]$  and a parameter  $0 \leq d \leq \theta n$ , there exists an algorithm that computes a common alignment  $\sigma_n$  such that  $\sum_{k \in [m]} |\bar{\sigma}_n(s_k)| \leq dm$  and  $|\cup_{j \in [q]} (\bar{\sigma}_n(s_1) \cap S_j)|$  is minimized in time  $\tilde{O}(2^m \theta^m mn^{m+1})$ .*

## 2.3 Algorithm for $(2 - \frac{3\theta}{16} + \epsilon)$ -approximation of $\mathcal{A}(s_1, \dots, s_m)$

To compute the value of  $\text{GapMultiAlignDist}(s_1, \dots, s_m, \theta, (2 - \frac{3\theta}{16}))$  the procedure  $\text{GapMultiAlignDist}(s_1, \dots, s_m, \theta)$  calls procedure  $\text{MultiAlign}(s_1, \dots, s_m, \theta)$  that returns a string  $\sigma$ . If  $\sigma$  is a null string it outputs 0 and otherwise it outputs 1.

We show if  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$ ,  $\text{MultiAlign}(s_1, \dots, s_m, \theta)$  computes a common subsequence  $\sigma$  such that  $|\bar{\sigma}(s_1)| \leq (2 - \frac{3\theta}{16})\theta n$  and otherwise if  $\mathcal{A}(s_1, \dots, s_m) > (2 - \frac{3\theta}{16})\theta n$  it computes a null string. Next we describe Algorithm  $\text{MultiAlign}(s_1, \dots, s_m, \theta)$ . It starts by partitioning the input strings into two groups  $G_1$  and  $G_2$  where  $G_1$  contains the strings  $s_1, \dots, s_{\lceil m/2 \rceil}$  and  $G_2$  contains the strings  $s_{\lceil m/2 \rceil + 1}, \dots, s_m$ .

Assume  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$ . Next we state an observation that is used as one of the key elements to conceptualize our algorithm. Any common subsequence of  $s_1, \dots, s_m$  is indeed a common subsequence of  $G_1$ . Therefore, as  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$ , if we can enumerate all common subsequences of  $G_1$  of length at least  $n - \theta n$ , we generate the optimal alignment as well. Notice after generating each common subsequence of  $G_1$ , it can be checked whether it is a common subsequence of  $G_2$  or not. The main hurdle here is that enumerating all common subsequences of  $G_1$  of length at least  $n - \theta n$  is time consuming.

We overcome this barrier by designing Algorithm  $\text{EnumerateAlignments}(s_1, \dots, s_m, \theta)$  that generates  $k$  (where  $k = O(1/\theta)$ ) different sets  $L_1, \dots, L_k$  where  $L_j \subseteq [n]$ ,  $|L_j| \leq \theta n$  and each  $L_j$  corresponds to a common subsequence  $\sigma_j$  of  $G_1$  such that  $L_j = \bar{\sigma}_j(s_1)$ . Moreover, we can ensure that either  $\exists j \in [k]$  where  $|L_j| \leq \frac{3\theta n}{4}$  or for any common subsequence  $\sigma$  of  $G_1$  with  $|\bar{\sigma}(s_1)| \leq \theta n$  there exists a  $L_i$  where  $|\bar{\sigma}(s_1) \cap L_i| \geq \frac{3\theta^2 n}{16}$ .

Algorithm  $\text{EnumerateAlignments}()$  starts by computing a LCS  $\sigma_1$  of  $G_1$  such that  $|\bar{\sigma}_1(s_1)| \leq \theta n$ . Let  $L_1 = \bar{\sigma}_1(s_1)$ . If  $|L_1| \leq \frac{3\theta n}{4}$  return  $L_1$ . Otherwise it calls the algorithm  $\text{MinDelSimilarAlignment}()$  to compute a common subsequence  $\sigma_2$  of  $G_1$  of cost at most  $\theta n$  such that  $L_2 \cap L_1$ , where  $L_2 = \bar{\sigma}_2(s_1)$  is minimized. At the  $i$ th step given  $i - 1$  sets  $L_1, \dots, L_{i-1}$ , the algorithm computes an alignment  $\sigma_i$  of  $G_1$  with  $L_i$  being the set of indices of unaligned characters of  $s_1$  such that the intersection of  $L_i$  with  $\cup_{j \in [i-1]} L_j$  is minimized. The algorithm continues with this process until it reaches a round  $k$  such that  $|\cup_{i \in [k-1]} (L_k(s_1) \cap L_i(s_1))| \geq \frac{\theta^2 n(k-1)}{4}$ . Let  $L_1, \dots, L_k$  be the sets generated. Output all these sets.

Next for each  $L_i$  returned by Algorithm EnumerateAlignments(), call the algorithm MaxDelSimilarAlignment() to find an alignment  $\sigma'$  of  $G_2$  of cost at most  $\theta n$  such that the intersection of  $L_i$  and  $L'_i = \bar{\sigma}'(s_1)$  is maximized. If  $|L_i \cup L'_i| \leq (2 - \frac{3\theta}{16})\theta n$ , define  $\sigma = s_1[i_1] \circ \dots \circ s_1[i_p]$  where,  $[n] \setminus (L_i \cup L'_i) = \{i_1, \dots, i_p\}$ . Output  $\sigma$ .

We now prove two crucial lemmas to establish the correctness.

► **Lemma 11.** *Given strings  $s_1, \dots, s_{|G_1|}$  of length  $n$  such that  $\mathcal{A}(s_1, \dots, s_{|G_1|}) \leq \theta n$ , where  $\theta \in (0, 1)$ , there exists an algorithm that computes  $k \leq 4/\theta$ , different sets  $L_1, \dots, L_k \subseteq [n]$  each of size at most  $\theta n$  such that  $\forall j \in [k]$ , there exists a common subsequence  $\sigma_j$  of  $G_1$  where,  $L_j = \bar{\sigma}_j(s_1)$  and one of the following is true.*

1.  $\exists j \in [k]$  such that  $|L_j| \leq \frac{3\theta n}{4}$ .
2. For any common subsequence  $\sigma$  of  $G_1$  with  $|\bar{\sigma}(s_1)| \leq \theta n$  there exists a  $L_i$ , where  $|\bar{\sigma}(s_1) \cap L_i| \geq \frac{3\theta^2 n}{16}$ . The running time of the algorithm is  $\tilde{O}(2^{mn|G_1|+1})$ .

**Proof.** Let  $\sigma' = \text{LCS}(s_1, \dots, s_{|G_1|})$ . If  $|\sigma'| \geq n - \frac{3\theta n}{4}$ , then  $|\bar{\sigma}'(s_1)| \leq \frac{3\theta n}{4}$  and we satisfy condition 1. Otherwise assume  $|\sigma'(s_1)| > \frac{3\theta n}{4}$ . Let  $L_1, \dots, L_k$  be the sets returned by Algorithm EnumerateAlignments(). By construction, for each  $L_i$  there exists a common subsequence  $\sigma_i$  of  $G_1$  where  $L_i = \bar{\sigma}_i(s_1)$ . Note every  $L_i$  has size at least  $\frac{3\theta n}{4}$ . Moreover if the algorithm does not terminate at round  $i$ , then  $|L_i(s_1) \setminus \{L_1(s_1) \cup \dots \cup L_{i-1}(s_1)\}| \geq \frac{3\theta n}{4} - \frac{\theta^2 n(i-1)}{4}$ . Hence after  $k$  steps we have

$$\begin{aligned} |\cup_{i \in [k]} L_i(s_1)| &\geq \frac{3\theta n}{4} + (\frac{3\theta n}{4} - \frac{\theta^2 n}{4}) + \dots + (\frac{3\theta n}{4} - \frac{(k-1)\theta^2 n}{4}) \\ &= \frac{3k\theta n}{4} - \frac{\theta^2 n}{4}(1 + 2 + \dots + (k-1)) \\ &= \frac{3k\theta n}{4} - \frac{k(k-1)\theta^2 n}{8} \\ &> \frac{3k\theta n}{4} - \frac{k^2\theta^2 n}{8} \end{aligned}$$

Substituting  $k = 4/\theta$ , we get  $|\cup_{i \in [k]} L_i(s_1)| > n$ . Now if the algorithm stops at round  $i < 4/\theta$ , then we know for each common subsequence  $\sigma$  of  $G_1$  of length at least  $n - \theta n$  if  $\sigma \notin \{L_1, \dots, L_{i-1}\}$ ,  $|\cup_{j \in [i-1]} (\sigma(s_1) \cap L_j(s_1))| \geq \frac{(i-1)\theta^2 n}{4}$ . Hence there exists at least one  $j \in [i-1]$  such that  $|L_j(s_1) \cap \sigma(s_1)| \geq \frac{\theta^2 n}{4}$ . Otherwise if the algorithm runs for  $4/\theta$  rounds then  $\cup_{i \in [4/\theta]} L_i(s_1) = [n]$ . Hence for each common subsequence  $\sigma$  of cost in  $[\frac{3\theta n}{4}, \theta n]$ ,  $\bar{\sigma}(s_1)$  will have intersection at least  $\frac{3\theta^2 n}{16}$  with at least one  $L_i$ .

As  $|k| \leq 4/\theta$  the algorithm runs for at most  $4/\theta$  rounds where at the  $i$ th round it calls MinDelSumilarAlignment() with strings in  $G_1$ , and sets  $L_1, \dots, L_{i-1}$  (where  $i \leq 4/\theta$ ) and parameter  $\theta n$ . By Corollary 10 each call to MinDelSumilarAlignment() takes time  $\tilde{O}(2^{|G_1|\theta|G_1|}|G_1|n^{|G_1|+1})$ . Hence the total running time taken is  $\tilde{O}(2^{|G_1|}|G_1|n^{|G_1|+1})$ . ◀

We set  $|G_1| = \lceil \frac{m}{2} \rceil \leq \lfloor m/2 \rfloor + 1$  to obtain a running time of  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} mn^{\lfloor m/2 \rfloor + 2})$ .

► **Lemma 12.** *Given  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$ , Algorithm MultiAlign( $s_1, \dots, s_m, \theta$ ) generates a string  $\sigma$ , such that  $\sigma$  is a common sequence of  $s_1, \dots, s_m$  and the alignment cost of  $\sigma$  is at most  $(2 - \frac{3\theta}{16})\theta n$ . Moreover the running time of the algorithm is  $\tilde{O}_m(n^{\lfloor \frac{m}{2} \rfloor + 2})$ .*

**Proof.** Let  $\eta$  be some LCS of  $s_1, \dots, s_m$  such that  $|\bar{\eta}(s_1)| \leq \theta n$ . First assume  $\mathcal{A}(G_1) \leq \frac{3\theta n}{4}$ . Then Algorithm EnumerateAlignments() computes a LCS  $\sigma_1$  of  $G_1$  and returns the set  $L_1 = |\bar{\sigma}_1(s_1)|$  where  $|L_1| \leq \frac{3\theta n}{4}$  to Algorithm MultiAlign(). Next Algorithm MultiAlign() calls MaxDelSimilarAlignment( $G_2, L_1, \theta n$ ) which returns a set  $L'_1 = \bar{\sigma}'(s_1)$ , where  $\sigma'$

is a common subsequence of  $G_2$  and  $|L'_1| \leq \theta n$ . Notice  $\sigma \leftarrow s_1[i_1] \circ \dots \circ s_1[i_p]$  (where,  $[n] \setminus (L_i \cup L'_i) = \{i_1, \dots, i_p\}$ ) is a common subsequence of  $s_1, \dots, s_m$  and  $|\bar{\sigma}(s_1)| \leq (|L_1| + |L'_1|) \leq (\frac{3\theta n}{4} + \theta n) = (2 - \frac{1}{4})\theta n \leq (2 - \frac{\theta}{4})\theta n$  as  $\theta \in (0, 1)$ . Hence Algorithm MultiAlign() computes a common subsequence  $\sigma$  of  $s_1, \dots, s_m$  such that the alignment cost of  $\sigma$  is at most  $(2 - \frac{\theta}{4})\theta n$ .

Next assume  $\mathcal{A}(G_1) > \frac{3\theta n}{4}$ . Then by Lemma 11, Algorithm EnumerateAlignments() computes a set  $L_i = \bar{\sigma}_i(s_1)$  where  $\sigma_i$  is a common subsequence of  $G_1$ ,  $|L_i| \leq \theta n$  and  $L_i \cap \bar{\eta}(s_1) \geq \frac{3\theta^2 n}{16}$ . Notice as  $\eta$  is a common subsequence of  $G_2$ , when Algorithm MultiAlign() calls MaxDelSimilarAlignmemnt( $G_2, L_i, \theta n$ ), it returns a set  $L'_1 = \bar{\sigma}'(s_1)$ , where  $\sigma'$  is a common subsequence of  $G_2$ ,  $|L'_1| \leq \theta n$  and  $|L_i \cap L'_1| \geq \frac{3\theta^2 n}{16}$ . Therefore  $|L_i \cup L'_1| \leq \theta n + \theta n - \frac{3\theta^2 n}{16} = (2 - \frac{3\theta n}{16})\theta n$ , and Algorithm MultiAlign() computes a common subsequence  $\sigma$  of  $s_1, \dots, s_m$  such that the alignment cost of  $\sigma$  is at most  $(2 - \frac{3\theta}{16})\theta n$ .

Next we analyze the running time of Algorithm MultiAlign(). First we compute  $LCS(G_1)$  which takes time  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} n^{\lfloor m/2 \rfloor + 1})$  using the classic dynamic program algorithm (note  $|G_1| = \lceil \frac{m}{2} \rceil \leq \lfloor \frac{m}{2} \rfloor + 1$ ). Next we call  $EnumerateAlignments(s_1, \dots, s_{m/2}, \theta)$ . By Lemma 11 this takes time  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} mn^{\lfloor m/2 \rfloor + 2})$ . Moreover it returns at most  $O(1/\theta)$  sets and for each of them Algorithm MultiAlign() calls  $MaxDelSimilarAlignment()$  on  $G_2$ . As  $|G_2| \leq \lfloor \frac{m}{2} \rfloor + 1$  and each set has size at most  $\theta n$ , each call takes time  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} \theta^{\lfloor m/2 \rfloor + 1} mn^{\lfloor m/2 \rfloor + 2})$ . Hence total time taken is  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} mn^{\lfloor m/2 \rfloor + 2})$ . Next each union of  $L_i$  and  $L'_i$  and corresponding  $\sigma$  can be computed in time  $O(n)$ . Hence the running time of Algorithm MultiAlign() is  $\tilde{O}(2^{\lfloor m/2 \rfloor + 1} mn^{\lfloor m/2 \rfloor + 2}) = \tilde{O}_m(n^{\lfloor \frac{m}{2} \rfloor + 2})$ . ◀

► **Lemma 13.** *GapMultiAlignDist( $s_1, \dots, s_m, \theta$ ) computes GapMultiAlignDist( $s_1, \dots, s_m, \theta, (2 - \frac{3\theta}{16})$ ) in time  $\tilde{O}_m(n^{\lfloor \frac{m}{2} \rfloor + 2})$ .*

**Proof.** First assume the case where  $\mathcal{A}(s_1, \dots, s_m) \leq \theta n$ . from Lemma 12 we have Algorithm MultiAlign() returns a common subsequence  $\sigma$  of  $s_1, \dots, s_m$  such that the alignment cost of  $\sigma$  is at most  $(2 - \frac{3\theta}{16})\theta n$ . Hence, Algorithm GapMultiAlignDist() outputs 1. Next assume  $\mathcal{A}(s_1, \dots, s_m) > (2 - \frac{3\theta}{16})\theta n$ . In this case in Algorithm MultiAlign(), for each set  $L_i \in \mathcal{E}$ ,  $|L_i \cup L'_i| > (2 - \frac{3\theta}{16})\theta n$ . Hence Algorithm MultiAlign() returns a null string and Algorithm GapMultiAlignDist() outputs 0. The bound on the running time is directly implied by the running time bound of Algorithm MultiAlign(). ◀

### 3 Below-2 Approximation for Multi-sequence Alignment Distance with One Pseudorandom String

In the last section we present an algorithm that given  $m$  strings, computes a truly below 2 approximation of the optimal alignment distance of the input strings provided the distance is large i.e.  $\Omega(n)$ . In this section we use this algorithm as a black box and show given a  $(p, B)$ -pseudorandom string  $s_1$ , and  $m - 1$  adversarial strings  $s_2, \dots, s_m$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes  $(2 - \frac{3p}{512} + \epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n\beta^{m-1} + n^{\lfloor m/2 \rfloor + 3})$ . Here  $\beta = \max(B, \sqrt{n})$ . Notice as the approximation factor is independent of  $\theta = \frac{\mathcal{A}(s_1, \dots, s_m)}{n}$ , we can assure truly below-2 approximation of the alignment cost for any distance regime. Formally we show the following.

► **Theorem 14 (2).** *Given a  $(p, B)$ -pseudorandom string  $s_1$ , and  $m - 1$  adversarial strings  $s_2, \dots, s_m$  each of length  $n$ , there exists an algorithm that for any arbitrary small constant  $\epsilon > 0$  computes  $(2 - \frac{3p}{512} + 89\epsilon)$  approximation of  $\mathcal{A}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n\beta^{m-1} + n^{\lfloor m/2 \rfloor + 3})$ . Here  $\beta = \max(B, \sqrt{n})$ .*

The details of the algorithm and the analysis are provided in the full version.



#### 4 $\frac{\lambda}{2+\epsilon}$ -approximation for Multi-sequence LCS

In this section we provide an algorithm that given  $m$  strings  $s_1, \dots, s_m$  each of length  $n$ , such that  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$ , where  $\lambda \in (0, 1)$  computes an  $\frac{\lambda}{2+\epsilon}$  approximation of  $\mathcal{L}(s_1, \dots, s_m)$ . The algorithm is nearly identical to the algorithm described in Section 2, and in fact slightly simpler which helps us to improve the running time further. In particular, we get the following theorem.

► **Theorem 15 (4).** *For any constant  $\epsilon > 0$ , given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$  such that  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$ , where  $\lambda \in (0, 1)$ , there exists an algorithm that computes an  $\frac{\lambda}{2+\epsilon}$  approximation of  $\mathcal{L}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 1} + mn^2)$ .*

Since we do not have any prior knowledge of  $\lambda$ , we solve a gap version: given  $m$  strings  $s_1, \dots, s_m$  of length  $n$  over some alphabet set  $\Sigma$ ,  $\lambda \in (0, 1)$  and a constant  $c > 1$ , the objective is to decide whether  $\mathcal{L}(s_1, \dots, s_m) \geq \lambda n$  or  $\mathcal{L}(s_1, \dots, s_m) < \frac{\lambda^2 n}{c}$ . More specifically if  $\mathcal{L}(s_1, \dots, s_m) \geq \lambda n$  we output 1, else if  $\mathcal{L}(s_1, \dots, s_m) < \frac{\lambda^2 n}{c}$  we output 0 otherwise output any arbitrary answer. We design an algorithm that decides this gap version for  $c = 2$ . This immediately implies an  $(\frac{\lambda}{2+\epsilon})$  approximation of  $LCS(s_1, s_2, \dots, s_m)$  following a similar logic of going from Theorem 6 to Theorem 5. We now prove Theorem 4.

##### 4.1 Algorithm for $\frac{\lambda}{2+\epsilon}$ -approximation of $LCS(s_1, \dots, s_m)$

We partition the input strings into two groups  $G_1$  and  $G_2$  where  $G_1$  contains the strings  $s_1, \dots, s_{\lceil m/2 \rceil}$  and  $G_2$  contains the strings  $s_{\lceil m/2 \rceil + 1}, \dots, s_m$ . Next compute a longest common subsequence  $L_1$  of  $G_1$  with  $|L_1| \geq \lambda n$ . Remove all aligned characters of  $s_1$  in  $L_1$ . We represent the modified  $s_1$  by  $s_1^{[n] \setminus L_1(s_1)}$ : string  $s_1$  restricted to the characters with indices in  $[n] \setminus L_1(s_1)$ . Compute an LCS  $L_2$  of  $s_1^{[n] \setminus L_1(s_1)}, s_2, \dots, s_{\lceil m/2 \rceil}$ . At  $i$ th step given  $i-1$  common subsequences  $L_1, \dots, L_{i-1}$ , we compute an LCS  $L_i$  of  $s_1^{[n] \setminus \bigcup_{j \in [i-1]} L_j(s_1)}, s_2, \dots, s_{\lceil m/2 \rceil}$ . We continue this process until it reaches a round  $k$  such that  $|L_k| < \lambda n - \frac{\lambda^2 n(k-1)}{2}$ . Let  $L_1, \dots, L_k$  be the sequences generated.

Next for each  $L_i$  returned, we compute a longest common subsequence  $L'_i$  of  $L_i, s_{\lceil m/2 \rceil + 1}, \dots, s_m$ . If there exists an  $i \in [k]$  such that  $|L'_i| \geq \frac{\lambda^2 n}{2}$ , output  $L'_i$ .

► **Lemma 16.** *Given strings  $s_1, \dots, s_{|G_1|}$  of length  $n$  and a parameter  $\lambda n$  as input, where  $\lambda \in (0, 1]$ , there exists a set of  $k \leq 2/\lambda$  different common subsequences  $L_1, \dots, L_k$  of  $s_1, \dots, s_{|G_1|}$  each of length at least  $\lambda n - \frac{\lambda^2 n(k-1)}{2}$  such that for any common subsequence  $\sigma$  of  $s_1, \dots, s_m$  of length at least  $\lambda n$  there exists a  $L_i$ , where  $|\sigma(s_1) \cap L_i(s_1)| \geq \frac{\lambda^2 n}{2}$ . These  $k$  subsequences can be computed in time  $\tilde{O}_m(n^{|G_1|} + mn^2)$ .*

**Proof.** Given  $k$  subsequences  $L_1, L_2, \dots, L_k$  of  $s_1$  such that for each  $i \in [k]$ ,  $|L_i| \geq \lambda n - \frac{\lambda^2 n(k-1)}{4}$ , and  $L_1(s_1), L_2(s_1), \dots$  are disjoint, we have

$$\begin{aligned} |\cup_{i \in [k]} L_i(s_1)| &\geq \lambda n + (\lambda n - \frac{\lambda^2 n}{2}) + \dots + (\lambda n - \frac{(k-1)\lambda^2 n}{2}) \\ &= k\lambda n - \frac{\lambda^2 n}{2}(1 + 2 + \dots + (k-1)) \\ &> k\lambda n - \frac{k^2 \lambda^2 n}{4} \end{aligned}$$

Substituting  $k = 2/\lambda$  we get  $|\cup_{i \in [k]} L_i(s_1)| > n$ . Now if we compute  $L_1, \dots, L_j$  and  $j < 2/\lambda$ , then we know for each common subsequence  $\sigma$  of  $s_1, \dots, s_m$  with length at least  $\lambda n$  if  $\sigma \notin \{L_1, \dots, L_{j-1}\}$ , then  $|\cup_{i \in [j-1]} (\sigma(s_1) \cap L_i(s_1))| \geq \frac{(j-1)\lambda^2 n}{2}$ . Hence there exists at



least one  $i \in [j-1]$  such that  $|L_i(s_1) \cap \sigma(s_1)| \geq \frac{\lambda^2 n}{2}$ . Otherwise if the algorithm runs for  $2/\lambda$  rounds then  $\cup_{j \in [2/\lambda]} L_j(s_1) = [n]$ . Then for any common subsequence  $\sigma$  of  $s_1, \dots, s_m$  with length at least  $\lambda n$ ,  $\sigma$  will have an intersection at least  $\frac{\lambda^2 n}{2}$  with at least one  $L_i$ .

As  $|k| \leq 2/\lambda$  the algorithm runs for at most  $2/\lambda$  rounds where at each round it computes the LCS of  $G_1$  strings such that  $\sum_j |L_j| \leq n$  where  $L_j$ s are pairwise disjoint. This can be performed using Theorem 17 in  $\tilde{O}_m(n^{|G_1|} + mn\lambda) = \tilde{O}_m(n^{|G_1|} + mn^2)$  time. ◀

By setting  $|G_1| = \lceil m/2 \rceil \leq \lfloor \frac{m}{2} \rfloor + 1$ , we get a running time of  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 1} + mn^2)$ . Now by Lemma 16 if  $\sigma$  is an LCS of  $s_1, s_2, \dots, s_m$ , then there exists a  $L_i$  such that  $|L_i(s_1) \cap \sigma(s_1)| \geq \frac{\lambda^2 n}{2}$ . Thus, when we compute the LCS of  $L_i(s_1), s_{\lceil m/2 \rceil + 1}, \dots, s_m$ , we are guaranteed to return a common subsequence of  $s_1, s_2, \dots, s_m$  of length at least  $\frac{\lambda^2 n}{2}$ . Hence, taking the right choice of  $\lambda$  following the gap version we get the claimed approximation bound. Using Theorem 17, the running time to compute a common subsequence of  $L_j(s_1), s_{\lceil m/2 \rceil + 1}, \dots, s_m$  for all  $j$  is  $\tilde{O}_m(n^{\lfloor m/2 \rfloor + 1} + mn^2)$ . This completes the proof of Theorem 4. ◀

---

## References

- 1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78, 2015.
- 2 Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. *ACM Transactions on Algorithms (TALG)*, 8(4):1–25, 2012.
- 3 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 377–386, 2010.
- 4 Alexandr Andoni and Negev Shekel Nosatzki. Edit distance in near-linear time: it’s a constant factor. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, 2020.
- 5 Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM J. Comput.*, 41(6):1635–1648, 2012.
- 6 Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *45th Symposium on Foundations of Computer Science, FOCS 2004*, pages 550–559, 2004.
- 7 Tugkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 316–324, 2003.
- 8 Tugkan Batu, Funda Ergün, and Süleyman Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 792–801, 2006.
- 9 Amey Bhangale, Diptarka Chakraborty, and Rajendra Kumar. Hardness of approximation of (multi-)lcs over small alphabet. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176, pages 38:1–38:16, 2020.
- 10 Guillaume Blin, Laurent Bulteau, Minghui Jiang, Pedro J Tejada, and Stéphane Vialette. Hardness of longest common subsequence for sequences with bounded run-lengths. In *Annual Symposium on Combinatorial Pattern Matching*, pages 138–148, 2012.
- 11 Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: Quantum and mapreduce. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1170–1189, 2018.

- 12 Mahdi Boroujeni, Masoud Seddighin, and Saeed Seddighin. Improved algorithms for edit distance and LCS: beyond worst case. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1601–1620, 2020.
- 13 Joshua Brakensiek and Aviad Rubinstein. Constant-factor approximation of near-linear edit distance in near-linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 685–698, 2020.
- 14 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucký, and Michael E. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 979–990, 2018.
- 15 Elazar Goldenberg, Aviad Rubinstein, and Barna Saha. Does preprocessing help in fast sequence comparisons? In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 657–670, 2020.
- 16 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
- 17 Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664–675, 1977.
- 18 Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24(5):1122–1139, 1995.
- 19 Michal Koucký and Michael E. Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 699–712, 2020.
- 20 William Kuszmaul. Efficiently approximating edit distance between pseudorandom strings. In *Proceedings of the thirtieth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1165–1180. SIAM, 2019.
- 21 Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27(2):557–582, 1998.
- 22 Gad M. Landau and Uzi Vishkin. Fast string matching with k differences. *J. Comput. Syst. Sci.*, 37(1):63–78, 1988.
- 23 David Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM)*, 25(2):322–336, 1978.
- 24 Eugene W. Myers. An  $O(ND)$  difference algorithm and its variations. *Algorithmica*, 1(2):251–266, 1986.
- 25 François Nicolas and Eric Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms*, 3(2-4):390–415, 2005.
- 26 Pavel A Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics*, 52(6):1763–1779, 1992.
- 27 Aviad Rubinstein. Approximating edit distance, 2018.
- 28 Aviad Rubinstein, Saeed Seddighin, Zhao Song, and Xiaorui Sun. Approximation algorithms for LCS and LIS with truly improved running times. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 1121–1145, 2019.
- 29 Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22:4673–4680, 1994.
- 30 Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.
- 31 Nuzzo R. Van Noorden R, Maher B. The top 100 papers. *Nature*, 2014.

### A An $\tilde{O}(m^2 k^m)$ algorithm for Alignment Distance of Multiple Sequences with $\tilde{O}(mn)$ Preprocessing

We first recall an algorithm developed in [30, 21, 22, 24] that computes edit distance between two strings in  $O(n + k^2)$  time.

#### Warm-up: An $O(n + k^2)$ algorithm for Edit Distance

The well-known dynamic programming algorithm computes an  $(n + 1) \times (n + 1)$  edit-distance matrix  $D[0..n][0..n]$  where entry  $D[i, j]$  is the edit distance,  $ED(A^i, B^j)$  between the prefixes  $A[1, i]$  and  $B[1, j]$  of  $A$  and  $B$ , where  $A[1, i] = a_1 a_2 \dots a_i$  and  $B[1, j] = b_1 b_2 \dots b_j$ . The following is well-known and easy to verify coupled with the boundary condition  $D[i, 0] = D[0, i] = i$  for all  $i \in [0, n]$ .

For all  $i, j \in [0, n]$

$$D[i, j] = \min \begin{cases} D[i - 1, j] + 1 & \text{if } i > 0; \\ D[i, j - 1] + 1 & \text{if } j > 0; \\ D[i - 1, j - 1] + 1(a_i \neq b_j) & \text{if } i, j > 0. \end{cases}$$

The computation cost for this dynamic programming is  $O(n^2)$ . To obtain a significant cost saving when  $ED(A, B) \leq k \ll n$ , the  $O(n + k^2)$  algorithm works as follows. It computes the entries of  $D$  in a greedy order, computing first the entries with value 0, 1, 2, ...,  $k$  respectively. Let diagonal  $d$  of matrix  $D$ , denotes all  $D[i, j]$  such that  $j = i + d$ . Therefore, the entries with values in  $[0, k]$  are located within diagonals  $[-k, k]$ . Now since the entries in each diagonal of  $D$  are non-decreasing, it is enough to identify for every  $d \in [-k, k]$ , and for all  $h \in [0, k]$ , the last entry of diagonal  $d$  with value  $h$ . The rest of the entries can be inferred automatically. Hence, we are overall interested in identifying at most  $(2k + 1)k$  such points. The  $O(n + k^2)$  algorithm shows how building a suffix tree over a combined string  $A\$B$  (where  $\$$  is a special symbol not in  $\Sigma$ ) helps identify each of these points in  $O(1)$  time, thus achieving the desired time complexity.

Let  $L^h(d) = \max\{i : D[i, i + d] = h\}$ . The  $h$ -wave is defined by  $L^h = \langle L^h(-k), \dots, L^h(k) \rangle$ . Therefore, the algorithm computes  $L^h$  for  $h = 0, \dots, k$  in the increasing order of  $h$  until a wave  $e$  is computed such that  $L^e(0) = n$  (in that case  $ED(A, B) = e$ ), or the wave  $L^k$  is computed in the case the algorithm is thresholded by  $k$ . Given  $L^{h-1}$ , we can compute  $L^h$  as follows.

Define

$$Equal(i, d) = \max_{q \geq i} (q \mid A[i, q] = B[i + d, q])$$

Then,  $L^0(0) = Equal(0, 0)$  and

$$L^h(d) = \max \begin{cases} Equal(L^{h-1}(d) + 1, d) & \text{if } h - 1 \geq 0; \\ Equal(L^{h-1}(d - 1), d) & \text{if } d - 1 \geq -k, h - 1 \geq 0; \\ Equal(L^{h-1}(d + 1) + 1, d) & \text{if } d + 1, h + 1 \leq k. \end{cases}$$

Using a suffix tree of the combined string  $A\$B$ , any  $Equal(i, d)$  query can be answered in  $O(1)$  time, and we get a running time of  $O(n + k^2)$ .

### An $\tilde{O}(m^2 k^m)$ algorithm for Alignment Distance of Multiple Sequences with $\tilde{O}(nm)$ Preprocessing

We now extend the above  $\tilde{O}(n + k^2)$  algorithm to computing alignment distance of  $m$  strings. Recall that we are given  $m$  strings  $s_1, s_2, \dots, s_m$  each of length  $n$ . The following is an  $O(n^m)$  time-complexity dynamic programming to obtain the edit distance of  $m$  strings. We fill up an

$m$ -dimensional dynamic programming matrix  $D[[0, n] \dots [0, n]]$  where the entry  $D[i_1, i_2, \dots, i_m]$  computes the edit distance among the prefixes  $s_1[1, i_1], s_2[1, i_2], \dots, s_m[1, i_m]$ . As a starting condition, we have  $D[0, \dots, 0] = 0$ . Let  $\vec{e}_i = [0, 0, \dots, \underbrace{1}_{i\text{th index}}, 0, 0, \dots]$ , and  $\mathbb{V}_j$  represents an  $m$ -dimensional vector  $\langle j, j, j, \dots, j \rangle$ . The dynamic programming is given by the following recursion. For all  $i_1, i_2, \dots, i_m \in [0, n]$

$$D[i_1, i_2, \dots, i_m] = \min \begin{cases} D[\langle i_1, i_2, \dots, i_m \rangle - \vec{e}_j] + 1 & \text{for all } j = 1, 2, \dots, m \\ & \text{if } \langle i_1, i_2, \dots, i_m \rangle - \vec{e}_j \geq \langle 0, 0, \dots, 0 \rangle; \\ D[i_1 - 1, i_2 - 1, \dots, i_m - 1] & \text{if } s_1[i_1] = s_2[i_2] = \dots = s_m[i_m] \text{ and} \\ & i_1, i_2, \dots, i_m > 0. \end{cases}$$

In order to compute  $D[i_1, i_2, \dots, i_m]$ , we can either delete an element  $s_j[i_j]$ , or align  $s_1[i_1], s_2[i_2], \dots, s_m[i_m]$  if they all match. The time to compute each entry  $D[i_1, i_2, \dots, i_m]$  is  $m$ . The overall running time is  $O(mn^m)$ .

**Observation.** In order to design an  $\tilde{O}(m^2 k^m)$  algorithm with preprocessing  $\tilde{O}(mn)$ , first observe that if  $\mathcal{A}(s_1, s_2, \dots, s_m) \leq k$  then it is not possible that in the final alignment, we have  $m$  indices  $i_1, i_2, \dots, i_m$  aligned to each other such that  $\max |i_j - i_k|, j, k \in [1, 2, \dots, m] > k$ . Since all strings have equal length, this would imply a total number of deletions  $> mk$ , or  $\mathcal{A}(s_1, s_2, \dots, s_m) > k$ .

**Algorithm.** Let diagonal  $\vec{d}$  of matrix  $D$  denotes an  $(m-1)$  dimensional vector, and contains all  $D[i_1, i_2, \dots, i_m]$  such that  $\langle i_2, i_3, \dots, i_m \rangle = \mathbb{V}_{i_1} + \vec{d}$ . Let  $D_k = \{\vec{d} \mid \max_j |d[j]| \leq k\}$ . Then  $|D_k| = (2k+1)^{m-1}$  since each entry  $i_j \in i_i + \{-k, -k+1, \dots, 0, 1, \dots, k\}$ , for  $j = 2, 3, \dots, m$ . We want to identify the entries with values in  $[0, km]$  located within diagonals  $D_k$ .

We similarly define  $L^h(d) = \max\{i : D[i, \mathbb{V}_i + \vec{d}] = h\}$ . The  $h$ -wave is defined by  $L^h = \{L^h(\vec{d}) \mid \vec{d} \in D_k\}$ . Therefore, the algorithm computes  $L^h$  for  $h = 0, \dots, km$  in the increasing order of  $h$  until a wave  $e$  is computed such that  $L^e(\vec{0}) = n$  (in that case  $\mathcal{A}(A, B) = e$ ), or the wave  $L^{km}$  is computed in the case the algorithm is thresholded by  $k$ . Given  $L^{h-1}$ , we can compute  $L^h$  as follows.

Define

$$Equal(i, \vec{d}) = \max_{q \geq i} (q \mid s_1[i, q] = s_2[i + d[1], q]) = s_3[i + d[2], q] = \dots = s_m[i + d[m-1], q]).$$

That is  $Equal(i, \vec{d})$  computes the longest prefix of the first string starting at index  $i$  that can be matched to all the other strings following diagonal  $\vec{d}$ .

Next, we define the neighboring diagonals  $N_1(\vec{d})$  and  $N_2(\vec{d})$  of  $\vec{d}$ .

$$N_1(\vec{d}) = \{\vec{d}' \mid \|\vec{d} - \vec{d}'\|_1 = 1 \text{ \& } \vec{d}' < \vec{d}\}.$$

$$N_2(\vec{d}) = \{\vec{d}' = \vec{d} + \mathbb{V}_{+1}\}.$$

Then,  $L^0(0) = Equal(0, \vec{0})$  and

$$L^h(\vec{d}) = \max \begin{cases} Equal(L^{h-1}(\vec{d}' \in N_1(\vec{d})), \vec{d}) & \text{if } \vec{d}' \in D_k, h-1 \geq 0; \\ Equal(L^{h-1}(\vec{d}' \in N_2(\vec{d})) + 1, \vec{d}) & \text{if } \vec{d}' \in D_k, h-1 \geq 0. \end{cases}$$

Next, we show that it is possible to preprocess  $s_i, i = 1, 2, \dots, m$  separately so that even then each  $Equal(i, \vec{d})$  query can be implemented in  $O(m \log n)$  time.

### Preprocessing Algorithm

The preprocessing algorithm constructs  $\log(n) + 1$  hash tables for each string  $s$ . The  $\ell$ -th hash table corresponds to window size  $2^\ell$ ; we use a rolling hash function (e.g. Rabin fingerprint) to construct a hash table of all contiguous substrings of  $s$  of length  $2^\ell$  in time  $O(n)$ . Since there are  $\log n + 1$  levels, the overall preprocessing time for  $s$  is  $O(n \log n)$ . Let  $H_{s_i}[\ell]$  store all the hashes for windows of length  $2^\ell$  of  $s_i$  for  $i = 1, 2, \dots, m$ . Hence the total preprocessing time is  $\tilde{O}(nm)$ .

### Answering $Equal(i, \vec{d})$ in $O(m \log n)$ time

$Equal(i, \vec{d})$  queries can be implemented by doing a simple binary search over the presorted hashes in  $O(m \log n)$  time. Suppose  $Equal(i, \vec{d}[j]) = q_j$ . We identify the smallest  $\ell \geq 0$  such that  $q_j < 2^\ell$ , and then do another binary search for  $q_j$  between  $i + 2^{\ell-1}$  to  $i + 2^\ell$ . Finally, we set  $Equal(i, \vec{d}) = \min(q_2, \dots, q_m)$ .

## **B** An $\tilde{O}_m(\lambda n^m)$ Algorithm for Multi-sequence LCS

► **Theorem 17.** *Given  $m$  strings  $s_1, \dots, s_m$  each of length  $n$  such that  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$  where  $\lambda \in (0, 1)$ , there exists an algorithm that computes  $\mathcal{L}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(\lambda n^m + nm)$ .*

The algorithm is build over the algorithm of [17], that given two strings  $x, y$  of length  $n$  such that  $\mathcal{L}(x, y) = \lambda n$ , computes  $\mathcal{L}(x, y)$  in time  $\tilde{O}(\lambda n^2)$ . Though the algorithm of [17] shares a similar flavor with the classical quadratic time dynamic program algorithm, the main contribution of this work is that it introduces the concept of minimal  $\ell$ -candidates that ensure that to compute the LCS, instead of enumerating the whole DP, it is enough to compute some selective entries that are important. Moreover they show if the LCS is small then the total number of minimal  $\ell$ -candidates can be bounded. Also they can be constructed efficiently.

### An $\tilde{O}(\lambda n^2)$ Algorithm for LCS

We first provide a sketch of the algorithm of [17]. We start with a few notations. Given two indices  $i, j \in [n]$ , let  $\mathcal{L}(i, j)$  denotes the length of the LCS of  $x[1, i]$  and  $y[1, j]$  and  $x_i$  denotes the  $i$ th character of string  $x$ .

Given indices  $i, j$  we call  $\langle i, j \rangle$  an  $\ell$ -candidate if  $x_i = y_j$  and  $\exists i', j' \in [n]$  such that  $i' < i, j' < j$  and  $\langle i', j' \rangle$  is an  $(\ell - 1)$ -candidate. We say that  $\langle i, j \rangle$  is generated over  $\langle i', j' \rangle$ . Also define  $\langle 0, 0 \rangle$  to be the 0-candidate. (for this purpose add a new symbol  $\alpha$  at the beginning of both  $x, y$ . Hence  $x[0] = y[0] = \alpha$ ) With this definition using induction we can claim that  $\langle i, j \rangle$  is an  $\ell$ -candidate iff  $\mathcal{L}(i, j) \geq \ell$  and  $x_i = y_j$ . Moreover as  $\mathcal{L}(x, y) = \lambda n$ , the maximum value of  $\ell$  for which there exists an  $\ell$ -candidate is  $\lambda n$ . Hence to compute the LCS what we need to do is to construct a sequence of 0-candidate, 1-candidate,  $\dots$ ,  $(\lambda n - 1)$ -candidate and a  $\lambda n$ -candidate such that the  $i$ th candidate can be generated from the  $(i - 1)$ th candidate. Note as for each  $i$ , there can be many  $\ell$ -candidates enumerating all of them will be time consuming.

Therefor the authors bring the notion of minimal  $\ell$ -candidate that are generated as follows. Consider two  $\ell$ -candidates  $\langle i_1, j_1 \rangle$  and  $\langle i_2, j_2 \rangle$ . If  $i_1 \geq i_2$  and  $j_1 \geq j_2$ , then it is enough to keep only  $\langle i_2, j_2 \rangle$  as any  $(\ell + 1)$ -candidate that is generated from  $\langle i_1, j_1 \rangle$ , can be generated from  $\langle i_2, j_2 \rangle$  as well. Call  $\langle i_1, j_1 \rangle$  a spurious candidate.

► **Lemma 18.** *Let the set  $\{ \langle i_\ell, j_\ell \rangle, \ell \in \{1, 2, \dots\} \}$  denotes the set of  $\ell$ -candidates. After discarding all spurious  $\ell$ -candidates it can be claimed that  $i_1 < i_2 < \dots$  and  $j_1 > j_2 > \dots$ .*

**Proof.** For any two  $\ell$ -candidates  $\langle i_1, j_1 \rangle$  and  $\langle i_2, j_2 \rangle$ , either 1)  $i_1 < i_2$  and  $j_1 \leq j_2$  or 2)  $i_1 < i_2$  and  $j_1 > j_2$  or 3)  $i_1 = i_2$  and  $j_1 \leq j_2$  or 4)  $i_1 = i_2$  and  $j_1 > j_2$ . In the first and third case  $\langle i_2, j_2 \rangle$  is spurious and in the forth case  $\langle i_1, j_1 \rangle$  is spurious. Hence after removing all spurious candidates it can be ensured that  $i_1 < i_2 < \dots$  and  $j_1 > j_2 > \dots$ . ◀

The  $\ell$ -candidates which are left after the removal of all spurious candidates are called minimal  $\ell$ -candidates. Notice as for each  $i$  there is at most one minimal  $\ell$  candidate, total number of minimal  $\ell$ -candidates for all choices of  $i$  and  $\ell$  is at most  $\lambda n^2$ . Using this bound and Lemma 3 in [17], an algorithm can be designed to compute all the minimal  $\ell$ -candidates and thus  $\mathcal{L}(x, y)$  in time  $\tilde{O}(\lambda n^2)$ .

### Generalization for $m$ strings

Now we provide an upper bound on the number of minimal  $\ell$ -candidates for  $m$  strings given  $\mathcal{L}(s_1, \dots, s_m) = \lambda n$ . Given indices  $i_1, \dots, i_m$  we call  $\langle i_1, \dots, i_m \rangle$  an  $\ell$ -candidate if  $s_1[i_1] = \dots = s_m[i_m]$  and  $\exists i'_1, \dots, i'_m \in [n]$  such that  $i'_j < i_j$ , and  $\langle i'_1, \dots, i'_m \rangle$  is an  $(\ell - 1)$ -candidate. We say that  $\langle i_1, \dots, i_m \rangle$  is generated over  $\langle i'_1, \dots, i'_m \rangle$ . Similar to the two string case using induction we can prove  $\langle i_1, \dots, i_m \rangle$  is an  $\ell$ -candidate iff  $\mathcal{L}(s_1, \dots, s_m) \geq \ell$  and  $s_1[i_1] = \dots = s_m[i_m]$ . Therefore to compute  $\mathcal{L}(s_1, \dots, s_m)$  it will be enough to generate a sequence of 0-candidate, 1-candidate,  $\dots$ ,  $(\lambda n - 1)$ -candidate and a  $\lambda n$ -candidate such that the  $i$ th candidate can be generated from the  $(i - 1)$ th candidate. Next we describe the notion of spurious candidates and bound the total number of minimal  $\ell$ -candidates.

For two  $\ell$ -candidates  $\langle i_1, \dots, i_{m-2}, i_{m-1}, i_m \rangle$  and  $\langle i_1, \dots, i_{m-2}, i'_{m-1}, i'_m \rangle$ , if  $i_{m-1} \geq i'_{m-1}$  and  $i_m \geq i'_m$  then we call the tuple  $\langle i_1, \dots, i_{m-2}, i_{m-1}, i_m \rangle$  spurious and discard it as any  $(\ell + 1)$  tuple that is generated from  $\langle i_1, \dots, i_{m-2}, i_{m-1}, i_m \rangle$  can be generated from  $\langle i_1, \dots, i_{m-2}, i'_{m-1}, i'_m \rangle$  as well. The tuples that survives are called minimal  $\ell$ -candidates. Hence following a similar argument as given for Lemma 18, we can claim the following.

► **Lemma 19.** *Let the set  $\{ \langle i_1, \dots, i_{m-2}, i_{m-1}^\ell, i_m^\ell \rangle, \ell \in \{1, 2, \dots\} \}$  denotes the set of  $\ell$ -candidates for fixed values of  $i_1, \dots, i_{m-2}$ . After discarding all spurious  $\ell$ -candidates it can be claimed that  $i_{m-1}^1 < i_{m-1}^2 < \dots$  and  $i_m^1 > i_m^2 > \dots$ .*

Note this implies that for a fixed choice of  $i_1, \dots, i_{m-1}$ , there exists at most one minimal  $\ell$ -candidate. As  $\ell = \lambda n$ , total number of minimal  $\ell$ -candidates over all choices of  $i_1, \dots, i_{m-1}$  and  $\ell$  is at most  $\lambda n^m$ .

Next we state a lemma that is a generalisation of Lemma 3 of [17] for  $m$  strings.

► **Lemma 20.** *For  $\ell \geq 1$   $\langle i_1, \dots, i_{m-2}, i_{m-1}, i_m \rangle$  is a minimal  $\ell$ -candidate iff  $\langle i_1, \dots, i_{m-2}, i_{m-1}, i_m \rangle$  is a  $\ell$ -candidate with the minimum  $m$ th coordinate value such that low  $< i_m < high$  where high is the minimum  $m$ th coordinate value of all  $\ell$ -candidates having first  $m - 2$  coordinate values  $i_1, \dots, i_{m-2}$  and the  $(m - 1)$ th coordinate value less than  $i_{m-1}$  and low is the minimum  $m$ th coordinate value of all  $(\ell - 1)$ -candidates having first  $m - 2$  coordinate values  $i_1, \dots, i_{m-2}$  and the  $(m - 1)$ th coordinate value less than  $i_{m-1}$ .*

Together with the above lemma and the bound on the number of minimal  $\ell$ -candidates following the algorithm of [17], we can design an algorithm that computes  $\mathcal{L}(s_1, \dots, s_m)$  in time  $\tilde{O}_m(\lambda n^m + nm)$ .





# A Primal-Dual Algorithm for Multicommodity Flows and Multicuts in Treewidth-2 Graphs

Tobias Friedrich 

Hasso Plattner Institute, Universität Potsdam, Germany

Davis Issac 

Hasso Plattner Institute, Universität Potsdam, Germany

Nikhil Kumar 

Hasso Plattner Institute, Universität Potsdam, Germany

Nadym Mallek 

Hasso Plattner Institute, Universität Potsdam, Germany

Ziena Zeif 

Hasso Plattner Institute, Universität Potsdam, Germany

---

## Abstract

We study the problem of multicommodity flow and multicut in treewidth-2 graphs and prove bounds on the multiflow-multicut gap. In particular, we give a primal-dual algorithm for computing multicommodity flow and multicut in treewidth-2 graphs and prove the following approximate max-flow min-cut theorem: given a treewidth-2 graph, there exists a multicommodity flow of value  $f$  with congestion 4, and a multicut of capacity  $c$  such that  $c \leq 20f$ . This implies a multiflow-multicut gap of 80 and improves upon the previous best known bounds for such graphs. Our algorithm runs in polynomial time when all the edges have capacity one. Our algorithm is completely combinatorial and builds upon the primal-dual algorithm of Garg, Vazirani and Yannakakis for multicut in trees and the augmenting paths framework of Ford and Fulkerson.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Approximation Algorithms, Multicommodity Flow, Multicut

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2022.55

**Category** APPROX

**Funding** This research was partially funded by the HPI Research School on Data Science and Engineering.

*Ziena Zeif*: HPI Research School.

## 1 Introduction

Given an undirected graph with edge capacities and  $k$  source-sink pairs, the *maximum multicommodity flow problem* asks for the maximum amount of flow that can be routed between the source-sink pairs. If the flows are restricted to be integral, then the problem is called the *maximum integral multicommodity flow*. An important special case of this problem is the *maximum edge disjoint paths problem*, where the objective is to find the maximum number of source-sink pairs that can simultaneously be connected by edge-disjoint paths. In a multicommodity flow with *congestion*  $c$ , an edge may be used by up to  $c$  flow paths. The maximum edge disjoint paths problem is NP-Hard, even in very restricted settings such as when the graph is series-parallel [14]. Maximum edge disjoint paths problem is hard to approximate in general (even with congestion, see Section 2.1 for further discussion). Multicommodity flow problems have been studied extensively over the last five decades and find extensive applications in VLSI design, routing and wavelength assignment etc. [17].



© Tobias Friedrich, Davis Issac, Nikhil Kumar, Nadym Mallek, and Ziena Zeif;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022).

Editors: Amit Chakrabarti and Chaitanya Swamy; Article No. 55; pp. 55:1–55:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

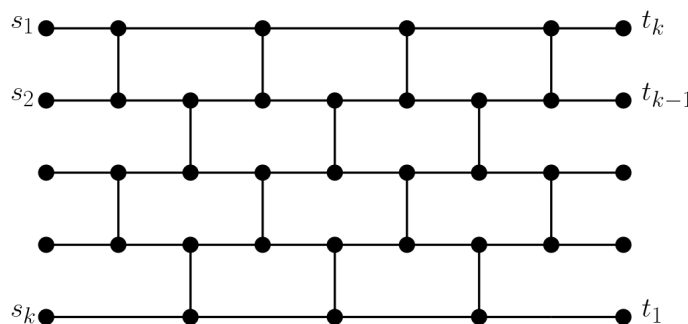
A natural dual to the maximum multicommodity flow problem is the *minimum multicut problem*. Given an edge-capacitated graph with  $k$  source-sink pairs, a multicut is a set of edges whose removal disconnects all the source-sink pairs, and the capacity (or value) of the cut is the sum of capacities of the edges in it. The value of any feasible multicommodity flow is at most the capacity of any feasible multicut. The ratio of the values of the minimum multicut and maximum multicommodity flow is called the *multiflow-multicut gap*. The ratio of the values of the minimum multicut and maximum multicommodity flow with congestion- $c$  is called the *multiflow-multicut gap* with congestion  $c$ . In case  $c$  is 1 or 2, we call it the integral or half-integral multiflow-multicut gap respectively. Minimum multicut is NP-Hard to compute, even in very restricted setting such as trees [11]. More precisely, it is known to be equivalent to the vertex cover problem in stars with unit weights [11], which implies that it is APX-Hard in series-parallel graphs. There is a rich literature on proving bounds on the multiflow-multicut gap. Perhaps the most famous of them is the max-flow min-cut theorem of Ford and Fulkerson [7], which states that the value of the minimum multicut is equal to the maximum (integral) flow when  $k = 1$ . Hu [12] extended the result of Ford and Fulkerson to show that the multiflow-multicut gap is 1 even when  $k = 2$ . Another tight example, closely related to our work, is the case where the graph obtained by adding an edge for each source-sink pair is series-parallel [5]. There are many other special cases where the multiflow-multicut gap is 1, for example when  $G$  is a path or a cycle, but in general it can be arbitrarily large. Garg et al. [10] proved a tight bound of  $\Theta(\log k)$  on the multiflow-multicut gap for any graph  $G$ . For  $K_r$  minor-free graphs, Tardos and Vazirani [16] used the decomposition theorem of Klein et al. [13] to prove a bound of  $O(r^3)$  on the multiflow-multicut gap. The integral multiflow-multicut gap can be  $\Omega(\sqrt{|V|})$ , even for planar graphs (see Figure 1).

Garg et al. [11] gave a tight bound of 2 on the integral multiflow-multicut gap when  $G$  is a tree. For graphs of treewidth  $r$ , Abraham et al. [1] gave a bound of  $O(r)$  on the multiflow-multicut gap by rounding a natural linear programming relaxation. Chekuri et al. [3] and Ene et al. [6] showed how to round a fractional multicommodity flow solution into an integral one by losing a factor of  $O(r^3)$ . Combining their results gives a bound of  $O(r^4)$  on the integral multiflow-multicut gap for graphs of treewidth  $r$ . Note that this implies a  $O(1)$  bound on the multiflow-multicut gap for treewidth 2 graphs. All the results mentioned above are algorithmic in nature and also imply an approximation algorithm for the (integral) multicommodity flow and multicut problems. Except for the case when  $G$  is a tree, all the results mentioned above are proved by rounding a natural linear programming relaxation to the problem.

We extend the augmenting paths framework of Ford and Fulkerson [7] to develop a primal-dual algorithm for multiflow and multicut for treewidth 2 graphs (see Theorem 2). It is a well known fact that the augmenting paths framework cannot be used for multicommodity flows in general. To the best of our knowledge, this is the first time augmenting paths framework has been adapted (in a non-trivial manner) for developing an algorithm for multicommodity flows and multicuts.

A simple topological obstruction of Garg et al. [11] shows that the integral multiflow-multicut gap is  $\Omega(r)$  for graphs with treewidth  $r$  (see Figure 1). Chekuri et al. [2] and Ene et al. [6] raised the question if the integrality gap of the natural linear programming for multicommodity flows is  $O(r)$  for graphs with treewidth  $r$ . We believe that the topological obstruction of Garg et al. [11] gives the best possible lower bound on the integral multiflow-multicut gap for graphs of treewidth  $r$ . To this end, we make the following conjecture, which strengthens the one stated by Ene et al. [6].

► **Conjecture 1.** *The integral multiflow-multicut gap for graphs with treewidth  $r$  is  $\Theta(r)$ .*



■ **Figure 1** In the above instance, all the edges have unit capacity and hence only one source-sink pair can be connected by edge-disjoint paths. We need at least  $k$  edges to disconnect all the source-sink pairs and hence the integral multiflow-multicut gap is at least  $\Omega(k)$ . The graph has a treewidth of  $\Theta(k)$ . This shows that the integral multiflow-multicut gap can be  $\Omega(r)$  for graphs with treewidth  $r$ .

It is known that the integrality gap for the linear programming relaxation for the multicut and the integer multicommodity flow for treewidth  $r$  graphs is  $\Omega(\log r)$  and  $\Omega(r)$  respectively. Hence, any algorithm which rounds the linear programming relaxation for multicommodity flow and multicut separately won't be able to resolve this conjecture. We believe that a primal-dual algorithm, which works with multicommodity flow and multicut simultaneously will lead to the resolution of this conjecture. We also believe that the techniques we develop in this paper makes important progress towards developing such an algorithm.

## 2 Our Contribution

As already noted in Section 1, results of Abraham et al. [1] and Ene et al. [6] imply an  $O(1)$  bound on the (integral) multiflow-multicut gap for treewidth 2 graphs, albeit with a large (unspecified) constant. Our main technical contribution is developing the first primal-dual algorithm for multiflow and multicut for treewidth 2 graphs. We prove the following approximate max-flow min-cut theorem for treewidth 2 graphs (see Section 3 for precise definitions):

► **Theorem 2.** *Let  $G$  be an undirected, (integer) edge capacitated treewidth 2 graph and  $\{(s_i, t_i)\}_{i=1}^k$  be the source-sink pairs. Then there exists an integral multicommodity flow of value  $f$  with congestion 4 and a multicut of value  $c$  such that  $c \leq 20f$ . Furthermore, there exists a primal-dual algorithm that computes such a flow and cut in time polynomial in size of the graph and the largest capacity. For unit capacity graphs, the algorithm runs in polynomial time.*

Our proof of Theorem 2 is completely combinatorial and does not require us to solve a linear program. It is based on the primal-dual framework. This leads to a more explicit algorithm and sheds further light on the structure of the multicuts and multicommodity flows in treewidth 2 graphs. All previous algorithms for computing multicommodity flows and multicuts were based on rounding the standard linear programming relaxation (except for some special cases, see Section 2.1). In many combinatorial optimization problems, algorithms based on the primal-dual schema give (near) optimal bounds on the approximation ratio, and we hope that further extensions of our approach will lead to tight results in the context of this problem as well. We would also like to point out that the bounds of Theorem 2 are the best known.

The broad outline of our proof follows the Ford-Fulkerson algorithm for computing the maximum  $(s, t)$ -flow and minimum  $(s, t)$ -cut in a graph. Since multiflows and multicuts are linear programming dual of each other, our algorithm can also be seen as a primal-dual algorithm. In each iteration, we increase the total flow by performing an augmenting step, ie. rerouting previously routed flow paths. This is done by generalizing the well known augmenting paths framework of Ford and Fulkerson [7] for single commodity flow. This generalization requires new ideas as it is well known that the augmenting paths framework can not be used directly for multicommodity flows. We then use the reachability graphs defined by the flows at the end of the algorithm to find a multicut for the instance, which can also be seen as generalisation of the cut-picking algorithm of Ford-Fulkerson [7].

The problem of computing minimum multicut can be formulated as an integer linear program. We can relax the integrality constraints to obtain a linear programming (LP) relaxation for multicut. The ratio between the optimum solution to the integer program and the LP relaxation is called the integrality gap of the relaxation. Theorem 2 also implies the same bound on the integrality gap of the integer programming relaxation for multicut in treewidth 2 graphs.

In Section 3, we formally define the problem statement and state the connection between treewidth 2 and series-parallel graphs. In Section 4, we give a quick overview of the augmenting paths algorithm of Ford-Fulkerson [7] for the single commodity case. In Section 5, we illustrate the basic ideas of our algorithm for a special case, ie. parallel-path graphs. In Section 7 and Section 8, we give the full algorithm for series-parallel graphs. We then go on to show how to pick a multicut in Section 9.

## 2.1 Other Related Work

Garg et al. [11] gave a primal-dual 2-approximation algorithm for finding an integral multicommodity flow and multicut for trees. Their result also implies a tight bound on the integral multiflow-multicut gap for trees. By combining the results of [8, 9], we can obtain a primal-dual algorithm for computing a multicut and integral flow when the graph obtained by adding an edge for every source-sink pair to  $G$  is planar. These also imply a tight half-integral multiflow-multicut gap of 2 and integral multiflow-multicut gap of 4 for such instances. To the best of our knowledge, there are no other completely combinatorial algorithm proving bounds on the multiflow-multicut gap for non-trivial class of instances.

The problem of finding maximum edge disjoint paths is NP-Hard, even in very restricted settings [14]. There is an  $O(\sqrt{n})$  approximation algorithm for finding maximum edge disjoint paths in general (undirected) graphs on  $n$  vertices [2]. This also matches the integrality gap of the natural linear programming relaxation for the problem [11]. Recently, Chuzhoy et al. showed that it is not possible to approximate the maximum edge disjoint paths problem better than  $2^{\Omega(\log^{1-\epsilon} n)}$  under reasonable hardness assumptions and it is an outstanding open problem to improve the  $O(\sqrt{n})$  approximation algorithm, even for planar graphs. If we relax the edge-disjointness condition and allow every edge to be used by up to  $c$  paths for some integer  $c \geq 2$ , then the problem is called the maximum edge disjoint paths with congestion  $c$ . A long line of impressive work culminated in a  $O(\text{polylog } n)$  approximation algorithm for general graphs [4] and a constant factor approximation algorithm for planar graphs [15] when a congestion of 2 is allowed. Both these results also imply the same bound on the integrality gap of the natural linear programming relaxation. The exact integrality gap of the maximum edge disjoint paths with congestion 2 for  $K_r$  minor-free graphs is still not known and is an interesting open question.

### 3 Preliminaries

Let  $G = (V, E)$  be a simple undirected graph with edge capacities  $c: E \rightarrow \mathbb{Z}_{\geq 0}$ ; we call this the supply graph. Let  $H = (V, F)$  be a simple graph each edge of which corresponds to a commodity and the endpoints of that edge are the source-sink of that commodity.  $H$  is the demand graph and its edges the demands.

Let  $\mathcal{P}$  be the set of all paths in  $G$  between a source and its corresponding sink. For a path  $P \in \mathcal{P}$ , we refer to  $f_P$  as the value of flow on  $P$ . A multiflow  $f: \mathcal{P} \rightarrow \mathbb{R}_{\geq 0}$  is feasible if for every edge  $e \in E$ , the total flow on all paths containing the edge,  $\sum_{P: e \in P} f_P$ , is at most the capacity of the edge,  $c(e)$ . We say that a multiflow has congestion  $l$  if the flow paths are allowed to use an edge up to  $l$  times its capacity, ie.  $\sum_{P: e \in P} f_P \leq l \cdot c(e)$ . If the value of flow on every path is an integer (resp. half-integer), then the flow is called an integral (resp. half-integral) multiflow.

A maximum multiflow is a feasible flow  $f$  which maximises  $\sum_{P \in \mathcal{P}} f_P$ . A multicut is a set of edges  $E' \subseteq E$  such that every  $P \in \mathcal{P}$  contains at least one edge in  $E'$ . Equivalently, a multicut is a set of edges whose removal disconnects every source-sink pair. Since a multicut contains an edge of every path in  $\mathcal{P}$ , the value of any feasible multicut is at least the value of any feasible multiflow. The ratio of the minimum multicut to the maximum (integral/half-integral) multiflow is called the (integral/half-integral) multiflow-multicut gap.

A cut  $S \subseteq V$  is a partition of the vertex set  $(S, V \setminus S)$ . Let  $\delta_E(S)$  denote the edges in  $E$  with exactly one endpoint in  $S$ . For a subset  $E' \subseteq E$  let  $c(E')$  be the total capacity of edges in  $E'$ . Let  $\delta_{\min}(u, v, G)$  denote the minimum value cut between  $u$  and  $v$  in  $G$ .

**Series-Parallel Graphs.** We will mostly focus on 2-terminal series-parallel graphs as the problem in treewidth-2 graphs can be easily converted to one in 2-terminal series-parallel graphs (see Proposition 3). From now on, we omit 2-terminal series-parallel graphs as simply series-parallel graphs. We will use a well known recursive definition of series-parallel graphs. A series-parallel graph has two distinguished vertices (also called the **merge vertices**)  $u, v$ . An edge is a series-parallel graph with its endpoints as the two merge vertices. Starting from an edge, any series-parallel graph can be constructed by two operations: parallel and series composition. Given two series-parallel graphs  $G_1, G_2$  with merge vertices  $(u_1, v_1), (u_2, v_2)$ , a parallel composition  $G_p$  of  $G_1, G_2$  is constructed by setting  $u = u_1 = u_2, v = v_1 = v_2$  and  $(u, v)$  as the merge vertices. Given two series-parallel graphs  $G_1, G_2$  with merge vertices  $(u_1, v_1), (u_2, v_2)$ , a series composition  $G_s$  of  $G_1, G_2$  is constructed by setting  $v_1 = u_2$  and  $(u_1, v_2)$  as the merge vertices. See Fig. 6 for an illustration. Consider  $k \geq 2$  simple node disjoint paths  $P_1, P_2, \dots, P_k$  between two vertices  $u, v$ . We call such a graph a *parallel-path graph*. In other words, parallel-path graphs have two distinguished vertices  $u$  and  $v$  and consist of internally vertex-disjoint  $u$ - $v$  paths.

**Series-Parallel Tree Decomposition.** For a series-parallel graph  $G$ , we associate with it a tree-decomposition  $T(G)$ . This is the canonical tree-decomposition of a series-parallel graph and consists of either 2 or 3 vertices in each bag. The tree-decomposition  $T(G)$  can be defined recursively as follows: if  $G$  is just an edge  $\{u, v\}$  then  $T(G)$  consists of a single bag  $\{u, v\}$ ; if  $G$  is a parallel-composition of  $G_1, G_2, \dots, G_r$  with merge vertices  $u$  and  $v$ , then  $T(G)$  is obtained by taking the bag  $R = \{u, v\}$  as the root and adding edges from  $R$  to the root of each of  $T(G_1), T(G_2), \dots, T(G_r)$ ; if  $G$  is a series composition of  $G_1, G_2$  with merge vertices  $u, v$  and the common merge vertex of  $G_1$  and  $G_2$  being  $w$ , then  $T(G)$  is obtained by taking the bag  $R = \{u, v, w\}$  as the root and adding edges from  $R$  to the root of each of

$T(G_1)$  and  $T(G_2)$ . We will use  $T$  throughout to denote the series-parallel tree-decomposition of the input series-parallel graph  $G$ , and for a node  $X$  of  $T$ , we use  $T_X$  to denote the sub-tree of  $T$  rooted at  $X$ . Also, we use  $G_X$  to denote the graph induced in  $G$  by the union of vertices in all the nodes in  $T_X$ .

We will work with series-parallel graphs in the paper. But the results apply also to treewidth-2 graphs because of the following proposition.

► **Proposition 3.** *Given an edge-capacitated treewidth-2 graph  $G$  and source-sink pairs  $T$ , one can in polynomial time find a series-parallel graph  $H \supseteq G$  such that any multicommodity flow with congestion  $g$  in  $H$  with respect to  $T$  is a multicommodity flow with congestion  $g$  in  $G$  with respect to  $T$ , and any multicut of  $H$  with respect to  $T$  is a multicut of  $G$  with respect to  $T$  having the same capacity.*

**Proof.** It is well known that every treewidth-2 graph is the sub-graph of a (2-terminal) series-parallel graph and such a super-graph that is (2-terminal) series-parallel can be found in polynomial time. We add the extra edges to make the graph series-parallel and set their capacities to 0. It is easy to see that then the proposition follows. ◀

For the sake of presentation, we make the following simplifying assumption. Let  $v \in V$  be a vertex and suppose that  $k$  source-sink pairs are incident on  $v$ . Then we add  $k$  edges  $(v, v_1), (v, v_2), \dots, (v, v_k)$  to  $G$  and set the capacity of each  $(v, v_i)$  to be equal to a large number, say  $\sum_{e \in E} c_e$ . If a source-sink pair  $(v, t)$  is incident on  $v$ , we replace it by  $(v_i, t)$ , such that each  $v_i$  has exactly one source-sink pair incident on it. We repeat this process for each vertex in the graph and let  $U$  be the set of new vertices introduced by this operation. Now every source-sink pair is incident on vertices in  $U$  and any vertex has at most one source-sink pair incident on it. Furthermore, there is one to one correspondence between any feasible multiflow and multicut with value at most  $\sum_{e \in E} c_e$  in the original and the modified graph. Hence, from now on we assume that exactly one source-sink edge is incident on any vertex of  $G$ .

## 4 Ford-Fulkerson Algorithm for Single Source

We heavily use the augmenting paths framework of Ford-Fulkerson [7] to design our algorithm. We give a brief overview of their algorithm here. Given a source vertex  $s$  and a set of sink vertices  $T = \{t_1, t_2, \dots, t_m\}$ , we wish to find the maximum amount of flow that can be routed from  $s$  to vertices in  $T$ . It is convenient to work with a directed network  $N = (V, E')$ , where each edge  $(u, v) \in E$  is replaced by two directed edges (arcs)  $(u, v)$  and  $(v, u)$  in  $N$ . The capacity of each of the arcs is equal to the capacity of the corresponding original edge. All the flow paths are directed from  $s$  to  $T$  in  $N$ . One can show that if a flow of value  $f$  can be routed in  $N$ , then a flow of value  $f$  can be routed in  $G$  as well. This allows us to work with  $N$  instead of  $G$ .

Let  $F$  be a set of flow paths directed from  $s$  to  $T$  in  $N$  and  $f(e)$  be the flow through arc  $e$  in  $F$ . We define the residual network with respect to  $F$ ,  $N_F = (V, E')$ , as follows: if  $f(u, v) \geq f(v, u)$ , then we set the capacity of  $(u, v)$  to  $c_{uv} - f(u, v) + f(v, u)$  and the capacity of the arc  $(v, u)$  to  $c_{uv} + f(u, v) - f(v, u)$  in  $N_F$ . Note that when  $f$  is empty, then the capacity of the forward and the backward arcs is equal to the capacity of the original edge in  $G$ .

The algorithm works in iterations. In each iteration, we increase the amount of flow from  $s$  to  $T$  by 1. At the beginning of each iteration, we find the set of reachable vertices  $R^F$  in the residual network  $N^F$  with respect to the current flow  $F$ . If there exists a  $t_i \in R_F$ , then we augment a unit of flow along a path from  $s$  to  $t_i$  in  $N_F$ . We update our residual graph as

described above and repeat the procedure until some vertex of  $T$  is reachable from  $s$  in the residual graph. We stop when none of the vertices of  $T$  are reachable from  $s$  in the residual graph. If the algorithm terminates after  $f$  iterations, then there exist  $f$  flow paths in the original graph  $G$  from  $s$  to  $T$ . In fact such flow paths can be computed directly from the final residual graph in polynomial time.

Let  $S$  be the set of reachable vertices in the residual network at the termination of the algorithm and  $f$  be the total number of flow paths routed. Ford-Fulkerson [7] showed that  $\delta(S) = f$ , ie. the maximum amount of flow from  $s$  to  $T$  in  $G$  is equal to the minimum (total) capacity set of edges which disconnect  $s$  from  $T$ . This is also known as the max-flow min-cut theorem for single-commodity flow.

**Residual Graph for Multicommodity Flow.** We can analogously define a residual network  $N_F$  of a graph  $G$  with respect to any (directed) flow  $F$ , and not just the single commodity flow. From now on, we will refer to  $N_F$  as the residual network of  $G$  with a current (directed) flow  $F$ . We will use  $f^-(v)$  and  $f^+(v)$  to denote the net incoming and outgoing flow incident at the vertex  $v$ .

## 5 Algorithm for Parallel-Path Graphs

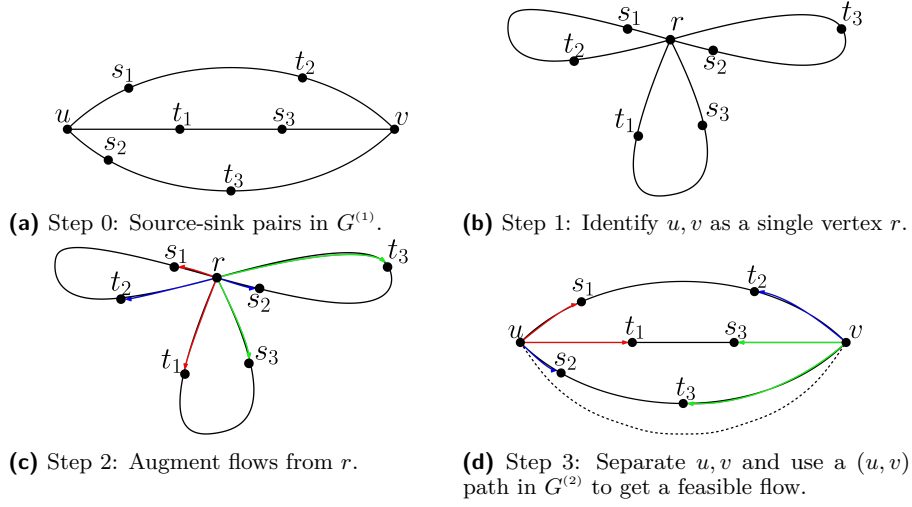
To illustrate the basic ideas of our approach, we first describe the algorithm for parallel-path graphs. Let  $G$  be a parallel-path graph and  $(u, v)$  be its merge vertices. We make a further simplifying assumption that all the source-sink pairs lie on different paths of  $G$ . This implies that all the source-sink paths contain either  $u$  or  $v$ . Let  $p$  be the maximum amount of flow that can be routed between  $u$  and  $v$  in  $G$ .

Our algorithm works with four copies of  $G$ , i.e.  $G^{(1)} = G^{(2)} = G^{(3)} = G^{(4)} = G$  each with the same capacities as  $G$ . Our flow paths at the end will lie in the union of the four copies. The capacity constraints on the edges will be satisfied within each copy. Thus, we will have a flow with congestion at most 4. We use the augmenting paths framework of Ford and Fulkerson to route flow in  $G^{(1)}$ . As it is well known, the augmenting paths framework can lead to infeasible flows when applied to a multicommodity setting. We carefully use the edges in  $G^{(2)}, G^{(3)}, G^{(4)}$  to *correct* the infeasible flows routed in  $G^{(1)}$ .

In  $G^{(1)}$ , we identify  $u, v$  as a single vertex and use the Ford-Fulkerson algorithm to construct a flow and cut as follows: let  $r$  be the vertex formed by identifying  $u, v$ . Observe that  $r$  is a cut-vertex and all the source-sink paths go through  $r$ . We think of a path between an  $s_i - t_i$  pair as the union of two (directed) paths: one from  $r$  to  $s_i$  and the other from  $r$  to  $t_i$ . To send  $f$  units of flow between an  $s_i - t_i$  pair, we first send a flow of value  $f$  from  $r$  to  $s_i$  and then another flow of value  $f$  from  $r$  to  $t_i$ . We call each of these as a **half-flow-path** of the flow between  $s_i$  and  $t_i$ . Note that all the half-flow-paths are directed away from  $r$ . Since every flow path is rooted at  $r$ , we treat it as the common source and use the augmenting paths algorithm of Ford-Fulkerson (see Figure 2). We use this process iteratively to route more flow between the source-sink pairs and distinguish between two cases:

**Case 1.** Suppose the algorithm terminates with a total flow of  $f < p$  (recall that  $p$  is the maximum  $u-v$  flow). Let  $S$  be the set of all the reachable vertices from  $r$  at the end of the algorithm. If there exists an  $i$  such that  $s_i, t_i \in S$ , then we would have been able to send more flow from  $r$  to  $s_i$  and  $r$  to  $t_i$ . Note that an  $r - s_i$  path does not overlap with an  $r - t_i$  path since  $s_i$  and  $t_i$  are assumed to be in different paths of the parallel-path graph  $G$ .





■ **Figure 2** Routing flow in a parallel-path graph.

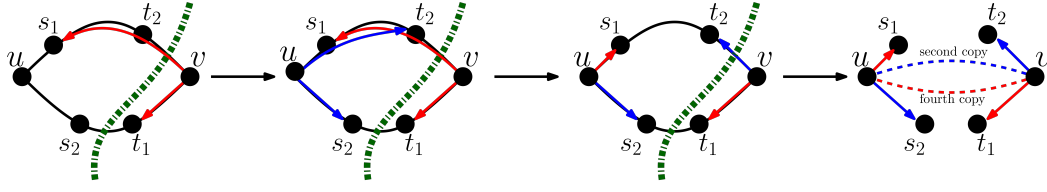
Since it is not possible to send any additional flow between the source-sink pairs, it must be true that  $S$  does not contain at least one of  $s_i, t_i$ , and hence the edges  $\delta(S)$  form a feasible multicut for this instance. Since  $r$  was formed by combining  $u, v$ , we may not have a feasible flow of value  $f$ , i.e. for a source-sink pair, one half-flow-path may be routed from  $u$  while the other one is routed from  $v$ . To convert this into a feasible flow, we use (at most)  $f$  units of  $u$ - $v$  flow in  $G^{(2)}$ . This results in a flow with congestion 2. Since every half-flow path uses at most one edge of the cut  $\delta(S)$ , we have that the value of the multicut is at most 2 times the total flow routed (with congestion 2).

**Case 2.** Suppose at some point in the algorithm, that the total flow routed (in  $G^{(1)}$ ) becomes exactly  $p$ . Since the maximum  $u$ - $v$  flow in  $G$  is  $p$ , there exists a set of edges, say  $C$ , of value  $p$  whose removal separates  $u$  and  $v$  in  $G^{(1)}$  (by the max flow-min cut theorem [7]). In this case, we pick a set of cut edges  $C$  with total value  $p$  in  $G^{(1)}$ . Let  $G_1^{(1)}, G_2^{(1)}$  be the graphs formed after removing the edges in  $C$  and let  $u \in G_1^{(1)}, v \in G_2^{(1)}$ .

Now, let us re-split  $r$  into  $u$  and  $v$  as it was. Each of the half-flow-paths are now rooted at either  $u$  or  $v$ . If both  $G_1^{(1)}$ , and  $G_2^{(1)}$  do not contain any source-sink pairs within them, we terminate. If there are source-sink pairs that are not separated by the removal of  $C$ , we augment flow from  $u$  in  $G_1^{(1)}$  and from  $v$  in  $G_2^{(1)}$  to increase our total flow. To do this, we use the augmenting paths algorithm with  $u$  (resp.  $v$ ) as the single source for  $G_1^{(1)}$  (resp.  $G_2^{(1)}$ ). Note that  $G_1^{(1)}$  (resp.  $G_2^{(1)}$ ) may contain flow edges of half-flows rooted at  $v$  (resp.  $u$ ). In the residual network, we orient a flow-edge in the opposite direction to the flow, irrespective of where the flow is rooted.

Since  $G_1^{(1)}$  possibly contains parts of half-flow-paths rooted at  $v$ , some of the half-flow-paths for source-sink pairs routed in  $G_1^{(1)}$  (after removing  $C$ ) may also be mismatched after augmentation (see Fig 3), i.e. one of them is rooted at  $u$  and the other is rooted at  $v$ , even though both were routed from the single source  $u$  in  $G_1^{(1)}$ . The same also can happen for  $G_2^{(1)}$ .

Let  $M$  be the set of pairs of mismatched half-flow-paths that were routed after removing the edges of  $C$ . In any pair of mismatched half-flow-paths in  $M$ , at least one of them uses an edge of  $C$ . Hence, total number of mismatched half-flow-paths in  $M$  is at most  $p$ . We use the  $p$   $u$ - $v$  flow paths in  $G^{(4)}$  to *correct* them, i.e., we obtain a complete flow path between  $s_i$  and  $t_i$  by using the two half-flow paths (ignoring direction) in  $G^{(1)}$  and a path from  $u$  to  $v$  in  $G^{(4)}$ .



**Figure 3** In the first picture (from left), we route a unit of flow from  $v$  to  $s_1$  and  $v$  to  $t_1$  and also pick a  $(u, v)$  cut (in green). This creates two connected components, one containing  $u$  and the other containing  $v$ . Observe that a part of the half-flow path from  $v$  to  $s_1$  is also present in the component containing  $u$ . In the second picture, we augment a unit of flow from  $u$  to  $s_2$  and  $u$  to  $t_2$ . This results in flow paths as shown in third figure, i.e.  $u$  to  $s_1, s_2$  and  $v$  to  $t_1, t_2$ . Since any mismatched flow-path routed after picking the (green) cut has to cross an edge of the cut, they can be at most its capacity. As shown in last figure, we use one  $(u, v)$  flow path in the second copy to correct  $(s_1, t_1)$  flow and another  $(u, v)$  path in the fourth copy to correct  $(s_2, t_2)$  flow.

Similarly, we correct the  $p$  units of flow routed before deleting  $C$  by using at most  $p$   $u$ - $v$  flow paths in  $G^{(2)}$ . After these corrections we have as much resultant flow between the terminal pairs as the number of half-flow pairs routed. Note that we did not use  $G^{(3)}$  yet, but we will need it for routing in the general case (see Section 7). Hence, we obtain a flow of congestion 3 in this case.

Let  $S_1$  (resp.  $S_2$ ) be the set of reachable vertices from  $u$  in  $G_1^{(1)}$  (resp. from  $v$  in  $G_2^{(1)}$ ) at the end of the algorithm (i.e. when we are not able to send any more flow in  $G_1^{(1)}$  and  $G_2^{(1)}$ ). We pick  $C \cup \delta(S_1) \cup \delta(S_2)$  as our multicut. It is straightforward from construction that this is indeed a multicut. Hence the value of the multicut is  $p + |\delta(S_1)| + |\delta(S_2)|$ . Note that  $|\delta(S_1)| + |\delta(S_2)|$  is at most the total number of half-flows routed as each edge in  $\delta(S_1)$  (resp.  $\delta(S_2)$ ) is saturated with flow going outside of  $S_1$  (resp.  $S_2$ ). Using the fact that  $p$  is at most the total number of half-flow pairs routed, we have that the value of the multicut is at most 3 times the total flow. It is easy to see that the run time of the above algorithm is similar to that of the Ford-Fulkerson algorithm, and hence we have the following theorem.

► **Theorem 4.** *Given an edge-capacitated parallel-path graph and source-sink pairs such that none of the source-sink pairs lie on one of the parallel paths, we can find an integral flow of value  $f$  with congestion 3, and a multicut of value at most  $3f$  in time polynomial in size of the graph and the largest capacity. For unit capacity graphs, the algorithm runs in polynomial time.*

## 6 Augmenting External Flows into a Parallel-Path Graph

We showed in the previous section how to successfully augment multicommodity flows in a parallel-path graph  $H$  (with no terminal pairs on a path). Now, suppose  $H$  occurs as a building block of a series-parallel composition during the construction of a (larger) series-parallel graph. In our algorithm for series-parallel graphs, it is crucial that we are able to augment flows coming from vertices outside  $H$  into  $H$  through its merge vertices. Moreover, this has to be done in a way that the flows routed already inside do not get destroyed. We show in this section that a careful use of copies of the graph allows us to extend the augmenting paths framework of Ford and Fulkerson [7] to augment external flows into a parallel-path graph.

We first process all the source-sink pairs which are contained inside  $H$  using the algorithm described in Section 5. If a cut separating  $u$  and  $v$  in  $H$  is picked by the algorithm, then as shown in Section 5, we may safely continue to augment flow coming into  $H^{(1)}$  by using

the augmenting paths algorithm. This is because the maximum number of mismatched half-flow-paths arising after a  $u$ - $v$  cut is picked is at most the value of the minimum  $(u, v)$  cut and can be corrected by using one of the  $u$ - $v$  flow paths in  $H^{(4)}$ . Also, mismatched half-flow-paths that were routed before the  $(u, v)$  cut was picked can be corrected using  $u$ - $v$  flow paths in  $H^{(2)}$ .

We next show that we can safely continue to augment flows into  $H^{(1)}$  from outside (i.e. for source-sink pairs not contained inside  $H$ ) even in the other case i.e. when a  $(u, v)$ -cut has not been picked by the algorithm. As before, let  $p$  denote the value of minimum  $u$ - $v$  cut in  $H$ . Suppose that a total flow less than  $p$  is routed by the algorithm. This implies that no  $(u, v)$ -cut is picked. Let the number of half-flow-paths incident at  $u, v$  be  $f_u, f_v$  respectively and the total flow be  $f = f_u + f_v$ . Let  $H_w^{(1)}$  be the graph formed by adding a vertex  $w$  to  $H^{(1)}$  and connecting it to  $u$  and  $v$  with edges of capacity  $f_{wu}$  and  $f_{wv}$  respectively. Suppose we are able to augment  $f_w = f_{wu} + f_{wv}$  units of flow in  $H_w^{(1)}$  from  $w$  in the residual graph (note that in the residual graph, all the flow paths in  $H^{(1)}$  are rooted and directed away from  $u$  and  $v$ ). Then we show how to use the additional  $2f \leq 2p$  edge-disjoint paths from  $u$  to  $v$ , in  $H^{(2)}$  and  $H^{(3)}$  ( $f$  flow-paths in each) to reconstruct feasible flow paths, i.e. for each flow augmentation that happened from  $w$  to a vertex  $y$ , we produce a flow path from  $w$  to  $y$ , in addition to the flow paths that were already routed inside  $H$ .

► **Lemma 5.** *All flow paths (old and new) in  $H_w^{(1)}$  can be reconstructed by using at most  $2f$   $u$ - $v$  paths.*

**Proof.** To prove the lemma, we need the following crucial observation: if we augment a unit flow from the vertex  $w$  to  $x$  in  $H_w^{(1)}$ , then the amount of outgoing and incoming flow after augmentation remains unchanged for every vertex on the augmenting path except for  $w$  and  $x$ . The net flow (i.e. the amount of outgoing flow minus the incoming flow) of  $w$  increases by 1 while that of  $x$  decreases by 1. Let  $(s_1, t_1), \dots, (s_q, t_q)$  be the  $q$  source-sink pairs which were routed inside  $H^{(1)}$  and  $h_1, h_2, \dots, h_q$  be the amount of flow routed for each one of them. Let  $w_1, w_2, \dots, w_l$  be the vertices to which we augmented flow from  $w$  in  $H_w^{(1)}$  and  $d_1, d_2, \dots, d_l$  be the flow routed for each of them. Let  $O = \{s_1, t_1, \dots, s_q, t_q\}$  and  $N = \{w_1, w_2, \dots, w_l\}$ . Before the augmentations from  $w$ , the net flow out of  $u$  and  $v$  in  $H^{(1)}$  is  $f_u$  and  $f_v$  respectively and the net flow out of each vertex in  $O$  is  $-h_i$ . After the augmentations, the net flow out of  $u$  and  $v$  within  $H^{(1)}$  (i.e. without taking into account flow on edges  $wu$  and  $wv$ ) are  $f_u + f_{wu}$  and  $f_v + f_{wv}$  respectively, while that of vertices in  $O, N$  are  $-h_i, -d_j$  respectively.

Since  $u$  and  $v$  have positive net flow in  $H^{(1)}$ , vertices in  $O \cup N$  have negative net flow and rest of the vertices have zero net flow, we must have flow paths (with suitable flow value) from  $u, v$  to all the vertices in  $O \cup N$ . We first correct the flow paths corresponding to the source-sink pairs  $(s_1, t_1), \dots, (s_q, t_q)$  by using  $\min(f_u, f_v) \leq f$  edge disjoint paths between  $u$  and  $v$  in  $H^{(2)}$  and  $H^{(3)}$ . If exactly  $f_{wu}$  (resp.  $f_{wv}$ ) edge disjoint paths starting at  $u$  (resp.  $v$ ) terminate at vertices in  $N$ , then we already have a feasible flow. If  $f_{wu} + g$  (resp.  $f_{wv} - g$ ) units of flow incident at  $u$  (resp.  $v$ ) terminate at vertices in  $N$ , then we use  $g$  flow paths from  $u$  to  $v$  to correct the flow paths originating at  $w$ . We now argue that  $|g| \leq \max(f_u, f_v)$ . This follows from the fact that  $f_u - g$  (resp.  $f_v + g$ ) paths incident at  $u$  (resp.  $v$ ) must terminate in  $O$ , hence  $|g| \leq f_u$  or  $|g| \leq f_v$  which gives  $|g| \leq \max(f_u, f_v)$ . Hence total number of paths between  $u$  and  $v$  used to correct the flows is at most  $\max(f_u, f_v) + \min(f_u, f_v) = f_u + f_v = 2f \leq 2p$ . ◀

We will build on the intuition developed in this section to give a routing algorithm for the general case in the next section.

## 7 Routing Algorithm for Series-Parallel Graphs

Building upon the ideas developed in the previous sections, we now describe the full algorithm for routing flows in series-parallel graphs. We will also pick some cut-edges during the routing here, but they will not form the whole multicut; the algorithm for picking the complete multicut will be presented later in Section 9. Our routing algorithm is recursive using the recursive construction of series-parallel graphs through series and parallel compositions.

Let  $G$  be the input series-parallel graph and let  $u$  and  $v$  be its merge vertices. We construct four copies of  $G$  denoted by  $G^{(1)}, G^{(2)}, G^{(3)}$  and  $G^{(4)}$ , each with the same capacities as  $G$ . The algorithm outputs the following: a set of (directed) flow paths  $F$  in  $G^{(1)}$ , a set of cut-edges  $C$  (not necessarily a multicut), and two numbers  $(l^{(2)}, l^{(4)})$ . During the algorithm we will *reserve* some flow-paths between the merge vertices  $u$  and  $v$  in  $G^{(2)}, G^{(3)}$ , and  $G^{(4)}$  for *flow correction*. The *reserved flow* will be used in the flow-correction phase in Section 8 to correct the *mismatched flows* in  $G^{(1)}$ . The number  $l^{(2)}$  gives the number of flow paths available in each of  $G^{(2)}$  and  $G^{(3)}$  between  $u$  and  $v$  for flow-correction in the future, after we have *reserved* the flow-paths for correcting the flows routed so far. The number  $l^{(4)}$  gives the same for  $G^{(4)}$ . In a sense,  $l^{(2)}, l^{(3)}, l^{(4)}$  are the *residual flow-correcting capacities* that  $G$  passes on up to its parent in the recursion call.

We also maintain a global tuple  $D = \{d_1, d_2, \dots, d_k\}$  where  $d_i$  denote the amount of flow routed for terminal pairs  $(s_i, t_i)$ . We will maintain throughout the algorithm that  $d_i = f^-(s_i) = f^-(t_i)$ , where  $f^-(x)$  denote the incoming flow to  $x$  in  $F$ . Whenever we augment a new unit of flow for an  $s_i$ - $t_i$  pair, we assume that  $d_i$  increases by one, even if not mentioned explicitly.

We first describe the **base case**, i.e. if  $G$  is an edge  $(u, v)$  with capacity  $c(u, v)$ . If  $(u, v)$  do not form a source-sink pair, then the algorithm returns an empty flow,  $l^{(2)} = l^{(4)} = c(e)$  and  $C = \emptyset$ . If  $(u, v)$  is a source-sink pair i.e. if  $(u, v) = (s_i, t_i)$ , we send  $c(e)$  units of (directed) flow from  $u$  to  $v$  in  $G^{(1)}$ , reserve  $c(e)$  amount of flow-paths from  $u$  to  $v$  in each of  $G^{(2)}, G^{(3)}, G^{(4)}$  and return  $l^{(2)} = l^{(4)} = 0$  and  $C = \{(u, v)\}$ .

Now, we go to the recursion step. Let  $G$  be composed of  $G_1$  and  $G_2$  in series or parallel. Let  $u_1, v_1$  be the merge vertices of  $G_1$  and  $u_2, v_2$  be the merge vertices of  $G_2$ . We first run the routing algorithm on  $G_1$  and  $G_2$  separately. For  $i = 1, 2$ , let  $(F_i, l_i^{(2)}, l_i^{(4)}, C_i)$  be the output of the algorithm. Depending on whether  $G_1$  and  $G_2$  are joined in series or parallel, the algorithm now branches out into two cases.

### 7.1 Parallel Case

Recall that in the parallel case,  $G$  is obtained by connecting  $G_1$  and  $G_2$  in parallel i.e. by setting  $u = u_1 = u_2$ , and  $v = v_1 = v_2$ . Before routing flow, we remove all the edges in  $C_1$  and  $C_2$  from  $G^{(1)}$ . Our algorithm here is similar to the parallel-path case in Section 5. We say that a terminal pair is *newly connected* if one of the terminals is in  $G_1$  and the other is in  $G_2$ . If no source-sink pairs get newly connected due to the parallel combination, we simply return  $F_1 \cup F_2$ ,  $l^{(i)} = l_1^{(i)} + l_2^{(i)}$  for  $i = 2, 4$  and  $C = C_1 \cup C_2$ .

Otherwise, some source-sink pairs get newly connected. All paths between the newly connected source-sink pairs have to contain either  $u$  or  $v$ . Let  $s$  be the vertex obtained by identifying  $u$  and  $v$  as a single vertex. We initialize the flow  $F$  to be  $F_1 \cup F_2$ . Let  $R_s$  be the set of reachable vertices from  $s$  in the residual graph of  $G^{(1)}$  with respect to the flow  $F$ . We say that a newly connected source-sink pair  $(s_j, t_j)$  is **reachable** from  $s$  if both  $s_j \in R_s$  &  $t_j \in R_s$ .

If there is such a reachable newly connected source-sink pair then we augment in  $F$ , one unit of flow each to  $s_j$  and  $t_j$  from the vertex  $s$  and set  $d_j = d_j + 1$ . Since  $s_j \in G_1$  and  $t_j \in G_2$ , the two augmenting paths from the vertex  $s$  to  $s_j$  and  $t_j$  are vertex disjoint except

at  $s$ . Hence we can augment along both the paths simultaneously. However, note that this does not mean we can directly construct a flow path between  $s_j$  and  $t_j$  by combining both of these half-flows (ignoring directions), as the half-flow path to  $s_j$  may begin at  $u$  while the half-flow path to  $t_j$  may begin at  $v$ . Later in the correction step in Section 8, we will use a  $(u, v)$  path in either  $G^{(2)}$ ,  $G^{(3)}$ , or  $G^{(4)}$  to obtain a feasible flow.

As in the parallel-path case, we repeat the above routing procedure until one of the following happens: either there are no more reachable source-sink pairs from  $s$ , or we have routed  $l_1^{(2)} + l_2^{(2)}$  many units. Let  $f$  denote the number of half-flow pairs routed after connecting  $G_1, G_2$  in parallel.

1. In case 1, i.e. if the routing terminates with  $f < l_1^{(2)} + l_2^{(2)}$ , then we reserve  $f$  out of the available  $l_1^{(2)} + l_2^{(2)}$   $u$ - $v$  paths for flow correction in each of  $G^{(2)}$  and  $G^{(3)}$ . We return the flow  $F$ , cut edges  $C = C_1 \cup C_2$ , and numbers  $l^{(2)} = l_1^{(2)} + l_2^{(2)} - f$ , and  $l^{(4)} = l_1^{(4)} + l_2^{(4)}$ .
2. In case 2, i.e. if  $f = l_1^{(2)} + l_2^{(2)}$ , then we pick a min-cut separating  $u$  and  $v$  in  $G^{(1)}$ , say  $C_s$ . We set  $C = C_1 \cup C_2 \cup C_s$ . Let  $G_u^{(1)}$  and  $G_v^{(1)}$  be the two graphs formed after removing the edges of  $C_s$  from  $G^{(1)}$ . Even after removing the cut edges in  $C_s$ , there might be source-sink pairs that are reachable from  $u$  in  $G_u^{(1)}$  or  $v$  in  $G_v^{(1)}$ . We augment  $F$  by routing from  $u$  in  $G_u^{(1)}$  (resp. from  $v$  in  $G_v^{(1)}$ ) to the reachable source-sink pairs and update  $D$  accordingly. We do this until no source-sink pairs are reachable from  $u$  in  $G_u$  and  $v$  in  $G_v$ . We reserve  $l_1^{(2)} + l_2^{(2)}$   $u$ - $v$  paths in each of  $G^{(2)}$  and  $G^{(3)}$  and  $l_1^{(4)} + l_2^{(4)}$   $u$ - $v$  paths in  $G^{(4)}$  for flow corrections and return  $l^{(2)} = l^{(4)} = 0$  along with the flow  $F$  and cut  $C = C_1 \cup C_2 \cup C_s$ .

## 7.2 Series Case

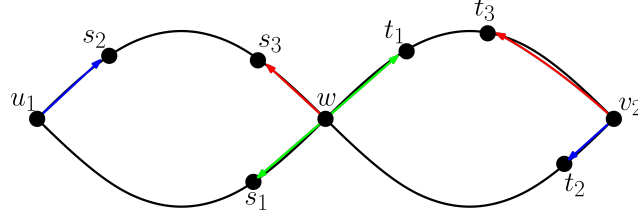
Recall that in the series case,  $G$  is obtained by connecting  $G_1$  and  $G_2$  in series, i.e. by identifying  $w = v_1 = u_2$ . Before routing flow, we remove all the edges in  $C_1$  and  $C_2$  from the first copy of  $G$ . To make the presentation simpler, w.l.o.g we assume that  $l_1^{(2)} \leq l_2^{(2)}$ .

If no new source-sink pairs get connected due to the series combination, we simply return  $F_1 \cup F_2$ ,  $l^{(i)} = \min\{l_1^{(i)}, l_2^{(i)}\}$  for  $i = 2, 4$  and  $C = C_1 \cup C_2$ .

Otherwise, some new source-sink pairs get connected. All paths between the newly connected source-sink pairs have to contain  $w$ . Nevertheless we route from any of  $u_1, w$  and  $v_2$  as below. We identify  $u_1, w$  and  $v_2$  into a super-source vertex, say  $s$ , and find a source-sink pair  $(s_j, t_j)$  such that both  $s_j$  and  $t_j$  are reachable from  $s$  in the residual graph of  $G^{(1)}$  with respect to flow  $F$ , which is initialized to  $F_1 \cup F_2$ . We call such source-sink pairs **reachable** from  $s$ . Note that if both are reachable then both can be routed simultaneously, as one of them lies in  $G_1$  and the other in  $G_2$ . We augment in  $F$  one unit of flow from  $s$  to  $s_j$  and from  $s$  to  $t_j$  and update  $D$  accordingly. However, note that this might not directly give us a flow path from  $s_j$  to  $t_j$ , as the half-flow path to  $s_j$  may begin at  $u_1$  while the half-flow path to  $t_j$  may begin at  $v_2$ . Later in the correction step, we will use a  $(u_1, v_2)$  path in  $G^{(2)}$ ,  $G^{(3)}$ , or  $G^{(4)}$  to obtain a feasible flow.

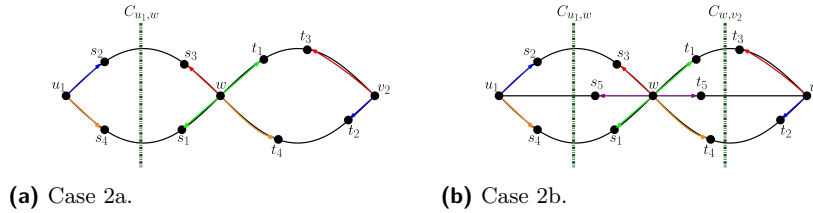
We keep augmenting as above until one of the following happens: either no more source-sink pairs are reachable from  $s$  or we have routed  $\min\{l_1^{(2)}, l_2^{(2)}\}$  units of flow. Let  $f$  denote the total source-sink flow routed after connecting  $G_1, G_2$  in series.

1. In case 1, i.e. if the routing terminates with  $f < \min\{l_1^{(2)}, l_2^{(2)}\}$ , then we reserve  $f$  flow paths between  $u_1$ - $v_2$  in  $G^{(2)}$  and  $G^{(3)}$  (note that these reserved flow-paths goes through  $w$  and in the flow-correction phase, we may use such a flow-path to correct a flow between  $u_1$  and  $w$ , or  $w$  and  $v_2$ , or  $u_1$  and  $v_2$ ). We return the flow  $F$ , the cut  $C = C_1 \cup C_2$ , and  $l^{(2)} = \min\{l_1^{(2)}, l_2^{(2)}\} - f$ , and  $l^{(4)} = \min\{l_1^{(4)}, l_2^{(4)}\}$ .



■ **Figure 4** On the left part  $(u_1, w)$ , we have  $l_1^{(2)} = 4$  and on the right part,  $(w, v_2)$  we have  $l_2^{(2)} = 5$ .

2. In case 2, i.e. if  $f = \min\{l_1^{(2)}, l_2^{(2)}\} = l_1^{(2)}$  (w.l.o.g), then we pick a min-cut separating  $u_1$  and  $w$  in  $G^{(1)}$ , say  $C_{u_1, w}$ . We set  $C = C_1 \cup C_2 \cup C_{u_1, w}$  and  $G_{u_1}^{(1)}$  and  $G_{w, v_2}^{(1)}$  be the two graphs formed after removing the edges of  $C$  from  $G^{(1)}$ . Let  $s'$  be the vertex formed by identifying  $w, v_2$  as a single vertex. Even after removing the cut edges in  $C_{u_1, w}$ , there might be source-sink pairs that are reachable from  $s'$  in  $G_{w, v_2}^{(1)}$ . We augment flow (in  $F$ ) from  $s'$  in  $G_{w, v_2}^{(1)}$  to the reachable source-sink pairs from  $s'$  until one of the following happens: either no more source-sink pairs are reachable from  $s'$  or we have augmented  $l_2^{(2)} - l_1^{(2)}$  units of such flow.
  - a. In case a), i.e. if no more terminal pairs are reachable from  $s'$  and  $f < l_2^{(2)}$  (here  $f$  is the total amount of flow augmented after connecting  $G_1$  and  $G_2$  in series), then we reserve  $f - l_1^{(2)}$  units of flow paths between  $w$  and  $v_2$  in  $G^{(2)}$  and  $G^{(3)}$ , reserve  $l_1^{(4)}$  flow-paths between  $u_1$  and  $w$  in  $G^{(4)}$ , and return the flow  $F$ ,  $l^{(2)} = l^{(4)} = 0$  and  $C = C_1 \cup C_2 \cup C_{u_1, w}$ .
  - b. In case b), i.e. if  $f = l_2^{(2)}$  (i.e.  $l_1^{(2)}$  units of flow was routed before deleting  $C_{u_1, w}$  and  $l_2^{(2)} - l_1^{(2)}$  units of flow afterwards), then we pick a min-cut separating  $w$  and  $v_2$  in  $G_{w, v_2}^{(1)}$ , say  $C_{w, v_2}$ . We set  $C = C_1 \cup C_2 \cup C_{u_1, w} \cup C_{w, v_2}$  and let  $G_w^{(1)}$  and  $G_{w, v_2}^{(1)}$  be the two graphs formed after removing the edges of  $C$  from  $G_{w, v_2}^{(1)}$ . Even after removing the cut edges in  $C_{w, v_2}$ , there might be source-sink pairs that are reachable from  $w$  in  $G_w^{(1)}$ . We augment flow (in  $F$ ) from  $w$  in  $G_w^{(1)}$  to the reachable source-sink pairs from  $w$ . We do this until no source-sink pairs are reachable from  $w$  in  $G_w^{(1)}$ . We reserve  $l_1^{(2)} + l_2^{(2)}$  amount of  $w - v_2$  flow-paths in  $G^{(2)}$  and  $G^{(3)}$ . We also reserve  $l_1^{(4)}$  amount of  $u_1 - w$  flow and  $l_2^{(4)}$  amount of  $w - v_2$  flow in  $G^{(2)}$ . We return  $F$ ,  $l^{(2)} = l^{(4)} = 0$ , and  $C = C_1 \cup C_2 \cup C_{u_1, w} \cup C_{w, v_2}$ .



(a) Case 2a.

(b) Case 2b.

## 8 Constructing Feasible Flows

Let  $D = \{d_1, d_2, \dots, d_k\}$  be the vector of all the source-sink flow values at the end of the algorithm. We will show that a feasible flow between the terminal pairs of value  $\sum_{i=1}^k d_i$  can be constructed using the second, third and fourth copy of  $G$ . First we note some of the properties of the routing algorithm, which will be helpful in proving further results.



Let  $G$  be a series-parallel graph with merge vertices  $(u, v)$ . Let  $(F, l^{(2)}, l^{(4)}, C)$  be the output of the algorithm on  $G$  and let  $f$  be the total flow (i.e. the number of half-flow pairs routed) in  $F$ . When  $G$  is not an edge we will assume that  $G$  is formed by the series or parallel composition of  $G_1$  and  $G_2$  with merge vertices  $(u_1, v_1)$  and  $(u_2, v_2)$  respectively. Let  $(F_i, l_i^{(2)}, l_i^{(4)}, C_i)$  denote the output of the algorithm for  $G_i$  for  $i = 1, 2$ .

► **Lemma 6.** *After  $G$  has been processed, there exists a  $(u, v)$ -cut of value at most  $l^{(4)}$  in  $G \setminus C$ .*

► **Lemma 7.**  $c(C) \leq 2f$ .

Recall that while routing in  $G$ , we reserved some flow paths between  $u$  and  $v$  for flow corrections. The next claim shows that the total value of reserved flow paths (across all iterations) is at most four times the total value of flow routed in  $G$ .

▷ **Claim 8.** The value of reserved flow paths in each of  $G^{(2)}$  and  $G^{(3)}$  is at most  $f$  and that in  $G^{(4)}$  is at most  $2f$ .

## 8.1 The Augmentation Property and Flow Correction

We now show that a feasible flow of value equal to the total augmented flow can be obtained by using the reserved flow paths, at each stage of the algorithm. To prove this result, we inductively maintain an invariant called as the **augmentation property**, specified below. Let  $G^*$  be the final graph and  $G$  be the graph obtained at an intermediate stage. Let  $(u, v)$  be the merge vertices of  $G$ . For giving the augmentation property, we distinguish between two cases, depending on whether a cut separating  $(u, v)$  has been picked by the algorithm so far. In both cases the augmentation property states that we can reconstruct all the source-sink flow paths that were augmented inside  $G$  (i.e. all the flow paths augmented inside  $G$  before its processing is finished), using only the flow paths reserved in the copies of  $G$ . In addition, to this, the property also states the following depending on the case.

- **Case 1.** No  $(u, v)$  cut has been picked by the algorithm so far: suppose a flow of  $f_1, f_2, \dots, f_k$  was augmented to (terminal) vertices  $t_1, t_2, \dots, t_k$  after the processing of  $G$  was finished (these are external flows that come from outside of  $G$ ). Furthermore, suppose that  $f_u$  and  $f_v$  units of flow was augmented from  $u$  and  $v$  respectively into  $G$  (by external flows) after the processing of  $G$  is finished, i.e.  $f_u + f_v = \sum_{i=1}^k f_i$ . Then the augmentation property states that we can additionally reconstruct these flow paths using only the reserved paths in copies of  $G$  such that: (i) exactly  $f_u$  (resp.  $f_v$ ) units of flow path emerge from  $u$  (resp.  $v$ ) (ii) there is exactly  $f_i$  units of incoming flow incident at each  $t_i$ . In other words, we reconstruct all flow paths corresponding to augmenting paths, except that they might originate from either  $u$  or  $v$  (there might have been a path originally augmented from  $u$  to  $t_i$ , but in the reconstructed paths the path to  $t_i$  might be from  $v$ ).
- **Case 2.** A  $(u, v)$  cut has been picked by the algorithm: let  $G_u, G_v$  be the two connected components of  $G$  (after deleting the cut edges) containing the vertices  $u, v$  respectively. Suppose a flow of value  $f_1, f_2, \dots, f_k$  was augmented into  $G_u$  (via  $u$ ) to (terminal) vertices  $t_1, t_2, \dots, t_k$  after the processing of  $G$  was finished. Then the augmentation property states that we can reconstruct feasible flow paths (in addition to the source-sink flow paths that were augmented inside  $G$  before its processing was finished) using only the reserved flows for  $G$ , such that there is a flow of value  $f_i$  from  $u$  into  $t_i$  for each  $i = 1, 2, \dots, k$ . The same holds true for  $G_v$  as well.



We show the following lemma by using induction on the structure of series-parallel graphs. This also implies that there exists a feasible flow of value  $\sum_{i=1}^k d_i$ .

► **Lemma 9.** *For any graph  $G$  obtained during an intermediate stage of the routing algorithm, the augmentation property holds.*

## 9 Picking a Multicut

Let  $C$  be the cut edges picked after the completion of routing phase for  $G$ . In this section, we assume that the edges of  $C$  have been removed from  $G$ . In addition to the cut edges  $C$ , we pick another set of edges  $Y$  such that  $C \cup Y$  is a feasible multicut for the given instance. We say that the edges in  $C$  were picked during the phase 1 of the algorithm. We now describe the phase 2 of the algorithm, where we pick the edges in  $Y$ . We start with all the vertices of  $G$  as unmarked and initialize the set  $Y$  as empty. We process the nodes of a tree-decomposition  $T$  of  $G$  (with treewidth 2) in a top-down manner, i.e. we process a node only after all its ancestors are processed. Let  $X$  be the current node we are processing. Recall that each node  $X$  corresponds to a series or a parallel combination of two subgraphs of  $G$  and it consists of union of the merge vertices of these two subgraphs. Let  $C_X$  be the set of reachable vertices in the residual graph of  $G_X$ , from  $X$ , just after the processing of  $X$  in phase 1 has been completed. Recall that the residual graph arises w.r.t to the current (directed) flow in the first copy of the graph. If all the vertices in  $X$  are already marked then do nothing. Let  $X'$  be the set of unmarked vertices in  $X$ . For any vertex  $x$ , let  $\text{Comp}_G(x)$  denote the connected component containing  $x$  in the current graph (i.e.  $G \setminus (Y \cup C)$ ). For each  $x \in X'$ , mark all the vertices in  $C_X \cap \text{Comp}_G(x)$ , add the edges in  $Y_X := \delta(C_X) \cap E(\text{Comp}_G(x))$  to  $Y$ , and delete those edges from  $G$ . Repeat this process until all the vertices of  $G$  have been marked. Then the union of  $Y$  and  $C$  is our required multicut.

► **Lemma 10.** *Let  $X'$  be a node of  $T$  and  $X$  be a node of  $T_{X'}$ . Then,  $C_{X'} \cap V(G_X) \subseteq C_X$ .*

**Proof.** Since any path in the residual graph from outside  $G_X$  has to enter through  $X$  and all edges in  $\delta(C_X)$  are directed inwards to  $C_X$  in the residual graph, the vertices in  $V(G_X) \setminus C_X$  can never become reachable from any vertex outside  $G_X$  in the residual graph. The lemma follows from this easily. ◀

► **Lemma 11.**  *$C \cup Y$  is a multicut of  $G$  for the given terminal-pairs.*

**Proof.** Suppose  $Y \cup C$  does not cut some terminal pair  $s, t$ . This means  $G - Y - C$  contains a path  $P$  between  $s$  and  $t$ . Let  $X$  be the bottom-most node in  $T$  such that  $G_X$  contains both  $s$  and  $t$ . Clearly  $P$  contains at least one vertex from  $X$ . Let this vertex be  $x$ . We branch into 2 cases depending on when  $x$  was marked in phase 2.

In Case 1, we suppose  $x$  was marked during the processing of  $X$ . Without loss of generality we can assume that the sub-path of  $P$  between  $x$  and  $s$  contains an edge of  $\delta_{G-C}(C_X)$ , say  $e$  (follows from phase 1 algorithm). Since  $e$  is in the same connected component as  $x$  in  $G - C - Y$ , and  $e \in C_X$ , we have that  $e$  would have been picked into  $Y$  during the processing of  $X$ , a contradiction.

In Case 2, we suppose  $x$  was marked before the processing of  $X$ . Let  $X'$  be the node during whose processing,  $x$  was marked. Clearly  $X$  is in  $T_{X'}$ . Thus, by Lemma 10, we have that  $C_{X'} \cap V(G_X) \subseteq C_X$ . Hence, without loss of generality we can assume that the sub-path of  $P$  between  $x$  and  $s$  contains an edge of  $\delta_{G-C}(C_{X'})$ , say  $e$ . Since  $e$  is in the same connected component as  $x$  in  $G - C - Y$ , and  $e \in C_{X'}$ , we have that  $e$  would have been picked into  $Y$  during the processing of  $X'$ . ◀

For a node  $X$  of  $T$ , let  $f(X)$  denote the number of half-flow paths introduced during the processing of  $X$  in phase 1. Since every flow path consists of two half flow paths, we have that total flow routed in phase 1,  $f = \sum_{X \in T} f(X)/2$ . For a node  $X$  of  $T$ , let  $r(X)$  denote the number of flow paths reserved between the vertices of  $X$  when  $T_X$  was being processed in phase 1. From Claim 8, it follows that  $\sum_{X \in T} r(X) \leq \frac{4}{2} \cdot \sum_{X \in T} f(X) \leq 2 \cdot \sum_{X \in T} f(X)$ .

Let  $M(X)$  denote the set of previously unmarked vertices that becomes marked during the processing of  $X$  in Phase 2. Let  $I(X)$  denote the set of nodes of  $T$  that have non-empty intersection with  $M(X)$ .

► **Lemma 12.** *For a node  $X$  of  $T$ , the total capacity of edges picked into the cut  $Y$  during the processing of  $X$  in Phase 2 is at most  $\sum_{X' \in I(X)} f(X') + \sum_{X' \in I(X)} r(X')$ .*

► **Lemma 13.** *For a node  $X'$  of  $T$ , the number of nodes  $X$  of  $T$  such that  $X' \in I(X)$  is at most 3.*

**Proof.** During the processing in Phase 2 of each  $X$  such that  $X' \in I(X)$ , at least one unmarked vertex in  $X'$  becomes marked. The lemma follows as there are at most 3 vertices in  $X$ . ◀

► **Lemma 14.**  *$|Y \cup C|$  is at most 20 times the amount of flow routed between the terminal pairs by our algorithm.*

**Proof.** Recall that  $\sum_{X \in T} r(X) \leq 2 \cdot \sum_{X \in T} f(X)$ . From Lemma 13 and Lemma 12, it follows that  $|Y|$  is at most  $3 \cdot (\sum_{X \in T} f(X) + 2 \cdot \sum_{X \in T} r(X)) \leq 9 \cdot \sum_{X \in T} f(X)$ . Hence, the total capacity of edges in  $Y$  is at most 18 times the total flow routed in phase 1. From Lemma 7, we have that  $|C|$  is at most twice the total flow routed by the phase 1 algorithm. Therefore, total capacity of edges in  $Y \cup C$  is at most 20 times the total flow routed by the phase 1 algorithm. ◀

This concludes our main result Theorem 2 and also implies the following corollary.

► **Corollary 15.** *Let  $G$  be an undirected, (integer) edge capacitated treewidth-2 graph and  $\{(s_i, t_i)\}_{i=1}^k$  be the source-sink pairs. Our algorithm gives an 80-approximation for computing a multicut w.r.t. the source-sink pairs.*

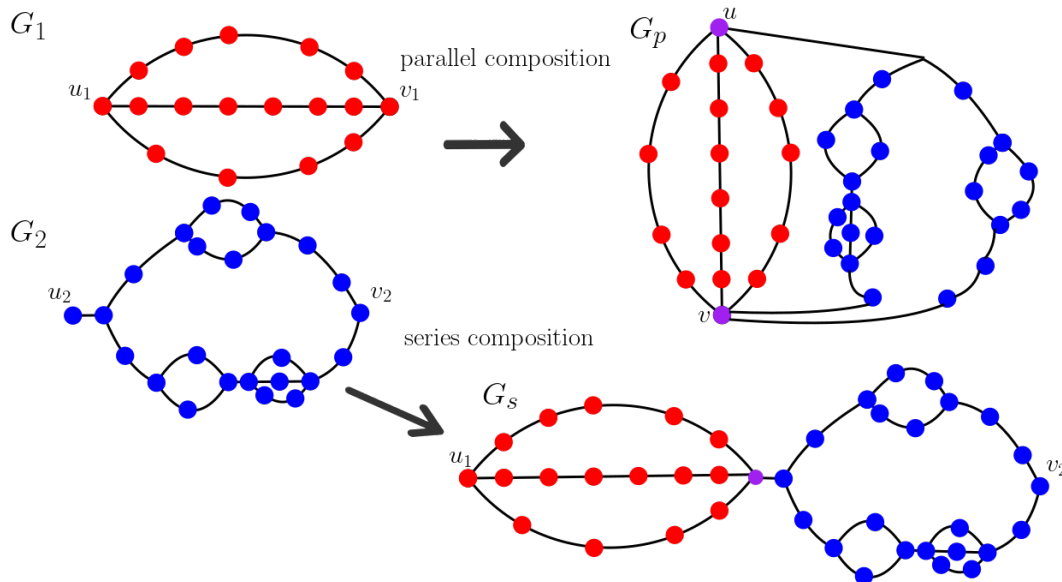
---

## References

- 1 Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 79–88, 2014.
- 2 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. An  $O(\sqrt{n})$  approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of computing*, 2(1):137–146, 2006.
- 3 Chandra Chekuri, Guylain Naves, and F Bruce Shepherd. Maximum edge-disjoint paths in k-sums of graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 328–339. Springer, 2013.
- 4 Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 233–242. IEEE, 2012.
- 5 Denis Cornaz. Max-multiflow/min-multicut for G+H series-parallel. *Discret. Math.*, 311(17):1957–1967, 2011. doi:10.1016/j.disc.2011.05.025.
- 6 Alina Ene, Matthias Mnich, Marcin Pilipczuk, and Andrej Risteski. On routing disjoint paths in bounded treewidth graphs. *arXiv preprint*, 2015. arXiv:1512.01829.

- 7 Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. In *Classic papers in combinatorics*, pages 243–248. Springer, 2009.
- 8 Naveen Garg and Nikhil Kumar. Dual half-integrality for uncrossable cut cover and its application to maximum half-integral flow. In *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173, page 55. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 9 Naveen Garg, Nikhil Kumar, and András Sebő. Integer plane multiflow maximisation: Flow-cut gap and one-quarter-approximation. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 144–157. Springer, 2020.
- 10 Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.
- 11 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
- 12 T Chiang Hu. Multi-commodity network flows. *Operations research*, 11(3):344–360, 1963.
- 13 Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 682–690. ACM, 1993.
- 14 Takao Nishizeki, Jens Vygen, and Xiao Zhou. The edge-disjoint paths problem is np-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115(1-3):177–186, 2001.
- 15 Loïc Seguin-Charbonneau and F Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 200–209. IEEE, 2011.
- 16 Eva Tardos and Vijay V Vazirani. Improved bounds for the max-flow min-multicut ratio for planar and  $k$ ,  $r$ -free graphs. *Information Processing Letters*, 47(2):77–80, 1993.
- 17 Hui Zang, Jason P Jue, and Biswanath Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. *Optical networks magazine*, 1(1):47–60, 2000.

## A Figures



■ **Figure 6** Series and Parallel Compositions.

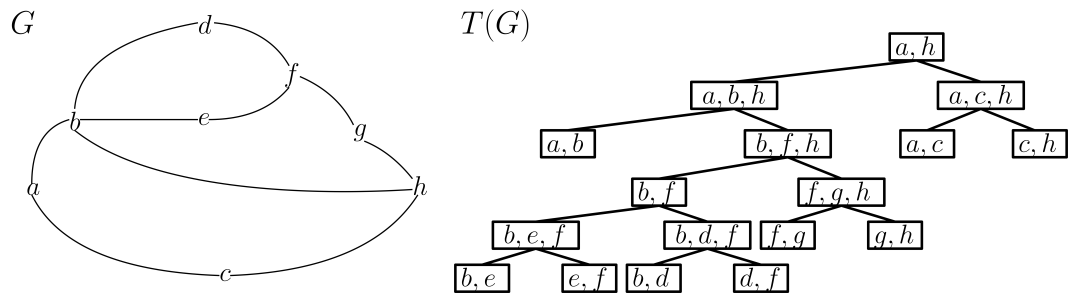


Figure 7 Series-Parallel Tree-Decomposition.